SYBASE iAnywhere

Ultra Light データベース管理とリファレンス

2009 年 2 月

バージョン 11.0.1

版権と商標

目次

はじめに

このマニュアルの内容

このマニュアルでは、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。

対象読者

このマニュアルは、Ultra Light リレーショナル・データベースのパフォーマンス、リソース効率、堅牢性、セキュリティを利用して、埋め込みデバイスやモバイル・デバイスのデータを格納、同期することを目的とするすべての開発者を対象にしています。

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。

- **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 リーダが必要です。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。

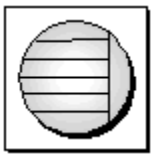
- **環境変数**　マニュアルでは、環境変数設定が引用されます。Windows のシェルでは、環境変数は構文 *%ENVVAR%* を使用して指定されます。UNIX のシェルでは、環境変数は構文 *$ENVVAR* または *${ENVVAR}* を使用して指定されます。

グラフィック・アイコン

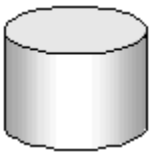
このマニュアルでは、次のアイコンを使用します。

- 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

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もありません。

iAnywhere のテクニカル・アドバイザとその他のスタッフは、時間のある場合にかぎりニュースグループでの支援を行います。こうした支援は基本的にボランティアで行われるため、解決策や情報を定期的に提供できるとはかぎりません。支援できるかどうかは、スタッフの仕事量に左右されます。

Ultra Light の概要

目次

Ultra Light は、SQL Anywhere の多くの特徴や機能を備えた、コンパクトなリレーショナル・データベースです。Ultra Light は、全社的なモバイル・データ管理用の SQL Anywhere ソリューションの一部として、またはスタンドアロンの組み込みソリューションの一部としてインストールできます。

Ultra Light には、コーポレート・データをモバイル化する機能が組み込まれています。Mobile Link クライアントとして配備されている場合は、業務に必要な情報を必要なときに安定して提供する、同期ソリューションを実装できます。ユーザは、コーポレート・ネットワークに直接アクセスできないときでも、必要なデータの記録やデータへのアクセスができます。

アプリケーション開発インタフェースにより、アプリケーションを作成し、ユーザに付加価値を提供することができます。

参照

- 「Mobile Link 同期の概要」『Mobile Link - クイック・スタート』
- 「Mobile Link クライアント」『Mobile Link - クライアント管理』
- Mobile Link - クイック・スタート
- Mobile Link - クライアント管理
- Mobile Link - サーバ管理

Ultra Light の機能比較

C/C++ バージョンでは、Ultra Light データベースおよび管理システムによるアプリケーション・サイズの追加分は 400 ～ 500 KB です。一方、SQL Anywhere のデータベース、データベース・サーバ、および同期クライアントによる追加分は 6 MB ほどになります。

機能	SQL Anywhere	Ultra Light	考慮事項
トランザクション処理、参照整合性、マルチテーブル・ジョイン	X	X	
トリガ、ストアド・プロシージャ、ビュー	X		
外部ストアド・プロシージャ (呼び出し可能な外部 DLL)	X		
組み込みの参照整合性とエンティティ整合性	X	X	削除と更新がカスケードされる宣言参照整合性は、Ultra Light データベースではサポートされていない機能です。ただし、同期中はこの目的で削除が自動的にカスケードされます。「外部キー循環に関連する同期の問題の防止」 142 ページを参照してください。
カスケード更新および削除	X	制限あり	
動的な複数データベースのサポート	X	X	Ultra Light エンジンと組み合わせた場合のみ
マルチスレッド・アプリケーションのサポート	X	X	
ローレベルのロック	X	X	

機能	SQL Anywhere	Ultra Light	考慮事項
XML のアンロードとロードのユーティリティ		X	Ultra Light では、XML のロードとアンロードを行うために別々の管理ツールを使用します。ランタイムには組み込まれていません。「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページと「Ultra Light データベースのアンロード・ユーティリティ (ulunload)」 298 ページを参照してください。
XML のエクスポートとインポートのユーティリティ	X		SQL Anywhere では、SQL 文を使用して XML にデータをエクスポートまたはインポートします。また、dbunload を使用して、データをエクスポートすることもできます。「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。
SQLX 機能	X		
SQL 関数	X	X	Ultra Light アプリケーションでは、すべての SQL 関数を使用できるわけではありません。サポートされていない関数を使用した場合は、「**Ultra Light では使用できない機能です。**」というエラーが発生します。「Ultra Light SQL 関数」 365 ページを参照してください。
SQL 文	X	制限あり	SQL Anywhere と比較すると、Ultra Light では SQL 文のスコープに制限があります。「Ultra Light SQL 文」 467 ページを参照してください。
統合 HTTP サーバ	X		
データベース・ファイルとネットワーク通信の強力な暗号化	X	X	
イベントのスケジュールと処理	X		
高パフォーマンス、セルフチューニング、コストベースのクエリ・オプティマイザ	X		Ultra Light にはクエリ・オプティマイザがありますが、SQL Anywhere のオプティマイザほど機能が豊富ではありません。したがって、Ultra Light のオプティマイザでは、複雑なクエリでは、SQL Anywhere オプティマイザほどパフォーマンスが上がらない可能性があります。

機能	SQL Anywhere	Ultra Light	考慮事項
複数のスレッド対応 API	X	X	Ultra Light には独特の柔軟なアーキテクチャがあるので、アプリケーション開発者は、変化する配備環境や複数の異なる配備環境を対象としたアプリケーションを作成できます。「Ultra Light プログラミング・インタフェースの選択」 24 ページを参照してください。
カーソルのサポート	X	X	「Ultra Light の制限事項」 7 ページを参照してください。
高度なキャッシュ管理システムによる動的なキャッシュのサイズ設定	X		Ultra Light ではキャッシュのサイズ設定は静的です。それでも、Ultra Light ではデータベースの起動時にキャッシュ・サイズを設定できるという拡張性があります。「Ultra Light CACHE_SIZE 接続パラメータ」 236 ページを参照してください。
システムまたはアプリケーションの障害後のデータベースのリカバリ	X	X	
バイナリ・ラージ・オブジェクト (BLOB) のサポート	X	X	Ultra Light では、BLOB のインデックス付けまたは比較を実行できません。
Windows パフォーマンス モニタの統合	X		
オンラインのテーブルとインデックスの断片化解除	X		
オンライン・バックアップ	X		
500 KB 程度の小さい占有容量		X	小型デバイスでは、比較的低速のプロセッサを使用していることがあります。Ultra Light では、これらのデバイスを対象としたアルゴリズムとデータ構造が使用されているので、Ultra Light はパフォーマンスが高く、メモリ使用量が少なくなっています。

機能	SQL Anywhere	Ultra Light	考慮事項
Palm OS をサポート		X	
デスクトップから Windows Mobile デバイスへの直接デバイス接続		X	SQL Anywhere データベースでは、Windows Mobile デバイスに配備するデータベースにデスクトップ接続できるようになるには、データベース・サーバが必要です。Ultra Light では、接続文字列のプレフィクスとして **WCE:¥** を使用するだけで済みます。「Windows Mobile」 50 ページを参照してください。
インデックスの使用による高パフォーマンスの更新と検索	X	X	Ultra Light には、テーブルの検索時にインデックスを使用するか、ローを直接スキャンするかを決定するメカニズムがあります。 また、インデックスをハッシュしてデータ検索を高速化することもできます。「Ultra Light max_hash_size 作成パラメータ」 201 ページを参照してください。
Oracle、DB2、Sybase Adaptive Server Enterprise、または SQL Anywhere との同期	X	X	
組み込みの同期		X	SQL Anywhere とは異なり、Ultra Light では、クライアント・エージェントによる同期を必要としません。同期は Ultra Light ランタイムに組み込まれているので、配備する必要があるコンポーネントが最小限に抑えられます。「Ultra Light クライアント」 131 ページを参照してください。
プロセス内実行		X	
計算カラム	X		
宣言されたテンポラリ・テーブル／グローバル・テンポラリ・テーブル	X		

機能	SQL Anywhere	Ultra Light	考慮事項
システム関数	X		Ultra Light では、プロパティ関数を含む SQL Anywhere のシステム関数はサポートされていません。Ultra Light アプリケーションにこれらの関数を含めることはできません。
timestamp カラム	X	X	SQL Anywhere の Transact-SQL のタイムスタンプ・カラムは DEFAULT TIMESTAMP がデフォルトで作成されます。 Ultra Light のタイムスタンプ・カラムは DEFAULT CURRENT TIMESTAMP がデフォルトで作成されます。したがって、Ultra Light では、ローの更新時にタイムスタンプが自動的に更新されません。
ユーザベースのパーミッションのスキームによるオブジェクトベースの所有権とアクセスの決定	X		Ultra Light は、認証システムの必要がない、単一ユーザのデータベース向けに設計されています。ただし、最大 4 つのユーザ ID とパスワードを設定し、認証のためだけに使用できます。これらのユーザは、すべてのデータベース・オブジェクトにアクセスできます。「Ultra Light ユーザ認証」 53 ページを参照してください。

Ultra Light の制限事項

Ultra Light の制限と SQL Anywhere の制限の比較については、「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。

統計情報	Ultra Light の最大値
接続数／データベース	単一スレッド・アプリケーションで最大 14
同時に開いている接続数	Palm OS の場合は最大 8 その他のすべての OS の場合は最大 32
アプリケーションあたりの同時接続数の合計	Palm OS の場合は最大 16 その他のすべての OS の場合は最大 64
SQL Communication Area	最大 63
ファイルベースの永続ストア (データベース・サイズ)	2 GB ファイル、または OS の制限によるファイル・サイズ
Palm Computing Platform データベース・サイズ	128 MB (主記憶装置) 2 GB (拡張カード・ファイル・システム)
テーブルあたりのロー数	最大 1600 万 [1]
データベースあたりのロー数	永続ストアによって制限される。
テーブルの大きさ	データベース・サイズによってのみ制限される。
データベースあたりのテーブル数	データベース・サイズによってのみ制限される。
テーブルあたりのカラム数	ロー・サイズはページ・サイズによって制限されるため、テーブルあたりのカラム数の上限は、実際にはページ・サイズによって決まる。実際には、4000 よりもかなり小さいのが普通。
テーブルあたりのインデックス数	データベース・サイズによってのみ制限される。
トランザクションあたりの参照テーブル数	制限なし

統計情報	Ultra Light の最大値
ストアド・プロシージャの長さ	不適用。
データベースあたりのストアド・プロシージャ数	不適用。
データベースあたりのトリガ数	不適用。
ネスト	不適用。
パブリケーション数	32 パブリケーション
ページ・サイズ	16 KB
ロー・サイズ	パックされた各ローの長さはページ・サイズ以下である必要があります。「ローのパックとテーブル定義」 74 ページを参照してください。 文字列がカラム・サイズよりも短い場合は埋め込みなしで格納されます。long binary と long varchar として宣言されたカラムは個別に格納されるため、除外されます。
接続ごとのカーソル数	Ultra Light データベースへの特定の接続で使用できるカーソルの最大数は 64 です (すべてのプラットフォーム)。
long binary と long varchar のサイズ	データベース・サイズによってのみ制限される。

[1] 場合によっては、ローに対する変更 (削除および更新) とその他のステータス情報がロー・データとともに保持されます。この情報により、変更が同期されます。必然的に、同期間のテーブルのトランザクション数や、テーブルの同期の有無によっては、実際のローの制限が 1600 万より小さくなります。「Ultra Light でのトランザクション処理」 16 ページを参照してください。

Ultra Light のデータ・アーキテクチャ

Ultra Light はプラットフォームに依存しないモバイル・データベースです。Ultra Light は、携帯電話、ハンドヘルド・コンピュータ、埋め込みデバイスなどの小型デバイス用のカスタム・ソリューションを作成できるように設計されています。

Ultra Light は完全なデータベース管理システムであり、次の機能を備えています。

- **開発レイヤ**　Ultra Light ではさまざまなプログラミング・インタフェースがサポートされているので、特定の配備プラットフォーム、開発ツール、または IT インフラストラクチャ製品に縛られることがありません。

  API の選択については、「Ultra Light プログラミング・インタフェースの選択」 24 ページを参照してください。

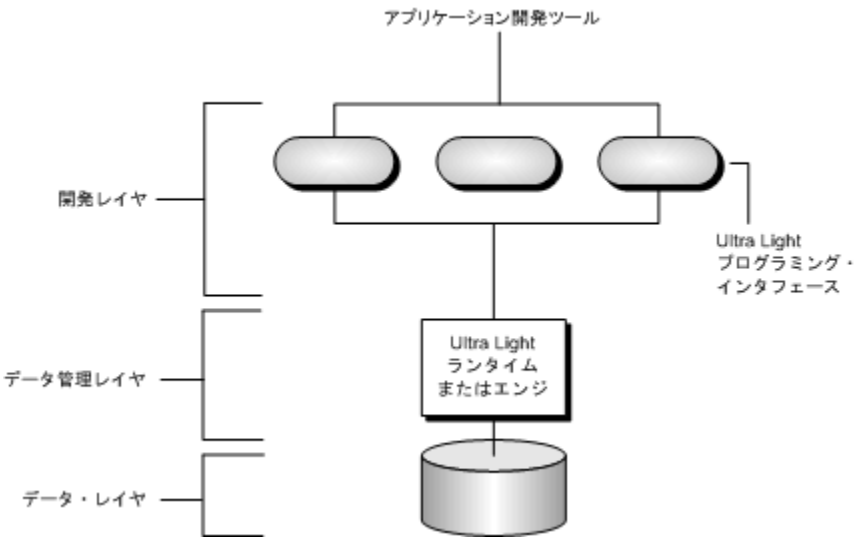
  Ultra Light プロジェクトの管理に役立つように、Ultra Light では、一連の管理ツールによって開発がサポートされます。これらのツールは、コマンド・ライン・ユーティリティとして、または Sybase Central の Ultra Light プラグインでウィザードとして実行できます。

- **データ管理レイヤと同期クライアント**　プロセス内ライブラリ (ランタイム) または独立プロセス (エンジン) を使用して、Ultra Light データベースに接続できます。どちらのプロセスでも、接続とデータ・アクセス要求を制御します。また、Ultra Light データベースと企業のバックエンド・システムとを結ぶ双方向の同期クライアントが組み込まれています。

  プロセスの選択については、「Ultra Light データ管理コンポーネントの選択」 23 ページを参照してください。

- **データ・レイヤ**　このレイヤは、ファイルまたは一連のデータ・レコード (Palm OS の場合) として保存されるローカル・データ・レポジトリです。「Ultra Light の保管方法とファイルの命名規則」 10 ページを参照してください。

次の図は、データ、管理、開発の各レイヤを示します。



Ultra Light の保管方法とファイルの命名規則

デバイスのファイル管理の要件によって、Ultra Light データベースの保管方法と命名規則が決まります。ほとんどのプラットフォームでは、従来のファイルベースの保管方法が使用されますが、Palm OS のレコードベース・ストアなどでは、データベースの保管方法が異なります。ただし、Ultra Light データベースは一般的にはファイルとして見なされます。プラットフォームの制限によっては、開発用デスクトップにデータベースを作成してから、そのデータベースを 1 つまたは複数のプラットフォームに配備できます。

参照

- 「Ultra Light 接続パラメータでのファイル・パスの指定」 49 ページ

Ultra Light データベース・スキーマ

データベースの論理的なフレームワークを「スキーマ」といいます。Ultra Light では、スキーマは、Ultra Light データベースのメタデータを含むシステム・テーブルのカタログとして管理されます。システム・テーブルには、次のメタデータが格納されています。

- インデックス定義。「sysindex システム・テーブル」 309 ページと「sysixcol システム・テーブル」 311 ページを参照してください。
- テーブル定義。「systable システム・テーブル」 307 ページを参照してください。
- カラム定義。「syscolumn システム・テーブル」 308 ページを参照してください。
- パブリケーション定義。「syspublication システム・テーブル」 312 ページと「sysarticle システム・テーブル」 313 ページを参照してください。
- ユーザ名とパスワード。「sysuldata システム・テーブル」 314 ページを参照してください。

DDL 文を使用したスキーマ変更

データベースのスキーマは、適切なデータ定義言語 (DDL) 文を使用して変更できます。変更の必要な箇所が多い場合は、ALTER DATABASE SCHEMA FROM FILE 文を使用し、SQL スクリプトを使用してスキーマ定義を変更できます。

スキーマの変更には時間がかかる場合があります。たとえば、カラム型が変更された場合は、関連するテーブルにあるすべてのローを更新する必要があります。DDL 文は、次の場合に正常に実行されます。

- コミットされていないトランザクションがない。
- スキーマのその他のアクティブな使用 (同期、準備されたが解放されていない文、実行中のデータベース操作など) がない。

このいずれかがあった場合、DDL 文は失敗します。DDL 文を実行すると、DDL 文によるスキーマの変更が完了するまで、その他のデータベースの操作はブロックされます。

参照

- 「Ultra Light ALTER DATABASE SCHEMA FROM FILE 文」 470 ページ
- 「Ultra Light SQL 文」 467 ページ

Ultra Light のテンポラリ・ファイル

データベース・ファイルに加えて、Ultra Light では、データベース操作時に**テンポラリ・ファイル**が作成され、管理されます。このファイルの操作や管理は必要ありません。

テンポラリ・ファイルは Ultra Light によってのみ使用され、Ultra Light データベース自体と同じファイル・パス (存在する場合) に作成されます。テンポラリ・ファイルのファイル名はデータベースと同じですが、次の点が異なります。

- **ファイルベース・プラットフォームの場合**　ファイルの拡張子にチルダが追加されます。たとえば、サンプル・データベース *CustDB.udb* を実行すると、このファイルと同じディレクトリに *CustDB.~db* というテンポラリ・ファイルが作成されます。
- **レコードベース・プラットフォームの場合**　ファイル名の末尾にチルダが追加されます。たとえば、*CustDB.udb* が Palm OS でレコードベースのファイルとして存在した場合、テンポラリ・ファイルは *CustDB.udb~* として作成されます。

ヒント
Ultra Light が実行中でなければ、テンポラリ・ファイルを削除してもデータは失われません。テンポラリ・ファイルには、セッション間で必要な情報は含まれません。

Ultra Light のテンポラリ・テーブル

テンポラリ・テーブルは、アクセス・プランによって、実行中にデータを格納するために、一時的なテーブル、つまりテンポラリ・ワーク・テーブルとして使用されます。このテーブルが存在するのは、アクセス・プランの実行中のみです。一般的に、テンポラリ・テーブルが使用されるのは、サブクエリをアクセス・プランの早い段階で評価する必要がある場合、または使用可能なメモリに中間結果が収まらない場合です。

テンポラリ・テーブルのデータは、単一の接続に対してだけ保持されます。テンポラリ・テーブルはローとカラムで構成されます。各カラムは電話番号や名前など情報の種類を特定し、各ローはデータのエントリを保持します。

テンポラリ・テーブルの使用を避けるには、ORDER BY 句または GROUP BY 句で使用されるカラムでインデックスを使用します。

参照

- 「Ultra Light のテンポラリ・ファイル」 11 ページ
- 「Ultra Light のパフォーマンスと最適化」 113 ページ
- 「実行プランを確認する場合」 360 ページ

Ultra Light のトランザクションとステータスの管理

Ultra Light では、データのほかにステータス情報をデータベースに保持します。ステータス情報を追跡および保持することで、次の管理を行っています。

- 同時接続数。Ultra Light で必要に応じてリソースを共有できるようにするためです。「Ultra Light での同時実行性」 12 ページを参照してください。
- 同期の進行状況のカウント。同期が正常に実行されるかを確認するためです。「Ultra Light の組み込みの同期機能」 132 ページを参照してください。
- ローのステータス。同期間でのデータの変更内容を追跡することでデータの整合性を維持するためです。「Ultra Light でのローのステータス」 13 ページを参照してください。
- トランザクション。データをいつ、どのようにコミットするかを決定するためです。Ultra Light では、1 つのトランザクションは完全に処理されるかまったく処理されないかのどちらかです。「Ultra Light でのトランザクション処理」 16 ページを参照してください。
- リカバリとバックアップの情報。オペレーティング・システムのクラッシュ、およびストレージ・カードの取り出しやデバイスのリセットなどの Ultra Light の実行中のエンド・ユーザ操作からデータを保護するためです。「Ultra Light でのデータのバックアップとリカバリ」 15 ページを参照してください。

Ultra Light での同時実行性

同時実行性とは、Ultra Light が複数の同時接続を許可することによってリソースを共有する形態のことです。Ultra Light では、次の方法を使用して同時実行性を管理しています。

- **複数のデータベース**　1 つの Ultra Light アプリケーションは、複数のデータベースに対する接続を開くことができます。Ultra Light で開くことができるデータベースは、Palm OS では最大 8 個、それ以外のプラットフォームでは最大 32 個です。
- **複数のアプリケーション**　Ultra Light データベースを開くことができるのは、一度に 1 つのプロセスだけです。複数のアプリケーション間での同時実行性をサポートする計画の場合は、データ管理コンポーネントとして Ultra Light エンジンを選択してください。「Ultra Light データ管理コンポーネントの選択」 23 ページを参照してください。
- **複数のスレッド**　Ultra Light では、マルチスレッド・アプリケーションがサポートされています。複数のスレッドを使用し、各スレッドが同じまたは異なるデータベースに接続できる、単一のアプリケーションを作成できます。

  ランタイムでデータベースを管理している場合は、Ultra Light がサポートする同時接続数の上限を超えないよう注意してください。上限は次のとおりです。

  ○ Palm OS の場合は 16

  ○ その他のすべてのプラットフォームの場合は 64

Ultra Light エンジンでデータベース接続を管理している場合は、一般的に使用できる SQLCA の数は 128 に制限されています。ただし、Ultra Light.NET API の実装では、この上限が、128 から Ultra Light .NET クライアントの実行数を差し引いた数に減少します。

- **複数のトランザクション／要求**　各接続は、進行中のトランザクションを一度に 1 つ使用できます。トランザクションは、1 つの要求または複数の要求で構成されます。トランザクションの実行中に行われたデータの変更は、トランザクションがコミットされるまでデータベースで永続化されません。トランザクションで行われたデータの変更は、すべてコミットされるか、すべてロールバックされるかのどちらかです。「Ultra Light でのトランザクション処理」 16 ページを参照してください。
- **同期**　アップロードおよびダウンロードでは、読み込み／書き込みアクセスが許可されます。ただし、アプリケーションがローを変更してからダウンロードによってこのローが変更されようとすると、ダウンロードが失敗し、ロールバックが行われます。同期時のデータへのアクセスを無効にするには、Disable Concurrency 同期パラメータを使用します。「Additional Parameters 同期パラメータ」 159 ページを参照してください。

  同期に失敗した場合、Ultra Light では、すべてのプラットフォームで再開可能なダウンロードがサポートされます。「失敗したダウンロードの処理」『Mobile Link - サーバ管理』を参照してください。

参照

- 「Ultra Light クライアント」 131 ページ
- 「Additional Parameters 同期パラメータ」 159 ページ

Ultra Light でのローのステータス

ローのステータス情報の管理は、Ultra Light の強力な機能の 1 つです。テーブルとローのステータスの追跡は、データ同期時に特に重要です。

データへの変更内容

Ultra Light データベースのローのステータスを記録するために、内部マーカが使用されます。ローのステータスは、トランザクション処理、リカバリ、同期を制御します。ローを挿入、更新、または削除すると、その操作内容、およびその操作を実行した接続を反映するようにローのステータスが変更されます。トランザクションがコミットされると、トランザクションに関係のあるすべてのローのステータスに、コミットの内容が反映されます。コミット中に予期せぬ障害が発生すると、トランザクション全体がロールバックされます。次のリストは、その動作をまとめたものです。

- **削除が実行された場合**　対象の各ローが削除されたことを反映してステータスが変更されます。削除を取り消すには、ローを元のステータスにリストアします。
- **削除がコミットされた場合**　対象のローがメモリから削除されない場合もあります。一度も同期が行われていないローは削除されます。同期が行われたローは削除されません。削除操作を行うには、まず統合データベースに同期される必要があるからです。次の同期が実行された後、ローはメモリから削除されます。

- **ローが更新された場合**　ローの新しいバージョンが作成されます。古いローと新しいローのステータスが変更され、古いローは表示されなくなり、新しいローが表示されます。
- **ローの更新がコミットされた場合**　トランザクションがコミットされると、トランザクションに関係のあるすべてのローのステータスに、コミットの内容が反映されます。更新が同期されると、ローの古いバージョンと新しいバージョンの両方で競合が検出され、解決が行われます。その後、古いローがデータベースから削除され、新しいローが通常のローになります。
- **ローが追加された場合**　ローがデータベースに追加され、コミット未実行のマークが付きます。
- **追加されたローがコミットされた場合**　ローにコミット済みのマークが付き、統合データベースとの同期が必要であると通知されます。

参照

- 「Ultra Light でのデータのバックアップとリカバリ」 15 ページ
- 「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」 125 ページ
- 「Ultra Light でのトランザクション処理」 16 ページ

Ultra Light データベースの検証

検証は、データベース上に他のアクティビティがないときに行うことをおすすめします。

Ultra Light データベースは、次のいずれかの方法で検証できます。

- Sybase Central の**データベース検証ウィザード**
- ValidateDatabase 関数の呼び出し
- ulvalid ユーティリティ
- ValidateDatabase メソッド

警告
データベースの検証は、接続がデータベースに変更を加えていない場合に実行してください。これを守らなかった場合、実際に破損していなくても、データベースが破損したことを示すエラーがレポートされる可能性があります。

♦ **データベースを検証するには、次の手順に従います (Sybase Central の場合)。**

1. Sybase Central の左側のウィンドウ枠で、Ultra Light データベースを選択します。
2. **[ファイル] - [データベースの検証]** を選択します。
3. **データベース検証ウィザード**の指示に従います。

ヒント
Sybase Central では、次の方法で**データベース検証ウィザード**を利用することもできます。

- Ultra Light データベースを右クリックし、**[データベースの検証]** を選択する。
- Ultra Light データベースを選択し、**[ツール] - [Ultra Light 11] - [データベースの検証]** を選択する。

参照

- 「ValidateDatabase メソッド」 『Ultra Light - .NET プログラミング』
- 「ValidateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- 「Ultra Light データベース検証ユーティリティ (ulvalid)」 303 ページ

Ultra Light でのデータのバックアップとリカバリ

Ultra Light データベースを使用しているアプリケーションが予期せずに停止した場合、アプリケーションを再起動すると、Ultra Light データベースは自動的に一貫性が保たれた状態にリカバリされます。障害が発生する前にメモリにフラッシュ済みのすべてのコミットされたトランザクションは、Ultra Light データベースに保持されます。障害発生時にフラッシュされていないすべてのトランザクションは、ロールバックされます。

Ultra Light では、リカバリの実行にトランザクション・ログを使用しません。各ローのステータス情報を保存して、リカバリ時のローの処理を決定します。「Ultra Light でのローのステータス」 13 ページを参照してください。

バックアップ

Ultra Light は、システム障害に対する保護機能は備えていますが、メディア障害に対する保護機能はありません。Ultra Light アプリケーションのバックアップ作成に最適な方法として、統合データベースとの同期を行うことが挙げられます。Ultra Light データベースをリストアするには、空のデータベースを作成し、統合データベースと同期してデータを移植します。

Ultra Light の小規模な配備環境では、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することで手動バックアップを実行できます。

注意
Palm OS の場合は、PALM_ALLOW_BACKUP 接続パラメータを true に設定することもできます。この設定により、データベースを HotSync を使用してバックアップできます。「Ultra Light PALM_ALLOW_BACKUP 接続パラメータ」 251 ページを参照してください。

参照

- 「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」 125 ページ

Ultra Light でのトランザクション処理

トランザクションは、自動的に実行される処理の論理セットです。つまり、トランザクションのすべての処理がデータベースに格納されるか、トランザクションの処理が 1 つもデータベースに格納されないかのどちらかです。Ultra Light ランタイムへの Ultra Light アプリケーションのアクセスは直列化されます。複数のトランザクションを同時に開くことは可能ですが、Ultra Light では一度に 1 つのトランザクションしか処理されません。これにより、アプリケーションでは以下の現象が発生しません。

- トランザクションがブロックされること (デッドロック)。Ultra Light が既存のロー・ロックに基づいて要求をブロックすることはありません。このような事態になった場合、Ultra Light はすぐにエラーを返します。
- 未処理の変更を上書きすること。トランザクションは、他のトランザクションの未処理の変更を上書きすることはできません。トランザクションによってローが変更されると、Ultra Light は、トランザクションが**コミット**または**ロールバック**されるまでこのローをロックします。ロックによって、他のトランザクションはローを読み込むことはできても、ローを変更できなくなります。

たとえば、A と B という 2 つのアプリケーションが 1 つのデータベースの同じローを読み取っており、どちらもそのデータに基づいて当該ローのカラムの 1 つに入る新しい値を計算するとします。A がローを新しい値で更新し、B が同じローを変更しようとした場合、B はエラーになります。ロックされたローを変更しようとするとエラー SQLCODE SQLE_LOCKED が設定され、削除されたローを変更しようとするとエラー SQLE_NOTFOUND が設定されます。したがって、データの変更を試行した後には SQLCODE 値をチェックするようアプリケーションをプログラムしてください。

エラー処理方法の詳細については、次の項を参照してください。

- Ultra Light for C++：「エラー処理」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「エラー処理」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「エラー処理」『Ultra Light - M-Business Anywhere プログラミング』

プログラミングのヒント

C++ API を除くすべての Ultra Light API は**オートコミット**モードで動作します。一部の API はデフォルトでオートコミットを使用します。

つまり、このデフォルトでは、各トランザクションは操作が終わるたびに自動的にコミットされます。これらのインタフェースのいずれかを使用している場合、複数操作トランザクションを使用するには、オートコミットをオフに設定します。オートコミットをオフに設定する方法は、使用しているプログラミング・インタフェースによって異なります。ほとんどのインタフェースでは、これは接続オブジェクトのプロパティです。

次の項を参照してください。

- Ultra Light.NET：「トランザクションの管理」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「トランザクションの管理」『Ultra Light - M-Business Anywhere プログラミング』

Ultra Light の独立性レベル

独立性レベルは、あるトランザクションの操作が、同時に処理されている別のトランザクションの操作からどの程度参照できるかを定義します。Ultra Light では、オートコミット・モードでの接続には、デフォルトの独立性レベルであるコミットされた呼び出しが使用されます。.NET の場合、パラメータを指定しないで ULConnection.BeginTransaction を呼び出すことによって作成された新しいトランザクションでは、ReadCommitted がデフォルトの独立性レベルになります。Ultra Light のデフォルトの独立性レベルは、データの一貫性を維持しながら最高のパフォーマンスを提供します。

独立性レベルが ReadCommitted の場合は、次のようになります。

- ダーティ・リードは発生しない。
- 読み込みロックは適用されない。
- コミットされていない削除が参照できる。
- 繰り返し不可能読み出しと幻ローが許可される。
- トランザクション中にデータが変更されないという保証はない。

独立性レベルが ReadUncommitted の場合は、次のようになります。

- ダーティ・リードが許可される。
- 読み込みロックは適用されない。
- 繰り返し不可能読み出しと幻ローが許可される。
- 同時トランザクションがローを変更しないこと、またはローに対しての変更がロールバックされないことは保証されない。

独立性レベルを ReadCommitted から ReadUncommitted に変更することができます。Ultra Light C++ の場合、独立性レベルを変更するには SetDatabaseOption メソッドを使用します。Ultra Light.NET 2.0 の場合、独立性レベルが ReadUncommitted のトランザクションを作成するには ULConnection.BeginTransaction を呼び出します。

注意

.NET の場合、トランザクションがアクティブになっているときに SetDatabaseOption を実行することはおすすめしません。もし実行すると、接続の独立性レベルは変更されますが、ULTransaction.IsolationLevel は更新されません。

トランザクションが進行中のときには、独立性の変更に SetDatabaseOption は使用しないでください。予期しないエラーが発生する可能性があります。

独立性レベルの副次的影響

Ultra Light は、デフォルトでは独立性レベル 0 で動作するため、次のような副次的影響が発生する可能性があります。

- SELECT 文を実行するときは、ロック・オペレーションは必要ありません。
- アプリケーションは、コミットされていないデータにアクセスできます (ダーティ・リード)。このシナリオでは、コミットされていないデータベースのローにアプリケーションがアクセスできるため、別のトランザクションによりロールバックされる可能性が発生します。これにより幻ローが発生します。幻ローとは元のクエリの後に追加されたローのことで、これにより、繰り返されて重複したクエリで返された結果セットの内容に相違が生じます。

  ダーティ・リードの影響を示すチュートリアルについては、「チュートリアル：ダーティ・リード」『SQL Anywhere サーバ - SQL の使用法』を参照してください。幻ローを示すチュートリアルについては、「チュートリアル：幻ロー」『SQL Anywhere サーバ - SQL の使用法』を参照してください。
- アプリケーションは、繰り返し不可能読み出しを実行できます。このシナリオでは、アプリケーションがデータベースからローを読み出し、他の操作の実行に移ります。そこへ、別のアプリケーションがローの更新や削除を行い、変更をコミットします。最初のアプリケーションが元のローを再度読み出すと、更新された情報が取得されるか、元のローが削除されたことを発見します。

  繰り返し不可能読み出しの影響を示すチュートリアルについては、「チュートリアル：繰り返し不可能読み出し」『SQL Anywhere サーバ - SQL の使用法』を参照してください。

例

たとえば、A と B という 2 つの接続があり、それぞれにトランザクションがあるとします。

1. 接続 A がクエリの結果セットを処理する場合、Ultra Light によって、現在のローのコピーが**フェッチ**されてバッファに格納されます。

   注意
   ローを読み込んだりフェッチしたりしても、ローはロックされません。接続 A がローをフェッチしても変更はしない場合、接続 B はこのローを変更できます。
2. 接続 A は現在のローを変更するときに、バッファ内のコピーを変更します。接続 A が Update メソッドを呼び出すか結果セットを閉じると、バッファ内のコピーはデータベースに書き戻されます。
3. このローに対して書き込みロックが掛けられ、他のトランザクションがこのローを変更できなくなります。この変更は、接続 A がコミットを実行するまで、コミットされないままになります。
4. 変更によっては、接続 B が現在のローをフェッチすると、次の現象が発生する可能性があります。

接続 A による変更	結果 [1]
ローの削除	接続 B は結果セットの次のローを取得します。
ローの変更	接続 B はローの最新のコピーを取得します。

[1] 接続 A および B で使用されたクエリには、テンポラリ・テーブルは含まれていません。テンポラリ・テーブルは、他の副次的な影響を与える可能性があります。

参照

- 「BeginTransaction() メソッド」 『Ultra Light - .NET プログラミング』
- 「BeginTransaction(IsolationLevel) メソッド」 『Ultra Light - .NET プログラミング』
- 「SetDatabaseOption メソッド」 『Ultra Light - .NET プログラミング』

Ultra Light ソリューションの実装

目次

Ultra Light ソリューションを実装する際は、次の点を検討します。

- Ultra Light データベースに接続する必要があるアプリケーション数。同時接続数によって、Ultra Light プロセス内ランタイムと Ultra Light エンジンのどちらを使用するかが決まります。それぞれの違いについては、「Ultra Light データ管理コンポーネントの選択」 23 ページを参照してください。
- サポートするプラットフォーム。ここで選択したプラットフォームによって、アプリケーションのプログラミングに使用できる API が影響を受ける場合があります。「Ultra Light プログラミング・インタフェースの選択」 24 ページを参照してください。
- データベースを実行するプラットフォーム。ファイル・フォーマットは統合されているので、複数のプラットフォームで実行できるデータベースを作成できる場合があります。「Ultra Light データベースの作成と設定」 29 ページを参照してください。

> **ヒント**
> 複数のプラットフォームに適した 1 つのファイル・フォーマットを作成する必要がある場合は、Sybase Central の**データベース作成ウィザード**を使用すると、これが可能かどうかを確認できます。

Ultra Light がサポートしているプラットフォームとプロトコル

TCP/IP、HTTP、HTTPS いずれかのネットワーク・プロトコルを使用して、Ultra Light データベースのデータを中央の統合データベースと同期できます。

どのプラットフォームでどのネットワーク・プロトコル (ストリーム) がサポートされているかについては、http://www.ianywhere.jp/developers/technotes/os_components_1101.html を参照してください。

参照

- 「Ultra Light クライアント」 131 ページ
- 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 182 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ

Ultra Light データ管理コンポーネントの選択

Ultra Light では、占有容量が少ないリレーショナル・データベース・ソリューションを構築できます。このとき、別個のデータベース・サーバを設定する追加オーバヘッドは必要ありません。ただし、Ultra Light プログラミング・インタフェースでは、次の 2 種類のライブラリのうちの 1 つを使用します。

Ultra Light インプロセス・ランタイム・ライブラリ

Ultra Light では、ランタイムとアプリケーションは一体のプロセスであり、データベースはアプリケーションに固有のものとなります。すべてのプラットフォームで、ランタイムによって Ultra Light のデータベースと組み込みの同期処理が管理されます。Ultra Light ランタイムは、一度に最大 14 のデータベースを管理できます。ただし、一度にデータベースにアクセスできるアプリケーションは 1 つのみです。

ランタイムにリンクするには、エンジンとは異なる特定のインポート・ライブラリ／DLL ペアを使用する必要があります。詳細については、「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。TLS 対応の同期が必要な場合は、他のライブラリも必要になります。「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページを参照してください。

Ultra Light データベース・エンジン・クライアント・ライブラリ

Ultra Light エンジンが使用可能なプラットフォームは、Windows デスクトップと Windows Mobile のみです。エンジンは別個の実行プログラムであり、データベース／アプリケーションとの同時接続を実現します。どちらで実行するアプリケーションも、Ultra Light エンジンを使用するときにはクライアント・ライブラリを使用する必要があります。クライアント・ライブラリによって、各アプリケーションが Ultra Light エンジンと通信できます。Ultra Light エンジンには、Ultra Light ランタイムよりも多くのシステム・リソースが必要なので、パフォーマンスが低下する可能性があります。

エンジンにリンクするには、ランタイムとは別のインポート・ライブラリ／DLL ペアを使用する必要があります。TLS 対応の同期、または AES FIPS データベース暗号化が必要な場合は、他のライブラリも必要になります。「AES_FIP データベース暗号化を使用した Ultra Light の配備」 61 ページと「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページを参照してください。

注意
Ultra Light エンジンがサポートする同時接続数の上限を超えないよう注意してください。

参照

- 「Ultra Light エンジンユーティリティ (uleng11)」 275 ページ
- 「Ultra Light での同時実行性」 12 ページ
- 「Ultra Light の機能比較」 2 ページ
- 「Ultra Light の制限事項」 7 ページ

Ultra Light プログラミング・インタフェースの選択

すべての Ultra Light API は、中核となるデータベース機能を公開します。次の API の一部は開発環境と統合されており、プログラミング・タスクを簡単にします。

- C/C++ インタフェース
- C/C++ 用 Embedded SQL
- Ultra Light.NET (C# または VB.NET)
- M-Business Anywhere (JavaScript)

Ultra Light API には、単純なテーブルベースのデータ・アクセス・インタフェースから、複雑なクエリ用の動的 SQL まで、さまざまなデータ・アクセス・モデルがあります。このように Ultra Light のアーキテクチャは柔軟なので、アプリケーション開発者はさまざまな配備環境向けのアプリケーションを作成できます。

注意
このバージョンの Ultra Light では、Pocket Builder をサポートしていません。Sybase PocketBuilder は、SQL Anywhere に含まれていません。詳細については、Sybase にお問い合わせください (http://www.sybase.com/products/developmentintegration/pocketbuilder を参照)。

参照

- Ultra Light.NET：Ultra Light - .NET プログラミング
- Ultra Light for C/C++ と Embedded SQL：Ultra Light - C/C++ プログラミング
- Ultra Light for M-Business Anywhere：Ultra Light - M-Business Anywhere プログラミング

♦ プログラミング・インタフェースを選択するには、次の手順に従います。

1. 1 つまたは複数のターゲット・プラットフォームを選択します。Ultra Light では、Palm OS、Windows Mobile、Windows XP、Windows XP Embedded、Java OS がサポートされています。
2. サポートするプラットフォームごとに、API でそのプラットフォームがサポートされているかどうかを確認します。API によってサポートされているプラットフォームが異なります。プラットフォームを問わない開発を行う場合は、すべてのターゲットをサポートする API を選択します。

   このサポート対応表を使用すると、選択できる開発のオプションを簡単に特定できます。

配置ターゲット	Ultra Light for C/C++ と Embedded SQL	Ultra Light.NET[1]	Ultra Light for M-Business Anywhere[2]
Palm OS	バージョン 4 以降	なし	なし
Windows Mobile	CE 5.0 以降	CE 5.0 以降と .NET Compact Framework 2.0	バージョン 3.0 以降

配置ターゲット	Ultra Light for C/C++ と Embedded SQL	Ultra Light.NET[1]	Ultra Light for M-Business Anywhere[2]
Windows XP Embedded	サポート	.NET Framework 2.0	バージョン 5.0 以降
Java	Java SE (1.5 以降) RIM BlackBerry OS (4.1 以降)	なし	なし

[1] Microsoft Visual Studio.NET の拡張としての開発。ドライバは、ADO.NET バージョン 2.0 をサポートする。

[2] JavaScript を使用した Ultra Light プログラミングのブラウザベースの配備

3. 次の要件を検討し、選択した内容を最終決定します。

   SQL Anywhere の互換性　SQL Anywhere とデータベースの互換性が問題であれば、次の点を検討します。

   - SQL Anywhere の Embedded SQL のサポートには、Ultra Light と SQL Anywhere の各データベースに共通のプログラミング・インタフェースがあります。
   - ADO.NET には、Ultra Light のコンポーネントと SQL Anywhere で共有される共通のプログラミング・モデルがあります。

   特に、両方のデータベースを使用できる Windows Mobile などのプラットフォームでは、共通のインタフェースを使用できれば便利です。Ultra Light から SQL Anywhere データベースに移行する必要がある場合、アプリケーションの移行作業をより簡単にするために、Embedded SQL または ADO.NET を使用してください。

   ヒント
   共通のインタフェースが存在する場合でも、アプリケーションの作成時に抽象データ・アクセス・レイヤを作成すると開発に役立ちます。

   単純化された配備　Ultra Light 配備の単純化が問題であれば、M-Business Anywhere API を使用したプログラミングを検討します。エンドユーザは、Ultra Light アプリケーションとデータベースを同時にダウンロードできます。

   アプリケーション・サイズ　占有領域ができるだけ小さいアプリケーションを作成することが重要な場合は、C/C++ API を使用してアプリケーションをプログラミングします。このようなアプリケーションは、通常、高パフォーマンスを実現しつつ、アプリケーション・ファイル・サイズが抑えられます。

   アプリケーションのパフォーマンス　各 API で実現されるパフォーマンスは異なります。たとえば、Ultra Light.NET のパフォーマンスは Ultra Light C++ のパフォーマンスを下回る場合があり、Ultra Light Embedded SQL のパフォーマンスは Ultra Light C++ を上回る場合があります。

Ultra Light データベースの使用

この項では、Ultra Light データベースの作成方法と使用について説明します。

Ultra Light データベースの作成と設定

目次

データベースには、次の 2 つの作成方法があります。

- データベース作成用の Ultra Light 管理ツールを使用してデスクトップで作成する方法。
- Ultra Light の API を使用してデバイス上で作成する方法。デバイス上で作成する方法については、主に各 API に固有の Ultra Light プログラミング・マニュアルで説明しています。

データベースを作成した後は、そのデータベースに接続して、テーブルやその他のデータベース・オブジェクトを構築できます。

デフォルト値を使用しない場合は、データベースを作成するときに作成パラメータの値を設定してください。作成パラメータの値はデータベースの作成後は変更できません。データベース・プロパティの変更が必要な場合は、作成パラメータの別の値を指定して、データベースを再作成する必要があります。

複数のプラットフォームでのデータベースの共有

オペレーティング・システムが異なるために設定が異なる場合でも、データベースをデバイス間でコピーできる場合があります。複数のプラットフォーム間でのプロパティの互換性が不明な場合は、Sybase Central で **Ultra Light のデータベース作成ウィザード**を使用してデータベースを作成します。このウィザードによって、ファイルの互換性の論理が処理されます。このため、配備デバイスの特定の組み合わせでサポートされないファイルを作成してしまうのを防ぐことができます。

参照

- 「Ultra Light の最適化方法」 127 ページ
- 「Ultra Light で使用するデータベース作成パラメータの選択」 35 ページ
- 「Ultra Light 永続データベース・オプション設定の変更」 233 ページ
- 「Ultra Light データベースの操作」 73 ページ

Ultra Light データベースの作成

Ultra Light のデータベース作成ウィザード　使用可能なデータベース作成パラメータのナビゲーションを使用して作成する場合は、この方法を使用します。このウィザードを使用すると、選択するターゲットのプラットフォームに基づいて設定できる項目が制限されるので、選択が簡単になります。データベースが作成されると、コマンド・ライン構文が表示されます。この構文は、記録して、ulcreate ユーティリティで使用できます。「データベース作成ウィザードを使用したデータベースの作成」 30 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　複数のリモート Ultra Light データベースと集中型の統合データベースがある同期システムを作成している場合は、この方法を使用します。

「Mobile Link 同期モデルからの Ultra Light データベースの作成」 31 ページを参照してください。

コマンド・ライン　次のいずれかのユーティリティを使用します。

- データベース作成パラメータに精通しており、空の新しい Ultra Light データベースを作成する方法より短時間でデータベースを作成する場合は、ulcreate ユーティリティを使用します。このユーティリティは、バッチ処理でデータベースを作成する場合に特に便利です。「コマンド・プロンプトからの Ultra Light データベースの作成」 31 ページを参照してください。
- SQL Anywhere のリファレンス・データベース・スキーマに基づいて、空の新しい Ultra Light データベースを作成する場合は、ulinit ユーティリティを使用します。「SQL Anywhere リファレンス・データベースからの Ultra Light データベースの作成」 31 ページを参照してください。
- 新しい Ultra Light データベースのスキーマかデータまたはその両方のソースとして使用する XML ファイルがある場合は、ulload ユーティリティを使用します。「XML からの Ultra Light データベースの作成」 33 ページを参照してください。

データベース作成ウィザードを使用したデータベースの作成

Sybase Central では、**データベース作成ウィザード**を使用してデータベースを作成できます。

♦ 新しい Ultra Light データベースを作成するには、次の手順に従います (Sybase Central の場合)。

1. **[スタート] - [プログラム] - [SQL Anywhere 11] - [Sybase Central]** を選択します。
2. **[ツール] - [Ultra Light 11] - [データベースの作成]** を選択します。
3. **データベース作成ウィザード**の指示に従います。

コマンド・プロンプトからの Ultra Light データベースの作成

ulcreate ユーティリティを使用して、コマンド・プロンプトからデータベースを作成します。このユーティリティをオプションとともに使用すると、データベースを設定できます。

新しい Ultra Light データベースを作成するには、次の手順に従います (コマンド・プロンプトの場合)。

必要なオプションを指定して ulcreate ユーティリティを実行します。たとえば、大文字と小文字を区別する *test.udb* という UTF-8 のデータベースを作成し、同じ名前のデータベースが存在する場合に上書きする場合は、次のコマンドを実行します。

```
ulcreate -c "DBF=test.udb" -o "case=respect;utf_encoding=1" -y
```

データベース・ファイルを接続文字列で指定する代わりに、他のオプションの後にデータベース・ファイルを指定しても、同じ処理が実行されます。次に例を示します。

```
ulcreate -o "case=respect;utf_encoding=1" -y test.udb
```

Mobile Link 同期モデルからの Ultra Light データベースの作成

開発を簡略化するために、Mobile Link には**同期モデル作成ウィザード**があります。このウィザードによって、Ultra Light データベースおよびサーバ側の同期論理を作成できます。

モデルを作成したら、Sybase Central を Mobile Link モデル・モードで操作し、同期モデルを配備する前にカスタマイズできます。モデルの準備ができたら配備できます。このウィザードによって、同期アプリケーションに必要なスクリプトやテーブルが生成されます。

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

SQL Anywhere リファレンス・データベースからの Ultra Light データベースの作成

リファレンス・データベースは、作成している Ultra Light データベースのテンプレートとして機能する SQL Anywhere データベースです。Ultra Light データベースは、リファレンス・データベース内のカラム、テーブル、インデックスのサブセットです。これらのオブジェクトは、リファレンス・データベース内のパブリケーションの一部として選択します。

Sybase PowerDesigner などのアーキテクチャ・ツールを使用して最初にデータをモデリングする場合は、リファレンス・データベースからデータベースを作成すると役立ちます。

リファレンス・データベースからデータベースを作成するには、ulinit ユーティリティを使用します。

♦ 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または抽出するには、次の手順に従います (コマンド・プロンプトの場合)。

1. リファレンス・データベースとして、新しい SQL Anywhere データベースを作成します。

dbinit ユーティリティか Sybase Central を使用して、新しい SQL Anywhere データベースを作成できます。また、サード・パーティのファイルからデータを移行することで、SQL Anywhere 以外のデータベースから SQL Anywhere データベースを作成することもできます。

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

Ultra Light での使用を考慮したデータベースの設定
Ultra Light データベースは、リファレンス・データベースと同じ設定で生成されます。次のオプションをリファレンス・データベースで設定することによって、Ultra Light データベースの動作も制御することになります。

- 日付フォーマット
- 日付順
- 基準年
- Precision
- Scale
- 時間フォーマット
- タイムスタンプ・フォーマット

2. Ultra Light データベースに必要なオブジェクトを追加してリファレンス・データベースを準備します。

   - **テーブルとキー**　テーブルを追加し、Ultra Light に必要なプライマリ・キーを設定します。必要な場合は、Ultra Light アプリケーション内で必要な外部キーの関係を割り当てることもできます。Sybase Central、Sybase PowerDesigner Physical Data Model、または別のデータベース設計ツールを使用できます。「Ultra Light のテーブルとカラムの操作」 74 ページを参照してください。
   - **インデックス**　特に低速のデバイスでは、インデックスによってパフォーマンスが大きく向上します。プライマリ・キーには自動的にインデックス付きになりますが、他のカラムは自動的にインデックス付きにはなりません。「インデックスを使用する場合」 85 ページを参照してください。
   - **パブリケーション**　テーブルによって同期のタイミングを変えるには、パブリケーションを使用します。複数の Ultra Light パブリケーションを使用してテーブルのサブセットを定義して、サブセットごとに同期の優先度を設定できます。「Ultra Light のパブリケーション」 141 ページを参照してください。

パフォーマンスに関するヒント
Ultra Light アプリケーションで情報を特定の順序で取り出すことが頻繁にある場合は、リファレンス・データベースにインデックスを追加できます。「インデックス・スキャンの使用」 114 ページを参照してください。

3. 必要なオプションを指定して ulinit ユーティリティを実行します。

   たとえば、2 つの異なるパブリケーションに含まれるテーブルを持つ *customer.udb* という Ultra Light データベースを初期化するには、次のコマンドを実行します。Pub1 には、優先的に同期するテーブルのサブセットが含まれ、Pub2 には残りのデータが含まれます。

```
ulinit -a DBF=MySource.db -c DBF=customer.udb -n Pub1 -n Pub2
```


参照

- 「データベース作成ウィザードを使用したデータベースの作成」 30 ページ
- 「コマンド・プロンプトからの Ultra Light データベースの作成」 31 ページ
- 「XML からの Ultra Light データベースの作成」 33 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light で使用するデータベース作成パラメータの選択」 35 ページ
- 「Ultra Light のアップグレード」 『SQL Anywhere 11 - 変更点とアップグレード』

XML からの Ultra Light データベースの作成

Ultra Light データベースを管理する中間のフォーマットとして XML を使用できます。このとき、このフォーマットは、Ultra Light で使用するための条件を満たしている必要があります。XML は次のように使用できます。

- データベースのプロパティやオプションが異なる新しいデータベースにデータをロードする。
- Ultra Light の旧バージョンで作成されたデータベースからスキーマをアップグレードする。
- Ultra Light データベースのテキスト・バージョンを作成し、バージョン管理システムにチェック・インする。

Ultra Light では任意の XML ファイルは使用できません。*install-dir¥UltraLite* ディレクトリには *usm.xsd* ファイルがあります。このファイルを使用して、XML フォーマットを確認してください。

♦ **XML ファイルから Ultra Light データベースを作成するには、次の手順に従います。**

1. XML ファイルを任意のディレクトリに保存します。次のいずれかの操作を行うことができます。
   - データベースを XML ファイルにエクスポートまたはアンロードします。SQL Anywhere データベースをアンロードする場合は、サポートされている任意のエクスポート方法を使用します。「リレーショナル・データを XML としてエクスポートする」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。
   - 別のソースから XML 出力を取得します。ソースには、別のリレーショナル・データベース、またはフラット・ファイルにトランザクションが記録される Web サイトを使用できます。常に、XML のフォーマットが Ultra Light の条件を満たしている必要があります。
2. 必要なオプションを指定して ulload ユーティリティを実行します。

   たとえば、*sample.xml* 内のテーブル・フォーマットとデータから、新しい Ultra Light データベースを *sample.udb* ファイルに作成するには、次のように入力します。

   ```
   ulload -c DBF=sample.udb sample.xml
   ```

参照

- 「データベース作成ウィザードを使用したデータベースの作成」 30 ページ
- 「コマンド・プロンプトからの Ultra Light データベースの作成」 31 ページ
- 「SQL Anywhere リファレンス・データベースからの Ultra Light データベースの作成」 31 ページ
- 「Ultra Light のアップグレード」 『SQL Anywhere 11 - 変更点とアップグレード』
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- 「Ultra Light で使用するデータベース作成パラメータの選択」 35 ページ

初回接続時での Ultra Light データベースの作成

接続時に Ultra Light データベースを検出できなかった場合に新しいデータベースを作成するようにアプリケーションをプログラミングできます。プログラミングすると、アプリケーションは、SQL を使用してテーブル、インデックス、外部キーなどを作成できるようになります。データベースにデータを移植するには、統合データベースと同期します。

考慮事項

追加データベースの作成と SQL コードを追加することで、アプリケーションのサイズが大幅に増大する可能性があります。ただし、アプリケーションをデバイスに配備するだけなので、配備作業は簡単になります。また、運用前の開発サイクルで、テストのためにデバイス上のデータベースを削除し、再構築する場合にもこの方法を使用できます。

参照

- 「Ultra Light データベースの作成」 30 ページ
- Ultra Light for C/C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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 で使用するデータベース作成パラメータの選択

Ultra Light データベース作成パラメータは、データベースのスキーマを構成するスキーマ・テーブルに、*name=value* の組み合わせで記録されます。これらの作成パラメータはシステム・テーブルに格納されるので、ユーザとアプリケーションが他のデータと同じようにアクセスできます。「sysuldata システム・テーブル」 314 ページを参照してください。

作成パラメータの値へのアクセス

データベース作成後に作成パラメータの値を変更することはできません。ただし、Sybase Central で対応するデータベース・プロパティを表示することはできます。「Ultra Light データベース・プロパティへのアクセス」 226 ページを参照してください。

データベース・プロパティは、API に適切な GetDatabaseProperty 関数を呼び出すことによって、Ultra Light アプリケーションからプログラム的にアクセスすることもできます。

API 固有の詳細情報については、次の項を参照してください。

- Ultra Light for C/C++ :「GetDatabaseProperty 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET :「GetDatabaseProperty メソッド」 『Ultra Light - .NET プログラミング』
- M-Business :「getDatabaseProperty メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

これらのデータベース作成パラメータに加えて、データベース・オプションや接続パラメータを使用して、データベースの他の設定を行うことができます。次の項を参照してください。

- 「Ultra Light データベース・オプション」 227 ページ
- 「Ultra Light データベースへの接続」 45 ページ
- 「Ultra Light 接続パラメータ」 235 ページ

プロパティ	説明
case	Ultra Light データベースの文字列を比較するときに大文字と小文字を区別するかどうかを設定します。「Ultra Light case 作成パラメータ」 189 ページを参照してください。
checksum_level	データベースのチェックサム検証のレベルを設定します。「Ultra Light checksum_level 作成パラメータ」 191 ページを参照してください。
collation	Ultra Light データベースで使用される照合順を設定します。このプロパティを UTF-8 プロパティと組み合わせて設定するかどうかで、データベースの文字セットが決まります。「Ultra Light 文字セット」 38 ページ、「Ultra Light collation 作成パラメータ」 193 ページ、および「Ultra Light utf8_encoding 作成パラメータ」 219 ページを参照してください。

プロパティ	説明
date_format	日付がデータベースから取り出されるときのデフォルトの文字列フォーマットを設定します。「Ultra Light date_format 作成パラメータ」 194 ページを参照してください。
date_order	年、月、日の日付の順序を解釈する方法を制御します。「Ultra Light date_order 作成パラメータ」 197 ページを参照してください。
fips	Certicom 認定暗号化アルゴリズムを使用して AES FIPS 準拠データの暗号化を制御します。FIPS エンコードは、強力な暗号化の一種です。「Ultra Light データベースの保護」 42 ページと「Ultra Light fips 作成パラメータ」 199 ページを参照してください。
max_hash_size	デフォルトのインデックス・ハッシュ・サイズをバイト単位で設定します。「Ultra Light max_hash_size 作成パラメータ」 201 ページを参照してください。
nearest_century	文字列から日付への変換で、2 桁の年の解釈を制御します。「Ultra Light nearest_century 作成パラメータ」 203 ページを参照してください。
obfuscate	データベースのデータを難読化するかどうか制御します。難読化は単純暗号化です。「Ultra Light データベースの保護」 42 ページと「Ultra Light obfuscate 作成パラメータ」 205 ページを参照してください。
page_size	データベース・ページ・サイズを定義します。「Ultra Light page_size 作成パラメータ」 206 ページを参照してください。
precision	計算結果が小数の場合の最大桁数を指定します。「Ultra Light precision 作成パラメータ」 208 ページを参照してください。
scale	計算結果が最大 precision にトランケートされる場合の、小数点以下の最小桁数を指定します。「Ultra Light scale 作成パラメータ」 210 ページを参照してください。
time_format	データベースから取り出した時刻のフォーマットを設定します。「Ultra Light time_format 作成パラメータ」 212 ページを参照してください。
timestamp_format	データベースから取り出したタイムスタンプのフォーマットを設定します。「Ultra Light timestamp_format 作成パラメータ」 214 ページを参照してください。
timestamp_increment	Ultra Light でのタイムスタンプのトランケート方法を指定します。「Ultra Light timestamp_increment 作成パラメータ」 217 ページを参照してください。

プロパティ	説明
utf8_encoding	Unicode 用の 8 ビット・マルチバイト・エンコードである UTF-8 フォーマットでデータをエンコードします。「Ultra Light 文字セット」 38 ページ と「Ultra Light utf8_encoding 作成パラメータ」 219 ページを参照してください。

Ultra Light 文字セット

文字列の比較結果とソート順は、データベースの文字セット、照合、エンコードのプロパティによって異なります。

データベースに適切な文字セット、照合、エンコードのプロパティを決定するには、主に次の点を考慮します。

- 必要なソート順。一般に、データベースに格納する文字を適切にソートできる照合を選択してください。
- デバイスのプラットフォーム。サポートされているデバイスによって要件が異なり、一部のデバイスでは、UTF-8 で文字をエンコードする必要があります。複数のデバイスをサポートする必要がある場合は、データベースを共有できるかどうかを確認する必要があります。
- データを同期する場合は、統合データベースでサポートされている言語と文字セット。Ultra Light データベースと統合データベースで使用する文字セットに互換性があることを確認します。互換性がなく、一方のデータベースの文字セットにある文字が他方の文字セットになかった場合、データが失われたり、予期せず変更される可能性があります。Ultra Light を多言語環境に配備する場合も、Ultra Light データベースを UTF-8 でエンコードしてください。

  同期のときに、Mobile Link サーバで次のように文字が変換されます。

  1. Ultra Light データベースの文字が Unicode に変換されます。
  2. Unicode 文字が統合データベースの文字セットに変換されます。

参照

- 「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」 39 ページ
- 「Ultra Light collation 作成パラメータ」 193 ページ
- 「Ultra Light utf8_encoding 作成パラメータ」 219 ページ
- 「文字セット、エンコード、照合の概要」 『SQL Anywhere サーバ - データベース管理』
- 「Ultra Light 接続パラメータ」 235 ページ
- 「文字セットの考慮事項」 『Mobile Link - サーバ管理』
- 「Ultra Light case 作成パラメータ」 189 ページ
- 「Ultra Light データベースの保護」 42 ページ

文字セットと照合の変更

Ultra Light によってファイルが書き込まれた後でデータベースの文字セット、照合、およびエンコードのプロパティを変更することはできません。変更するには、データベースを再作成してください。

♦ **新しい照合を使用して Ultra Light データベースを再作成するには、次の手順に従います。**

1. データベースをアンロードします。

Sybase Central の**データベース・アンロード・ウィザード**または ulunload ユーティリティを使用します。「Ultra Light データベースのアンロード・ユーティリティ (ulunload)」 298 ページを参照してください。

2. 適切な照合を使用して新しいデータベースを作成します。
3. 省略可。必要に応じて、新しい照合に合わせてデータを変換します。
4. データベースを再ロードします。

   Sybase Central の**データベース・ロード・ウィザード**または ulload ユーティリティを使用します。「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページを参照してください。

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

各プラットフォームには特定の文字セットとエンコードの要件があります。

Palm OS

UTF-8 エンコードは使用しないでください。Palm では Unicode 文字はサポートされません。常に対象デバイスのコード・ページと一致する照合を選択してください。

Windows デスクトップと Windows Mobile

UTF-8 でエンコードされたデータベースを Windows で使用するときは、ワイド文字をデータベースに渡してください。これらのプラットフォームで UTF-8 エンコードを使用している場合、Ultra Light ではワイド文字以外の文字列パラメータが UTF-8 でエンコードされていると見なされますが、これは Windows で標準の文字セットではありません。接続文字列は例外であり、文字列パラメータはアクティブなコード・ページであると見なされます。ただし、ワイド文字を使用することで、このような複雑な状況を避けることができます。

参照

- 「Ultra Light utf8_encoding 作成パラメータ」 219 ページ
- 「文字セット、エンコード、照合の概要」 『SQL Anywhere サーバ - データベース管理』
- 「Ultra Light 接続パラメータ」 235 ページ
- 「文字セットの考慮事項」 『Mobile Link - サーバ管理』
- 「Ultra Light データベースの保護」 42 ページ

Ultra Light でサポートされている照合

次の表に、Ultra Light でサポートされている CHAR 照合を示します。次のコマンドを入力してこのリストを生成することもできます。

```
ulcreate -l
```

照合ラベル	説明
1250LATIN2	Code Page 1250、Windows Latin 2、中央／東ヨーロッパ言語
1250POL	Code Page 1250、Windows Latin 2、ポーランド語
1251CYR	Code Page 1251、Windows キリル語
1252LATIN1	Code Page 1252、Windows Latin 1、西ヨーロッパ言語
1252NOR	Code Page 1252、Windows Latin 1、ノルウェー語
1252SPA	Code Page 1252、Windows Latin 1、スペイン語
1252SWEFIN	Code Page 1252、Windows Latin 1、スウェーデン語／フィンランド語
1253ELL	Code Page 1253、Windows ギリシア語、ISO8859-7 拡張付き
1254TRK	Code Page 1254、Windows トルコ語、ISO8859-9 拡張付き
1254TRKALT	Code Page 1254、Windows トルコ語、拡張付き ISO8859-9、I-dot と I-no-dot は同じ
1255HEB	Code Page 1255、Windows ヘブライ語、ISO8859-8 拡張付き
1256ARA	Code Page 1256、Windows アラビア語、ISO8859-6 拡張付き
1257LIT	Code Page 1257、Windows バルト諸国語、リトアニア語
874THAIBIN	Code Page 874、Windows Thai、ISO8859-11、バイナリ順序
932JPN	Code Page 932、日本語シフト JIS、Microsoft 拡張文字付き
936ZHO	Code Page 936、中国語 (簡体文字)、PRC GBK
949KOR	Code Page 949、韓国語 KS C 5601-1987 コード、完成型
950ZHO_HK	Code Page 950、中国語 (繁体文字)、Big 5 コード (HKSCS を含む)
950ZHO_TW	Code Page 950、中国語 (繁体文字)、Big 5 コード
EUC_CHINA	中国語 (簡体文字)、GB 2312-80 コード
EUC_JAPAN	日本語の EUC JIS X 0208-1990 と JIS X 0212-1990 コード
EUC_KOREA	Code Page 1361、韓国語 KS C 5601-1992 8 ビット・コード、Johab (組合型)
EUC_TAIWAN	Code Page 964、EUC-TW コード

照合ラベル	説明
ISO1LATIN1	ISO8859-1、ISO Latin 1、西ヨーロッパ言語、Latin 1 順序
ISO9LATIN1	ISO8859-15、ISO Latin 9、西ヨーロッパ言語、Latin 1 順序
ISO_1	ISO8859-1、ISO Latin 1、西ヨーロッパ言語
ISO_BINENG	バイナリ順序、英語 ISO/ASCII 7 ビット文字ケース・マッピング
UTF8BIN	UTF-8、Unicode 用 8 ビット・マルチバイト・エンコード、バイナリ順序

Ultra Light データベースの保護

デフォルトでは、Ultra Light データベースはディスク上で暗号化されません。テキスト・カラムやバイナリ・カラムは、16 進エディタなどの表示ツールを使用して読むことができます。セキュリティ強化のためにデータを暗号化する場合は、次のオプションを検討してください。

- **難読化**　単純暗号化とも呼ばれるこのオプションは、データベース内のデータに対する何気ないアクセスからデータを保護します。このオプションには、強力な暗号化ほどのセキュリティはありません。難読化は、パフォーマンスにほとんど影響しません。難読化は obfuscate 作成パラメータで設定します。エンド・ユーザが対応する接続パラメータを指定する必要はありません。デバイスで使用するのが単純な難読化の場合、特別な設定は不要です。「Ultra Light obfuscate 作成パラメータ」 205 ページを参照してください。
- **AES 128 ビットの強力な暗号化**　Ultra Light データベースは、AES 128 ビット・アルゴリズムを使用して強力に暗号化できます。このアルゴリズムは、SQL Anywhere データベースを暗号化するために使用しているアルゴリズムと同じです。強力な暗号化を使用すると、巧妙で容赦ないデータへのアクセス試行に対抗するセキュリティが提供されますが、パフォーマンスに大きな影響を与えます。暗号化を設定するには、ウィザードで **[Encrypt Database]** オプションを選択してから **[AES Strong Encryption]** を選択します。作成ユーティリティを使用して、key 接続パラメータでキーを設定します。データベースの作成後にエンド・ユーザがデータベースに接続するときには同じパラメータを使用します。デバイスで使用するのが AES 暗号化の場合、特別な設定は不要です。「Ultra Light fips 作成パラメータ」 199 ページを参照してください。
- **AES FIPS 140-2 準拠の暗号化**　Ultra Light には、米国政府とカナダ政府の FIPS 140-2 規格に準拠した暗号化ライブラリが用意されています (Certicom 認定の暗号化モジュールを使用)。FIPS 準拠の暗号化は fips 作成パラメータで設定します。ユーザは、接続文字列で必要なキーを指定する必要があります。AES FIPS 暗号化を使用するには、デバイスを適切に設定する必要があります。「AES_FIP データベース暗号化を使用した Ultra Light の配備」 61 ページと「Ultra Light fips 作成パラメータ」 199 ページを参照してください。

ヒント
Mobile Link サーバの同期ストリームでは、パブリック・キーまたはプライベート・キーを使用してストリーム・データを暗号化できます。配備を容易にするには、これらの証明書を Ultra Light データベースの作成時に埋め込むことができます。「トランスポート・レイヤ・セキュリティを使用する Mobile Link 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。

注意
FIPS と AES の両方のデータベース暗号化で 128 ビット AES が使用されます。したがって、同じ暗号化キーを使用すると、選択する規格に関係なく、データベースは同じ方法で暗号化されます。

警告
暗号化キーはデータベースの作成後に変更できますが、変更する場合は細心の注意が必要です。次の項を参照してください。

- Ultra Light for C++：「ChangeEncryptionKey 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「ChangeEncryptionKey メソッド」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「ChangeEncryptionKey メソッド」『Ultra Light - M-Business Anywhere プログラミング』

この操作は負荷が高く、元に戻せません。操作が途中で停止すると、データベースが完全に失われます。

強力な暗号化が適用されたデータベースの場合、キーのコピーは必ず安全な場所に保管してください。暗号化キーがわからなくなった場合は、テクニカル・サポートに依頼してもデータにはアクセスできません。アクセスできなくなったデータベースは、廃棄して、新しくデータベースを作成する必要があります。

参照

- 「Ultra Light fips 作成パラメータ」 199 ページ
- 「Ultra Light obfuscate 作成パラメータ」 205 ページ
- 「Ultra Light DBKEY 接続パラメータ」 244 ページ
- 「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページ
- Ultra Light.NET：「暗号化と難読化」『Ultra Light - .NET プログラミング』
- Ultra Light for C++：「データの暗号化」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「データベースの暗号化と難読化」『Ultra Light - M-Business Anywhere プログラミング』

Ultra Light データベースへの接続

目次

データベースを使用するアプリケーションでは、データベースへの接続を確立してから、トランザクションが行われます。アプリケーションには、Ultra Light のコマンド・ライン・ユーティリティ、Sybase Central ツールまたは Interactive SQL の接続ウィンドウ、またはカスタム・アプリケーションがあります。

Ultra Light データベースに接続することで、アプリケーションのすべてのアクティビティが行われるチャネルが作成されます。接続が試行されるたびに、データベースに固有の SQL トランザクションが作成されます。

Ultra Light データベース接続パラメータ

設定する接続パラメータの一部は、データベースの作成時に定義するプロパティと重複します。一般に、作成時には必要なプロパティを定義します。接続時には、作成時に設定した値を指定します。これらの共有設定の違いと共通点を理解し、いつ、何を指定すればいいかがわかっていることが重要です。

パラメータ名	説明
CACHE_SIZE	データベース・キャッシュのサイズを定義します。「Ultra Light CACHE_SIZE 接続パラメータ」 236 ページを参照してください。
CON	現在の接続の名前を指定します。「Ultra Light CON 接続パラメータ」 241 ページを参照してください。
DBF と、CE_FILE、PALM_FILE、または NT_FILE	データベースの作成時には、これらのパラメータはデータベースのロケーションを設定します。その後の接続では、ファイルの場所を指定します。 単一プラットフォームのアプリケーションを作成している場合や、Ultra Light 管理ツールに接続している場合は、DBF を使用できます。さまざまなプラットフォーム固有データベースに接続する Ultra Light クライアントをプログラミングしている場合は、その他のプラットフォーム固有バージョンを使用します。次の項を参照してください。 ● 「Ultra Light DBF 接続パラメータ」 242 ページ ● 「Ultra Light CE_FILE 接続パラメータ」 237 ページ ● 「Ultra Light PALM_FILE 接続パラメータ」 252 ページ ● 「Ultra Light NT_FILE 接続パラメータ」 248 ページ
DBN	ファイル名ではなく名前で実行中のデータベースを指定します。「Ultra Light DBN 接続パラメータ」 245 ページを参照してください。
DBKEY	データベースの作成時には、このパラメータは使用する暗号化キーを設定します。その後の接続では、データベースの作成時に設定した暗号化キーを指定し、渡します。指定したキーが間違っていた場合は、接続に失敗します。「Ultra Light DBKEY 接続パラメータ」 244 ページを参照してください。
MIRROR_FILE	データベース・ミラー・ファイルの名前を指定します。「Ultra Light MIRROR_FILE 接続パラメータ」 246 ページを参照してください。

パラメータ名	説明
PALM_ALLOW_BACKUP	Palm デバイス上の HotSync のバックアップ動作を制御します。「Ultra Light PALM_ALLOW_BACKUP 接続パラメータ」 251 ページを参照してください。
PWD	データベースの作成時には、ユーザの初期パスワードを設定します。その後の接続では、ユーザ ID のパスワードを指定します。「Ultra Light PWD 接続パラメータ」 254 ページを参照してください。
RESERVE_SIZE	実際にデータを挿入することなく、Ultra Light データベースが必要とするファイル・システム領域を事前に割り付けます。「Ultra Light RESERVE_SIZE 接続パラメータ」 256 ページを参照してください。
START	Ultra Light エンジンの実行プログラムのロケーションを指定し、起動します。「Ultra Light START 接続パラメータ」 258 ページを参照してください。
UID	データベースの作成時には、初期のユーザ ID を設定します。その後の接続では、データベースに対してユーザを識別します。ユーザ ID は、Ultra Light データベースに格納されている最大 4 つのユーザ ID のうちの 1 つである必要があります。「Ultra Light UID 接続パラメータ」 259 ページを参照してください。

参照

- 「ユーザ ID とパスワードの組み合わせの解釈」 53 ページ
- 「Ultra Light page_size 作成パラメータ」 206 ページ

Ultra Light 接続パラメータの指定

接続情報の収集方法は、入力を体系化または自動化する程度によって異なります。入力を体系化するほど、接続情報の信頼性が高くなります。

接続の詳細は、Ultra Light のカスタム・アプリケーションから接続するか、SQL Anywhere の Ultra Light 用管理ツールから接続するかによって、異なる方法で収集できます。

方法	管理ツール	カスタム・アプリケーション
サポートされている 4 つのデータベース・ユーザのうちの 1 つとしてユーザが認証される必要がある場合、接続時にエンド・ユーザにプロンプトを表示します。Ultra Light のグラフィカル管理ツールでは接続オブジェクトが使用されます。 可能な場合は、ULConnectionParms オブジェクトまたは ConnectionParms オブジェクトを使用します。Open メソッドの引数である接続文字列を使用するよりも確認が簡単で、インタフェースがより体系化されます。次の項を参照してください。 ● Ultra Light.NET：「ユーザの認証」 『Ultra Light - .NET プログラミング』 ● Ultra Light for C/C++：「ユーザの認証」 『Ultra Light - C/C++ プログラミング』 ● Ultra Light for M-Business Anywhere：「ユーザの認証」 『Ultra Light - M-Business Anywhere プログラミング』 ● Ultra Light for Embedded SQL：「ユーザの認証」 『Ultra Light - C/C++ プログラミング』	X	X
ユーザ認証が必要ない場合は、接続文字列を使用します。ユーザを認証しない主な理由としては、配備先が単一ユーザのデバイスである、ユーザがアプリケーションを起動するたびにプロンプトを表示するのは不便であるなどがあります。Ultra Light のコマンド・ライン・ユーティリティでは、データベースへの接続が必要な場合に接続文字列が使用されます。格納されているファイルから値を読み取るように Ultra Light アプリケーションをプログラミングするか、アプリケーションに値をハードコードできます。次の項を参照してください。 ● 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 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 M-Business Anywhere：「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』 ● Ultra Light for Embedded SQL：「データベースへの接続」 『Ultra Light - C/C++ プログラミング』	X[1]	X[3]

方法	管理ツール	カスタム・アプリケーション
ULSQLCONNECT 環境変数を使用して、繰り返し使用する接続パラメータを格納します。パラメータを格納することによって、開発段階でそのパラメータを繰り返し指定する必要がなくなります。ULSQLCONNECT でパラメータとして指定されている値が、Ultra Light のデスクトップ管理ツールのデフォルトになります。 すべての Ultra Light のデスクトップ管理ツールでは、パラメータの優先規則に従って、接続文字列に指定がないパラメータが ULSQLCONNECT の値で確認されます。これらの値を変更にするには、接続文字列で別の値を指定します。「ULSQLCONNECT 環境変数を使用した Ultra Light パラメータの保管」 55 ページを参照してください。	X[2]	なし

[1] 通常はユーザが指定

[2] デスクトップ管理ツールのみ

[3] 通常はハードコードまたはファイルに保管

参照

- 「Ultra Light 管理ツールの接続パラメータの優先度」 52 ページ

Ultra Light 接続パラメータでのファイル・パスの指定

デバイスの物理記憶領域によって、次の項目が決定されます。

- データベースをファイルとして保存するかどうか
- データベースを指定するときの命名規則

単一のプラットフォームに配備する場合や、Ultra Light デスクトップ管理ツールを使用する場合は、DBF パラメータが最適です。次に例を示します。

```
ulload -c DBF=sample.udb sample.xml
```

Windows Mobile のヒント
Ultra Light 管理ツールを使用して、接続しているデバイスにすでに配備されているデータベースを管理できます。「Windows Mobile」 50 ページを参照してください。

プラットフォームを問わないアプリケーションを作成している場合は、プラットフォーム固有のファイル接続パラメータ (CE_FILE、NT_FILE、または PALM_FILE) を使用して汎用の接続文字列を組み立てます。次に例を示します。

```
Connection = DatabaseMgr.OpenConnection("UID=JDoe;PWD=ULdb;
CE_FILE=¥database¥MyCEDB.udb;PALM_FILE=MyPalmDB")
```


Windows デスクトップ

デスクトップでは、絶対パスまたは相対パスを使用できます。

Windows Mobile

Windows Mobile デバイスでは、すべてのパスが絶対パスである必要があります。

Windows Mobile データベースはデスクトップまたは接続しているデバイスで管理できます。Windows Mobile デバイスにあるデータベースを管理するには、絶対パスの前に **wce:¥** を付ける必要があります。たとえば、次のように ulunload ユーティリティを使用します。

```
ulunload -c DBF=wce:¥UltraLite¥myULdb.udb c:¥out¥ce.xml
```

この例では、Ultra Light によってデータベースが Windows Mobile デバイスから、Windows デスクトップの *c:¥out* フォルダにある *ce.xml* ファイルにアンロードされます。

ulunloadold ユーティリティまたは ulunload ユーティリティを使用して、Windows Mobile デバイスにあるデータベースを直接管理する場合、データベースはアンロードまたはアップグレードの前に Ultra Light によってバックアップされません。したがって、この操作を手動で行ってから、これらのユーティリティを実行してください。

Palm OS

Palm OS では、ファイル・パスの概念が使用されていません。したがって、定義方法は、ストアのタイプ (レコードベースまたは VFS) によって異なります。

ファイルベース・ストア (VFS)　VFS ボリュームにあるデータベースについては、次の構文でファイルを定義します。

vfs: [*volume-label***:** | *volume-ordinal***:**] *filename*

volume-label は、**INTERNAL** (組み込みのドライブ) または **CARD** (拡張カードまたはボリュームのラベル名) に設定できます。*volume-label* にデフォルトの文字列はありません。

volume-ordinal を設定してボリュームを指定することもできます。マウントされているボリュームの列挙はさまざまなので、選択する内部または外部のボリュームの正しい順序を設定する必要があります。デフォルト値は 0 です (プラットフォームで列挙されている最初のボリューム)。

filename の場合は、Palm OS のファイルとパスの命名規則に従って、常に絶対ファイル・パスを指定します。パスで指定したディレクトリが見つからない場合は、新しく作成されます。

レコードベースのデータ・ストア　レコードベースのデータ・ストアについては、データベース名は Palm OS のデータベース名のすべての規則に従う必要があります。たとえば、データベース名は 32 文字以内である必要があり、パスを含めることはできません。

また、データベースのロケーションに従って、DBF または PALM_FILE に適切な値を使用する必要があります。

- ulload などを使用して Palm OS のデータベースをデバイス以外の場所に保管する場合は、DBF に *.pdb* 拡張子を使用する。

- ファイルをデバイスに移動すると、*.pdb* 拡張子は HotSync コンジットによって削除される。たとえば、デスクトップに作成したデータベースが *CustDB.pdb* という名前の場合、このデータベースをデバイスに配備するときにファイル名が *CustDB* に変わります。

注意
Palm Install Tool では、VFS ボリュームにデータベースを配備できません。代わりに、カード・リーダなどのツールを使用してメディアに直接データベースをコピーする必要があります。

参照

- 「Ultra Light DBF 接続パラメータ」 242 ページ
- 「Ultra Light NT_FILE 接続パラメータ」 248 ページ
- 「Ultra Light CE_FILE 接続パラメータ」 237 ページ
- 「Ultra Light PALM_FILE 接続パラメータ」 252 ページ

接続文字列を使用した Ultra Light 接続の確立

接続文字列は、接続を定義し、確立できるようにアプリケーションからランタイムに渡す一連のパラメータです。

データベースへの接続の確立は、次の 3 つの手順で行われます。

1. 「パラメータの定義」

   サポートされているパラメータの組み合わせで接続を定義します。一部の接続パラメータは、接続を確立するために必須です。その他のパラメータは、単一の接続のデータベース機能を調整するために使用します。

   パラメータを指定する方法は、Ultra Light 管理ツールから接続するか、Ultra Light アプリケーションから接続するかによって異なります。「Ultra Light 接続パラメータの指定」 47 ページを参照してください。

2. 「文字列のアセンブル」

   開発者かまたはアプリケーションで、指定するパラメータを文字列にアセンブルします。接続文字列は、*keyword=value* をセミコロンで区切ったパラメータのリストです。「パラメータの Ultra Light 接続文字列へのアセンブル」 52 ページを参照してください。

   たとえば、ファイル名、ユーザ ID、パスワードを指定する接続文字列フラグメントは次のようになります。

   ```
   DBF=myULdb.udb;UID=JDoe;PWD=token
   ```

3. 「送信」

   接続文字列をアセンブルしたら、Ultra Light ランタイムへの Ultra Light API を使用して、処理のためにデータベースに渡します。接続の試行が検証されると、接続が許可されます。接続は、次の場合に失敗します。

   - データベース・ファイルが存在しない
   - 認証に失敗した

パラメータの Ultra Light 接続文字列へのアセンブル

管理ツールでも、Ultra Light のカスタム・アプリケーションでも、アプリケーションの接続コードで指定する一連の接続パラメータを「接続文字列」と呼びます。場合によっては、アプリケーションで ConnectionParms オブジェクトのフィールドが解析され、文字列が作成されます。それ以外の場合は、接続文字列は、各パラメータ名と値をセミコロンで区切り、1 行で入力します。

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

Ultra Light ランタイムによって、パラメータが接続文字列にアセンブルされてから、その接続文字列を使用して接続が確立されます。たとえば、ulload ユーティリティを使用した場合、次の接続文字列を使用して新しい XML データが既存のデータベースにロードされます。次の文字列を指定するまで、指定のデータベース・ファイルに接続できません。

```
ulload -c "DBF=sample.udb;UID=DBA;PWD=sql" sample.xml
```

認識できない接続パラメータがあった場合は、Ultra Light でエラーが発生します。

Ultra Light 管理ツールの接続パラメータの優先度

すべての Ultra Light 管理ツールでは、次の順位で接続パラメータが優先されます。

- CE_FILE、NT_FILE、PALM_FILE のいずれかのパラメータが指定されている場合は、常に DBF より優先されます。
- 2 つの DBF パラメータを指定した場合は、最後に指定した方が優先されます。
- 接続文字列で重複するパラメータを指定すると、最後に指定したパラメータが使用されます。他のパラメータはすべて無視されます。
- 接続文字列のパラメータが、ULSQLCONNECT 環境変数または接続オブジェクトで指定されたパラメータより優先されます。
- 接続文字列で指定されていない接続パラメータは、ULSQLCONNECT 環境変数で確認されます。
- UID と PWD の両方の値が接続文字列でも ULSQLCONNECT でも指定されていない場合は、デフォルトの **UID=DBA** と **PWD=sql** が想定されます。

制限事項

接続文字列パラメータ値に含まれる前後のスペースはすべて無視されます。接続パラメータ値に、先頭の一重引用符 (')、先頭の二重引用符 (")、またはセミコロン (;) を含めることはできません。

参照

- 「ULSQLCONNECT 環境変数を使用した Ultra Light パラメータの保管」 55 ページ

Ultra Light ユーザ認証

Ultra Light ユーザ認証は、無効にできません。正常に接続するには、ユーザが認証される必要があります。SQL Anywhere とは異なり、Ultra Light データベースでは、オブジェクトの所有権のためではなく、認証のためだけにユーザを作成し、管理します。ユーザが認証され、データベースに接続すると、ユーザはスキーマ・データを含むデータベース内のすべてのデータに無制限にアクセスできます。

Ultra Light のユーザは既存の接続からのみ追加または変更できます。したがって、Ultra Light のユーザベースを変更するには、その前に有効なユーザ ID とパスワードで接続する必要があります。

初めて接続するときに必要な UID と PWD は、最初にデータベースを作成したときに設定した値です。初期ユーザを設定しなかった場合は、デフォルトの **UID=DBA** と **PWD=sql** を指定します。

認証回避

認証を無効にすることはできませんが、データベースの作成時とデータベースへの接続時に Ultra Light のデフォルトを使用することで認証を回避できます。

UID パラメータと PWD パラメータの両方を指定しなかった場合、Ultra Light ではデフォルトの **UID=DBA** と **PWD=sql** が使用されます。

♦ **Ultra Light で認証を回避するには、次の手順に従います。**

1. データベースの作成時に UID 接続パラメータと PWD 接続パラメータを設定しません。
2. Ultra Light データベース内のデフォルト・ユーザを削除または変更しません。
3. 作成したデータベースへの接続時に UID 接続パラメータと PWD 接続パラメータを設定しません。

参照

- 「制限事項」 93 ページ
- 「Ultra Light のユーザの操作」 93 ページ
- 「ユーザ ID とパスワードの組み合わせの解釈」 53 ページ

ユーザ ID とパスワードの組み合わせの解釈

Ultra Light では、UID パラメータと PWD パラメータのいずれか一方だけ設定するか、どちらも設定しないか、両方とも設定することができます。ただし、部分的な定義が原因でユーザが Ultra Light で識別できない場合があります。次の表は、不完全なユーザ定義が Ultra Light で解釈される方法を示します。

データベース作成時の設定	結果
ユーザ ID とパスワードの**両方**を設定しない	Ultra Light によって UID が **DBA** で PWD が **sql** のデフォルト・ユーザが作成されます。その後の接続時にこれらの接続パラメータを指定する必要はありません。
ユーザ ID パラメータのみ設定 例 ● UID=JaneD ● UID=JaneD;PWD= ● UID=JaneD;PWD=""	Ultra Light によって UID が **JaneD** で PWD が空のデフォルト・ユーザが作成されます。接続時には、常に UID パラメータを指定します。PWD パラメータは不要です。
パスワード・パラメータのみ設定 例 ● PWD=3saBys ● UID=;PWD=3saBys ● UID="";PWD=3saBys	Ultra Light でエラーが発生します。Ultra Light では、ユーザ ID なしでパスワードを設定することはできません。

参照

- 「Ultra Light ユーザ認証」 53 ページ
- 「制限事項」 93 ページ
- 「Ultra Light のユーザの操作」 93 ページ

ULSQLCONNECT 環境変数を使用した Ultra Light パラメータの保管

ULSQLCONNECT 環境変数はオプションです。インストール・プログラムでは設定を行いません。ULSQLCONNECT は、*keyword=value* をセミコロンで区切ったパラメータのリストです。

ULSQLCONNECT を使用すると、開発中に同じ接続パラメータを繰り返し Ultra Light 管理ツールに指定する必要がなくなります。カスタム・アプリケーションには ULSQLCONNECT を使用できません。

警告
等号の代わりにシャープ記号 (#) を使用しないでください。シャープ記号は Ultra Light では無視されます。Ultra Light でサポートされているすべてのプラットフォームで、環境変数の設定内で = を使用できます。

♦ Ultra Light デスクトップ・ツール用に ULSQLCONNECT を設定するには、次の手順に従います。

1. 次のコマンドを実行します。

   set ULSQLCONNECT="*parameter=value*; ..."

2. 管理ツールに追加パラメータが必要な場合や、この環境変数で設定したデフォルト値を変更する必要がある場合は、これらの値を設定してください。ユーザが指定する値が常にこの環境変数より優先されます。

参照

- 「Ultra Light 管理ツールの接続パラメータの優先度」 52 ページ
- 「Ultra Light 接続パラメータの指定」 47 ページ

例

ULSQLCONNECT を使用して *c:¥database¥myfile.udb* というファイルに、ユーザ **demo** とパスワード **test** で接続するには、ULSQLCONNECT 環境変数で次の変数を設定します。

```
set ULSQLCONNECT="DBF=c:¥database¥myfile.udb;UID=demo;PWD=test"
```

この環境変数を設定した場合は、これらのデフォルト値を変更する必要がなければ、-c 接続オプションを使用してこれらの値を指定する必要はありません。

たとえば、ulload を使用して *extra.xml* ファイルからデータベースに情報を追加する場合は、次のようにコマンドを実行します。

```
ulload -a extra.xml
```

Ultra Light のデバイスへの配備

目次

ほとんどの場合、開発は、Ultra Light をモバイル・デバイスとして使用することを最終リリース目標として、Windows デスクトップ上で行われます。ただし、開発環境によっては、さまざまな配備メカニズムを使用して Ultra Light をインストールできます。

Ultra Light アプリケーション・プロジェクトは、同じ Ultra Light データベースを開発データベース、テスト・データベース、配備された運用データベースなど、さまざまな用途に使用することによって、発展させることができます。配備されたデータベース・アプリケーションの存続期間の間、変更や改善は、まず開発データベース行われ、テスト・データベースに伝達された後、最終的に運用データベースに配布されます。

初期インストール

Ultra Light ソリューションのデバイスへの初期インストールは、デバイスの継続的な管理に欠かせない手順です。

継続したメンテナンス

主なメリットおよびコストは、実際に配備され、使用されているアプリケーションに関連しています。このため Ultra Light では、アプリケーションを配信し、データを同期するためのさまざまなメカニズムがサポートされています。アプリケーションの配信およびデータの同期は、Ultra Light 配備で Ultra Light の柔軟性を高める主要な機能となります。

方法	Windows Mobile	Palm OS	Java
ファイル転送	あり	レコードベース、VFS	

方法	Windows Mobile	Palm OS	Java
Ultra Light を Mobile Link 同期クライアントとして使用	あり	レコードベース、VFS	
デバイスでのファイル・システム・ブラウズ・ツールまたはカード・リーダ		VFS のみ (true)[1]	
ActiveSync	あり	なし	
HotSync	なし	レコード・ベースのみ	
ALTER DATABASE SCHEMA	あり	あり	

[1] ターゲットが VFS ボリュームの場合、Palm Install Tool を使用して Ultra Light データベースを配備することはできません。代わりに、カード・リーダなどのツールを使用してメディアに直接データベースをコピーする必要があります。

参照

- 「Mobile Link ファイル転送の使用」 146 ページ
- 「Ultra Light クライアント」 131 ページ
- 「Palm OS の HotSync」 150 ページ
- 「Windows Mobile の ActiveSync」 154 ページ
- 「Ultra Light スキーマのアップグレードの配備」 70 ページ
- 「Ultra Light ALTER DATABASE SCHEMA FROM FILE 文」 470 ページ

Ultra Light エンジンを持つ複数の Ultra Light アプリケーションの配備

Ultra Light エンジンは、32 ビット Windows デスクトップと Windows Mobile 上のみのアプリケーションからの同時 Ultra Light データベース接続を管理するデータ管理モジュールです。このエンジンは、SQL Anywhere インストーラによってデスクトップに自動的にインストールされます。このため、Ultra Light エンジンは、Windows Mobile デバイスに配備するだけで済みます。

♦ **uleng を Windows Mobile デバイスに配備するには、次の手順に従います。**

1. *uleng11.exe* ファイルと適切な **.dll* ファイルをコピーします。コピーする **.dll* ファイルには、データベースの暗号化、同期の暗号化、または圧縮に必要な **.dll* ファイルも含まれます。

ファイル名	基本	ECC TLS	RSA TLS	FIPS RSA TLS	HTTPS	圧縮	FIPS AES データの暗号化
uleng11.exe	X	X	X	X	X	X	X
mlcecc11.dll[1]		X			X		
mlcrsa11.dll[1]			X		X		
mlcrsafips11.dll				X	X		
mlczlib11.dll						X	
sbgse2.dll				X	X		X
ulfips11.dll							X
ulrt11.dll[2]	X	X	X	X	X	X	X

[1] アプリケーションが *ulecc.lib* および *ulrsa.lib* に対してそれぞれ直接リンクしている場合、ファイルは必要ありません。

[2] アプリケーションが *ulimp.lib* に対してリンクしている場合にのみファイルが必要となります。

2. 適切なディレクトリにファイルを保存します。ほとんどの場合、コピー先ディレクトリは次のどれかです。
   - *¥windows* ディレクトリ。クライアントが自動的にこのディレクトリでエンジンを検索するため、このディレクトリを使用することをおすすめします。
   - 他の Ultra Light アプリケーション・ファイルのディレクトリ。
3. *¥windows* ディレクトリ以外のディレクトリを使用する場合は、START 接続パラメータを含めてください。アプリケーションが Ultra Light データベースに接続すると、このパラメータによって、Ultra Light エンジンが起動します。

たとえば、データベースへの接続文字列または Windows Mobile クライアント・アプリケーションの接続コードでは、この START パラメータを使用することがあります。

```
"START=¥Program Files¥MyApp¥uleng11.exe"
```

参照

- 「AES_FIP データベース暗号化を使用した Ultra Light の配備」 61 ページ
- 「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページ

AES_FIP データベース暗号化を使用した Ultra Light の配備

強力なデータベース暗号化方式では、キー (パスワードの一種) がないとデータベースの操作やアクセスを行うことができません。アルゴリズムは、データベースやトランザクション・ログ・ファイルに含まれる情報をエンコードして解読できないようにしています。ただし、データベースの暗号化では、適切な数のファイルをデータベースに配備する必要があります。

-fips オプションを指定して Ultra Light に接続する場合、AES または AES_FIPS の強力な暗号化方式で暗号化されたデータベースを実行できます。AES_FIPS を使用して実行していることを確認するには、**-fips=1** を使用します。

データベースの暗号化に AES FIPS 暗号化を使用した場合、デバイスをプラットフォーム別に設定して配備する必要があります。

♦ アプリケーションとデバイスを AES FIPS 暗号化された Ultra Light データベース用に設定するには、次の手順に従います。

1. Ultra Light データベースをプロパティ **fips=1** を使用して作成します。「Ultra Light fips 作成パラメータ」 199 ページを参照してください。
2. アプリケーションの接続文字列で、接続パラメータとして **DBKEY=*key*** を使用します。「Ultra Light DBKEY 接続パラメータ」 244 ページを参照してください。
3. Palm OS では、ULEnableRsaFipsStrongEncryption を呼び出して、データベースの暗号化を有効にします。「ULEnableFIPSStrongEncryption 関数」 『Ultra Light - C/C++ プログラミング』を参照してください。
4. Palm OS では、*ulrt.lib* の他に、次のファイルへリンクしていることを確認します。
   - *ulfips.lib*
   - *gse1st.lib*
5. 適切なファイルをデバイスに配備していることを確認します。
   - Windows デスクトップと Windows Mobile では、*ulfips11.dll* および *sbgse2.dll* が必要です。Windows Mobile コンポーネントには、コンポーネント DLL ファイルも必要です。
   - Palm OS では *libsbgse_4i.prc* が必要です。

参照

- 「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページ
- Ultra Light.NET :「暗号化と難読化」 『Ultra Light - .NET プログラミング』
- Ultra Light for C++ :「データの暗号化」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「データベースの暗号化と難読化」 『Ultra Light - M-Business Anywhere プログラミング』
- 「Ultra Light fips 作成パラメータ」 199 ページ
- 「Ultra Light データベースの保護」 42 ページ

TLS が有効化された同期を使用した Ultra Light の配備

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 11 - 紹介』を参照してください。

Mobile Link の Ultra Light クライアント・アプリケーションは、TLS が有効になった同期を使用するように設定する必要があります。トランスポート・レイヤ・セキュリティにより、暗号化、改ざん検出、証明書ベースの認証が実現します。「トランスポート・レイヤ・セキュリティの概要」 『SQL Anywhere サーバ - データベース管理』を参照してください。

プラットフォームのサポート
RSA、ECC、FIPS の各暗号化は、すべてのプラットフォームで使用できるわけではありません。各プラットフォームでサポートされている暗号化方法については、http://www.ianywhere.jp/developers/technotes/os_components_1101.html を参照してください。

♦ Ultra Light クライアント・アプリケーションおよびデバイスで TLS 同期を設定するには、次の手順に従います。

1. アプリケーション・コードで次のいずれかを呼び出して、暗号化された同期を有効にします。
   - RSA 暗号化を有効にするには、ULEnableRsaSyncEncryption を呼び出します。「ULEnableRsaSyncEncryption 関数」 『Ultra Light - C/C++ プログラミング』を参照してください。
   - ECC 暗号化を有効にするには、ULEnableEccSyncEncryption を呼び出します。「ULEnableEccSyncEncryption 関数」 『Ultra Light - C/C++ プログラミング』を参照してください。
   - FIPS RSA 暗号化を有効にするには、ULEnableRsaFipsSyncEncryption を呼び出します。この機能が必要なのは、Palm OS クライアントだけです。「ULEnableRsaFipsSyncEncryption 関数」 『Ultra Light - C/C++ プログラミング』を参照してください。
2. 同期の情報ストリームを TLS または HTTPS に設定します。
3. ECC または FIPS 暗号化を有効にしている場合は、以下の設定を行う必要もあります。
   - **ECC** **tls_type** ネットワーク・プロトコル・オプションを **ECC** に設定します。「tls_type」 『Mobile Link - クライアント管理』を参照してください。
   - **FIPS** **fips** ネットワーク・プロトコル・オプションを **Yes** に設定します。「fips」 『Mobile Link - クライアント管理』を参照してください。
4. 適切なライブラリにリンクしていることを確認します。

プラットフォーム	リンク	RSA 暗号化	ECC 暗号化	FIPS 暗号化
Windows デスクトップ	静的 [1]	*ulrsa.lib*	*ulecc.lib*	なし
Windows デスクトップ	動的 [2]	なし	なし	なし
Windows Mobile	静的 [1]	*ulrsa.lib*	*ulecc.lib*	なし
Windows Mobile	動的 [2]	なし	なし	なし
Palm OS	静的 [1]	*ulrsa.lib*	*ulecc.lib*	*ulfips.lib*、*gse1st.lib*

[1] *ulrt.lib* にもリンクする必要があります。

[2] *ulimp.lib* にもリンクする必要があります。

5. 適切なファイルがデバイスにコピーされていることを確認します。

プラットフォーム	リンク	RSA 暗号化	ECC 暗号化	FIPS 暗号化
Windows デスクトップ	静的	なし	なし	*mlcrsafips11.dll* *sbgse2.dll*
Windows デスクトップ	動的 [1]	*mlcrsa11.dll*	*mlcecc11.dll*	*mlcrsafips11.dll* *sbgse2.dll*
Windows Mobile	静的	なし	なし	*mlcrsafips11.dll* *sbgse2.dll*
Windows Mobile	動的 [1]	*mlcrsa11.dll*	*mlcecc11.dll*	*mlcrsafips11.dll* *sbgse2.dll*
Palm OS	静的	なし	なし	*libsbgse_4i.prc*
Windows Mobile コンポーネントおよび Ultra Light エンジン	静的 [2]	*mlcrsa11.dll*	*mlcecc11.dll*	*mlcrsafips11.dll* *sbgse2.dll*

[1] *ulrt11.dll* を配備する必要もあります。

[2] コンポーネント *.dll* ファイルと *uleng11.exe*、またはそのいずれかを配備する必要もあります。

参照

- 「トランスポート・レイヤ・セキュリティを使用する Ultra Light クライアントの設定」『SQL Anywhere サーバ - データベース管理』
- Ultra Light.NET：「暗号化と難読化」『Ultra Light - .NET プログラミング』
- Ultra Light for C++：「データの暗号化」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「データベースの暗号化と難読化」『Ultra Light - M-Business Anywhere プログラミング』

Ultra Light HotSync コンジットの配備

Ultra Light HotSync コンジットは、ユーザがデスクトップからデバイスにアクセスするためのソフトウェア・モジュールです。他のソフトウェア・コンポーネントと同様に、必要なファイルをデバイスに配備して、Ultra Light が Windows Mobile ActiveSync または Palm OS HotSync ソフトウェアとともに動作するようにする必要があります。

Ultra Light は、開発の段階で SQL Anywhere インストーラを使用して、デスクトップにインストールすることになります。ただし、その後に必要な HotSync コンジット・ファイルをエンド・ユーザのコンピュータに配備する必要があります。インストーラでソフトウェアを検出し、実行するには、HotSync マネージャがコンピュータにインストールされている必要があります。

Ultra Light の HotSync コンジット・ファイル

- ***install-dir¥Bin32¥Condmgr¥condmgr.dll*** HotSync のインストール・パスを特定し、HotSync でコンジットを登録するユーティリティの DLL
- ***install-dir¥Bin32¥ulcond11.exe*** デスクトップ・コンピュータへの Ultra Light HotSync コンジットのインストールとアンインストールを行う Ultra Light HotSync コンジット・インストール・ユーティリティ。「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)」 278 ページを参照してください。
- ***install-dir¥Bin32¥dbhsync11.dll*** HotSync によって呼び出されるコンジットの DLL
- ***install-dir¥Bin32¥dblgen11.dll*** 言語リソース・ライブラリ。英語以外の言語では、ファイル名の *en* という文字が、言語を示す 2 文字の省略形に置換され、*dblgde11.dll* (ドイツ語) や *dblgja11.dll* (日本語) になります。
- **ストリーム用の DLL** 省略可。Ultra Light HotSync コンジットと Mobile Link サーバ間の暗号化されたネットワーク通信に必要なストリーム用の DLL。
  - TLS と HTTPS を使用する RSA 暗号化の場合は、*install-dir¥Bin32¥mlcrsa11.dll*。
  - TLS と HTTPS を使用する ECC 暗号化の場合は、*install-dir¥Bin32¥mlcecc11.dll*。
  - TLS と HTTPS を使用する RSA FIPS 暗号化の場合は、*install-dir¥Bin32¥mlcrsafips11.dll*。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 認定の暗号化されたストリーム用 DLL には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

別途ライセンスが必要なコンポーネントの注文については、http://www.iAnywhere.jp/sas/price.html を参照してください。

コンポーネントおよびプラットフォームのサポートについては、http://www.ianywhere.jp/developers/technotes/os_components_1101.html を参照してください。

♦ Ultra Light HotSync コンジットを配備して登録するには、次の手順に従います。

1. エンド・ユーザのデスクトップで、次のディレクトリを作成します。
   - *MyDir¥win32*

- *MyDir¥win32¥condmgr*

2. 次のファイルのコピーを *MyDir¥win32* ディレクトリに配備します。
   - *ulcond11.exe*
   - *dbhsync11.dll*
   - *dblgen11.dll*
3. *Condmgr.dll* ファイルのコピーを *MyDir¥win32¥condmgr* ディレクトリに配備します。
4. 次のレジストリ・キーを見つけます。

   HKEY_CURRENT_USER¥Software¥Sybase¥SQL Anywhere¥11.0¥
5. レジストリ・キーに **Location** という名前の値を作成し、コンジットの配備用ルート・フォルダの名前をデータに指定します。たとえば、*MyDir* のように指定します。
6. エンド・ユーザに通信ストリームを暗号化するための証明書が必要な場合は、ルート証明書をデスクトップ・コンピュータにインストールしてコンジットがアクセスできるようにします。
7. **ulcond11** を実行して、**-c** オプション (場合によっては、**-a** オプション) を使用して各 Ultra Light データベースの接続文字列を設定していることを確認してください。また、正しい作成者 ID が設定されていることも確認してください。

   このユーティリティにより、Ultra Light HotSync コンジットが配備され、正しく設定されます。

   ヒント
   暗号化キーを使用している場合は、接続文字列にはキーを設定しないでください。キーを設定すると、セキュリティ上のリスクが発生する場合があります。その代わりに、コンジットでユーザにキーの入力を要求するプロンプトを表示するようにします。

   たとえば、次のコマンドは、作成者 ID が Syb2 であるアプリケーションの CustDB というコンジットをインストールします。

   ulcond11 -c "DBF=custdb.udb;UID=DBA;PWD=sql" -n CustDB Syb2
8. Ultra Light アプリケーションの ul_synch_info 構造体に同期パラメータを指定しなかった場合は、HotSync または ulcond11 で設定してください。「Mobile Link 同期のプロトコル・オプションの設定」 152 ページを参照してください。

♦ **HotSync コンジットが正しく配備されたことをチェックするには、次の方法に従います。**

- コンピュータのシステム・トレイで、**[HotSync マネージャ]** を右クリックし、 **[カスタム]** を選択します。

  HotSync ユーザ別のコンジット・リストが表示されます。コンジットがリストされていることを確認してから、Mobile Link サーバを起動して、同期操作をテストします。

Ultra Light の ActiveSync プロバイダの配備

Ultra Light ActiveSync プロバイダは、ユーザがデスクトップからデバイスにアクセスするためのソフトウェア・モジュールです。他のソフトウェア・コンポーネントと同様に、必要なファイルをデバイスに配備して、Ultra Light が Windows Mobile ActiveSync または Palm OS HotSync ソフトウェアとともに動作するようにする必要があります。

Ultra Light は、開発の段階で SQL Anywhere インストーラを使用して、デスクトップにインストールすることになります。ただし、Ultra Light をエンド・ユーザに配備する場合は、エンド・ユーザのコンピュータに ActiveSync プロバイダを手動でインストールして登録してください。これにより、ActiveSync は特定のアプリケーションに対してプロバイダの特定のインスタンスを呼び出すタイミングを認識します。

- **mlasinst.exe** ActiveSync プロバイダをインストールして、ActiveSync Manager で登録します。このユーティリティは、同期用に ActiveSync プロバイダで使用するアプリケーションも登録します。
- **mlasdesk.dll** ActiveSync Manager によってデスクトップにロードされる DLL。DLL ファイルのロケーションは、*mlasinst.exe* によって ActiveSync Manager に登録されます。
- **mlasdev.dll** ActiveSync Manager によってデバイスにロードされる DLL。DLL ファイルは、*mlasinst.exe* によってデバイスの正しいロケーションに配備されます。
- **dblgen11.dll** 言語リソース・ライブラリ。

サポートされているプロバイダ・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。

♦ ActiveSync アプリケーションをインストールするには、次の手順に従います。

1. エンド・ユーザのコンピュータで次のことを確認します。
   - ActiveSync Manager がインストールされていること。
   - ActiveSync プロバイダ・ファイルが開発コンピュータからエンド・ユーザのハード・ドライブにコピーされていること。
2. mlasinst を実行して、ActiveSync のプロバイダをインストールします。このユーティリティを使用して、Ultra Light アプリケーションをユーザの Windows Mobile デバイスに登録して配備することもできます。ただし、可能かどうかは、使用するコマンド・ライン構文によります。Ultra Light アプリケーションで複数のファイルを使用する場合は、必要なファイルを手動でコピーしてください。

   次の例では、*mlasdesk.dll* および *mlasdev.dll* が現在のディレクトリにあることを前提にしています。-k オプションと -v オプションが使用されています。-p オプションと -x オプションは、ActiveSync によって起動されるアプリケーションに対するコマンド・ライン・オプションです。

   ```
   mlasinst "C:¥My Files¥myULapp.exe" "¥Program Files¥myULapp.exe"
     "My Application" MYAPP -p -x -v -k
   ```

このユーティリティを使用して、事前にコンパイルされた ARM 5.0 プロセッサの CustDB を配備すると、コマンド・ラインは次のようになります。

```
mlasinst -v "install-dir¥UltraLite¥ce¥arm.50"
"install-dir¥UltraLite¥ce¥arm.50¥custdb.exe" custdb.exe CustDB CUSTDBDEMO
```

注意
ActiveSync を使用して、後で Ultra Light アプリケーションを登録することもできます。「ActiveSync Manager を使用したアプリケーションの登録」 69 ページを参照してください。

3. ActiveSync が新しいプロバイダを認識できるよう、コンピュータを再起動します。
4. Mobile Link プロバイダを有効にします。
   a. [ActiveSync] ウィンドウで **[オプション]** をクリックします。
   b. リストにある **[Mobile Link クライアント]** を有効にして **[OK]** をクリックし、Mobile Link プロバイダをアクティブにします。
   c. 登録されたアプリケーションのリストを表示するには、**[オプション]** をクリックし、**[Mobile Link クライアント]** を選択して **[設定]** をクリックします。

参照

- 「ActiveSync Manager を使用したアプリケーションの登録」 69 ページ
- 「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』

ActiveSync Manager を使用したアプリケーションの登録

ActiveSync で使用するアプリケーションを登録するには、ActiveSync プロバイダのインストール・ユーティリティを使用する方法と、ActiveSync Manager 自体を使用する方法があります。この項では、ActiveSync Manager を使用する方法について説明します。

♦ **ActiveSync Manager で使用するアプリケーションを登録するには、次の手順に従います。**

1. ActiveSync を起動します。
2. [ActiveSync] ウィンドウで **[オプション]** を選択します。
3. 情報タイプのリストから **[Mobile Link クライアント]** を選択し、**[設定]** をクリックします。
4. **[Mobile Link 同期]** ウィンドウで **[新規]** をクリックします。
5. アプリケーションについて次の情報を入力します。
   - **[アプリケーション名]** ActiveSync ユーザ・インタフェースに表示されるアプリケーションを識別する名前。
   - **[クラス名]** アプリケーションの登録済みクラス名。「アプリケーションに対するクラス名の割り当て」『Ultra Light - C/C++ プログラミング』を参照してください。
   - **[パス]** アプリケーションのデバイス上のロケーション。
   - **[引数]** ActiveSync でアプリケーションの起動時に使用するコマンド・ライン引数。
6. **[OK]** をクリックしてアプリケーションを登録します。

参照

- 「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』

Ultra Light スキーマのアップグレードの配備

スキーマをアップグレードするには、SQL 文 ALTER DATABASE SCHEMA FROM FILE を使用します。

アップグレード処理

警告
スキーマのアップグレード中にデバイスをリセットしないでください。スキーマのアップグレード中にデバイスをリセットすると、データが失われ、Ultra Light データベースが「不正」としてマークされます。

1. 新しいデータベース・スキーマと既存のデータベース・スキーマの違いが比較されます。
2. 既存のデータベースのスキーマは変更されます。
3. 新しいスキーマに適合しないローは削除されます。次に例を示します。
   - テーブルに一意性制約を追加し、同じ値を持つローが複数ある場合は、1 つのロー以外すべて削除される。
   - カラム・ドメインを変更して変換エラーが発生した場合は、そのローが削除される。たとえば、VARCHAR カラムを INT カラムに変更し、ローの値が **ABCD** の場合は、このローが削除されます。
   - 新しいスキーマに新しい外部キーがあり、外部ローに一致するプライマリ・ローがない場合は、これらのローが削除される。
4. ローが削除されると、SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE (130) 警告が発生します。

♦ Ultra Light スキーマをアップグレードするには、次の手順に従います。

1. 新しいスキーマを定義する DDL 文の SQL スクリプトを作成します。SQL スクリプト・ファイルの文字セットは、アップグレード対象のデータベースの文字セットに一致している必要があります。

   ulinit または ulunload のいずれかを使用して、スクリプトに必要な DDL 文を抽出する必要があります。これらのユーティリティを次のオプションとともに使用して、DDL 文が構文的に正しいことを確認してください。
   - ulunload を使用している場合は、-n および -s [*file*] オプションを使用する。
   - ulinit を使用している場合は、-l [*file*] オプションを使用する。

   ulunload および ulinit を使用しない場合は、スクリプトで次の事項を確認してください。
   - テーブル、カラム、およびパブリケーションの名前を変更しない。RENAME 操作はサポートされていません。テーブルの名前を変更すると、DROP TABLE 操作または CREATE TABLE 操作として処理されます。

- DDL 文以外の文を含めない。DDL 文以外の文を含めると、想定外の影響が発生する可能性があります。
- SQL 文の語はスペースで区切る。
- 各行に表示されるのは 1 つの SQL 文のみ。
- コメントの先頭には、二重ハイフン (--) を付ける。コメントを指定できるのは、行の先頭のみです。

2. アップグレードを実行するデータベースをバックアップします。
3. 新しい文を実行します。次に例を示します。

```
ALTER DATABASE SCHEMA FROM FILE 'MySchema.sql';
```

エラー通知

アップグレード処理時には Ultra Light エラー・コールバックがアクティブなため、変換処理中に発生したエラーが通知されます。たとえば、SQLE_CONVERSION_ERROR は、パラメータに変換されなかったすべての値をレポートします。エラーが発生しても、処理が失敗したわけではありません。この場合、文が返された後の最終的な SQL コードは、130 警告となります。この警告では、変換処理の操作が示されます。この警告によって、アップグレード処理は停止されません。

参照

- 「Ultra Light ALTER DATABASE SCHEMA FROM FILE 文」 470 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースのアンロード・ユーティリティ (ulunload)」 298 ページ
- 「Ultra Light SQL 文」 467 ページ
- 「Ultra Light のコメント」 321 ページ

Ultra Light データベースの操作

目次

Ultra Light のテーブルとカラムの操作

テーブルは、データを格納するため、またテーブル内のデータの関係を定義するために使用します。テーブルはローとカラムで構成されます。各カラムは電話番号や名前など情報の種類を特定し、各ローはデータのエントリを保持します。

Ultra Light データベースを初めて作成したときには、システム・テーブルだけが表示されます。システム・テーブルには Ultra Light のスキーマが保管されます。システム・テーブルは、必要に応じて Sybase Central で表示／非表示を切り替えることができます。

その後、アプリケーションに必要なテーブルを新しく追加できます。また、これらのテーブル内のデータをブラウズしたり、ソース・データベース内の既存のテーブル間や開いている宛先データベース間でデータをコピー・アンド・ペーストしたりできます。

ローのパックとテーブル定義

Ultra Light では、次の 2 つの形式のローを扱います。

- **アンパックされたロー**　非圧縮形式です。個々のカラム値を読み込んだり書き込んだりするには、その前に、該当するローをアンパックする必要があります。
- **パックされたロー**　アンパックされたローの圧縮形式です。各カラム値が圧縮され、ロー全体に必要なメモリが最小限に抑えられています。パックされたローのサイズは、各カラムに含まれている値だけに依存します。たとえば、同じテーブルに属する 2 つのローで、パックされたサイズが大きく異なる場合があります。LONG BINARY カラムと LONG VARCHAR カラムは、パックされたローとは別に格納されます。

Ultra Light には、パックされたローがデータベース・ページに収まらなければならないという制約があります。LONG BINARY カラムと LONG VARCHAR カラムは、パックされたローには格納されないので、ページ・サイズを超えてもかまいません。

テーブル定義は Ultra Light ランタイムがデータをパックする**前**のローについて記述していることを理解することが重要です。パックされたローのサイズは各カラムの値によって異なるので、パックされたローの要件が満たされているかどうかをテーブル定義を基にして事前に判断することはできません。この理由から、Ultra Light ではアンパックされたローがページに収まらないテーブルを定義できるようになっています。ローがページに収まるかどうかを確認するには、ロー自体を挿入または更新する必要があります。ローが収まらなかった場合、Ultra Light はエラーを検出して通知します。

注意
テーブル・サイズを必要に応じて自由に大きく宣言することはできません。Ultra Light では、宣言されたテーブル・ロー・サイズの上限は 64 KB に固定されています。アンパックされたローのサイズがこの上限を超えることができるテーブルを定義しようとすると、Ultra Light によって SQL エラー・コード SQLE_MAX_ROW_SIZE_EXCEEDED (-1132) が生成されます。

参照

- 「Ultra Light page_size 作成パラメータ」 206 ページ
- 「データベース・テーブル」 『SQL Anywhere 11 - 紹介』
- 「SQL Anywhere でのデータベースの作成」 『SQL Anywhere サーバ - SQL の使用法』
- 「Ultra Light のシステム・テーブル」 305 ページ

Ultra Light のテーブルの作成

Interactive SQL の SQL 文または Sybase Central のいずれかを使用して、リレーショナル・データを保持する新しいテーブルを作成できます。

Ultra Light では、ベース・テーブルだけを作成できます。ベース・テーブルは、永続的なデータを保管するテーブルです。テーブルとそのデータは、それらを明示的に削除しないかぎり存在し続けます。Ultra Light では、グローバル・テンポラリ・テーブルや宣言されたテンポラリ・テーブルはサポートされていません。

注意
Ultra Light アプリケーション内のテーブルには、プライマリ・キーが含まれます。プライマリ・キーは、Ultra Light データベースのローを統合データベースのローに関連付けるため、Mobile Link 同期を行うときにも必要です。

Sybase Central

Sybase Central では、選択したデータベースを操作しながら、これらのタスクを実行できます。

♦ **Ultra Light テーブルを作成するには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. 左ウィンドウ枠で **[テーブル]** を右クリックし、**[新規]** - **[テーブル]** を選択します。
3. **[新しいテーブルの名前を指定してください。]** フィールドに、新しいテーブルの名前を入力します。
4. **[完了]** をクリックします。
5. **[ファイル]** - **[テーブルの保存]** を選択します。

Interactive SQL

Interactive SQL では、新しいテーブルの作成時にカラムを宣言できます。

♦ **Ultra Light テーブルを作成するには、次の手順に従います (Interactive SQL の場合)。**

1. Ultra Light データベースに接続します。
2. CREATE TABLE 文を実行します。

たとえば、次の文は、会社内の従業員のさまざまなスキルや職業適性を記述するテーブルを新しく作成します。テーブルには、各スキルの ID 番号、名前、種別 (たとえば technical または administrative) のカラムが作成されます。

```
CREATE TABLE Skills (
  SkillID INTEGER PRIMARY KEY,
  SkillName CHAR( 20 ) NOT NULL,
  SkillType CHAR( 20 ) NOT NULL
);
```

参照

- 「Ultra Light CREATE TABLE 文」 489 ページ
- 「Ultra Light テーブルへのカラムの追加」 77 ページ

allsync と nosync の各サフィックスの使用

テーブル名に **_allsync** または **_nosync** を追加して、同期のデータ制限を制御できます。これらのサフィックスは、パブリケーションを使用してデータ制限を制御する方法の代わりに使用できます。データの優先度を制御するには、1 つまたは複数のパブリケーションを定義します。

- 名前の末尾が **_allsync** のテーブルを作成すると、最後の同期以降に変更されていない場合でも、同期時にこのテーブル内のすべてのローが同期されます。

  ヒント
  allsync テーブルには、ユーザ固有またはクライアント固有のデータを保管できます。同期するときに、統合データベースのテンポラリ・テーブルにこの Ultra Light テーブルのデータをアップロードできます。同期スクリプトでデータを制御できるため、統合データベースにそのデータを保持する必要はありません。

- 名前の末尾が **_nosync** のテーブルを作成すると、このテーブルのローはすべて同期から除外されます。これらのテーブルは、統合データベースのテーブルに必要がない永続的なデータ用として使用できます。

参照

- 「Ultra Light のパブリケーションの操作」 89 ページ
- 「Ultra Light での同期の設計」 139 ページ
- 「Ultra Light の nosync テーブル」 140 ページ
- 「Ultra Light の allsync テーブル」 140 ページ
- 「Ultra Light CustDB サンプル」 99 ページ

例

CustDB.udb サンプル・データベースでは、ULIdentifyEmployee_nosync というテーブルが非同期のテーブルとして宣言されていることがそのテーブル名からわかります。したがって、このテーブルのデータが変更されても、Mobile Link で同期されず、情報は *CustDB.db* 統合データベースに表示されません。

Ultra Light テーブルへのカラムの追加

テーブルが空の場合は、新しいカラムを簡単に追加できます。ただし、テーブルにデータがすでに格納されている場合は、カラムの定義にデフォルト値が含まれるか、カラムに NULL 値が許可される場合にのみ、カラムを追加できます。

Sybase Central を使用するか、SQL 文 (Interactive SQL など) を実行して、このタスクを実行できます。

Sybase Central

Sybase Central では、選択したテーブルを操作しながら、このタスクを実行できます。

♦ 新しいカラムを Ultra Light テーブルに追加するには、次の手順に従います (Sybase Central の場合)。

1. Ultra Light データベースに接続します。
2. 左ウィンドウ枠で、**[テーブル]** をダブルクリックします。
3. テーブルをダブルクリックします。
4. **[カラム]** タブでテーブルの下の空白スペースを右クリックし、**[新規]** - **[カラム]** を選択します。
5. 新しいカラムの属性を設定します。
6. **[ファイル]** - **[テーブルの保存]** を選択します。

Interactive SQL

Interactive SQL では、テーブルの作成時または変更時にのみカラムを宣言できます。

♦ 新しい Ultra Light テーブルにカラムを追加するには、次の手順に従います (Interactive SQL の場合)。

1. Ultra Light データベースに接続します。
2. CREATE TABLE 文または ALTER TABLE 文を実行し、名前とその他の属性を宣言することでカラムを定義します。

   次の例では、図書データベース用にテーブルを作成して、貸し出し図書の情報を保持します。date_borrowed のデフォルト値は、エントリが作成される日に本が貸し出されることを示します。date_returned カラムは、本が返却されるまでは NULL です。

   ```
   CREATE TABLE borrowed_book (
     loaner_name CHAR(100) PRIMARY KEY,
     date_borrowed         DATE NOT NULL DEFAULT CURRENT DATE,
     date_returned         DATE,
     book                  CHAR(20)
     );
   ```

   次の例は、customer テーブルを変更して、最大 50 文字まで格納できる住所のカラムを追加します。

```
ALTER TABLE customer
ADD address CHAR(50);
```

参照

- 「オブジェクト名の選択」 『SQL Anywhere サーバ - SQL の使用法』
- 「Ultra Light のデータ型」 328 ページ
- 「カラムのデータ型の選択」 『SQL Anywhere サーバ - SQL の使用法』
- 「Ultra Light CREATE TABLE 文」 489 ページ
- 「Ultra Light ALTER TABLE 文」 474 ページ

Ultra Light のカラム定義の変更

テーブルでは、さまざまなカラムの属性を変更するか、カラム全体を削除することで、カラム定義の構造を変更できます。変更したカラム定義は、カラムにすでに格納されているデータの条件を満たす必要があります。たとえば、カラムにすでに NULL のエントリがある場合は、カラムを変更して NULL を不許可にすることはできません。

Sybase Central または Interactive SQL のいずれかを使用して、このタスクを実行できます。

Sybase Central

Sybase Central では、選択したテーブルを操作しながら、これらのタスクを実行できます。

♦ **既存の Ultra Light カラムを変更するには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. 左ウィンドウ枠で、**[テーブル]** をダブルクリックします。
3. テーブルをダブルクリックします。
4. **[カラム]** タブをクリックし、カラムの属性を変更します。
5. **[ファイル]** - **[テーブルの保存]** を選択します。

Interactive SQL

Interactive SQL では、ALTER TABLE 文を使用してこれらのタスクを実行できます。

♦ **既存の Ultra Light カラムを変更するには、次の手順に従います (Interactive SQL の場合)。**

1. Ultra Light データベースに接続します。
2. ALTER TABLE 文を実行します。

この後に示す例は、データベースの構造を変更する方法を示しています。いずれの場合も、文はすぐにコミットされます。したがって、変更を行うと、このテーブルを参照する項目が機能しなくなる可能性があります。

次の文は、SkillDescription カラムを最大 254 文字から最大 80 文字に変更します。


```
ALTER TABLE Skills
MODIFY SkillDescription CHAR( 80 );
```

次の文は、Classification カラムを削除します。

```
ALTER TABLE Skills
DROP Classification;
```

次に示すコマンドは、テーブル全体の名前を変更します。

```
ALTER TABLE Skills
RENAME Qualification;
```

参照

- 「オブジェクト名の選択」 『SQL Anywhere サーバ - SQL の使用法』
- 「Ultra Light のデータ型」 328 ページ
- 「カラムのデータ型の選択」 『SQL Anywhere サーバ - SQL の使用法』
- 「Ultra Light ALTER TABLE 文」 474 ページ

Ultra Light テーブルの削除

次の条件を満たすテーブルを削除できます。

- パブリケーションでアーティクルとして使用されていない
- 別のテーブルの外部キーで参照されているカラムがない

このような場合は、パブリケーションを変更するか、外部キーを削除**してから**、テーブルを削除します。

Sybase Central または Interactive SQL のいずれかを使用して、このタスクを実行できます。

Sybase Central

Sybase Central では、選択したテーブルを操作しながら、これらのタスクを実行できます。

♦ **Ultra Light テーブルを削除するには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. 左ウィンドウ枠で、**[テーブル]** をダブルクリックします。
3. テーブルを右クリックして、**[削除]** を選択します。
4. **[はい]** をクリックします。

Interactive SQL

Interactive SQL では、テーブルの削除をテーブルのドロップとも呼びます。テーブルを削除するには、DROP TABLE 文を使用します。

♦ **Ultra Light テーブルを削除するには、次の手順に従います (Interactive SQL の場合)。**

1. Ultra Light データベースに接続します。
2. DROP TABLE 文を実行します。

   たとえば、次に示す DROP TABLE 文は、Skills テーブルからすべてのレコードを削除し、データベースから Skills テーブルの定義を削除します。

   ```
   DROP TABLE Skills;
   ```

   CREATE 文と同様、DROP 文はテーブルを削除する前と後に COMMIT 文を自動的に実行します。したがって、最後に COMMIT または ROLLBACK を実行した後の変更はすべて確定されます。DROP 文では、テーブル上のインデックスもすべて削除されます。

参照

- 「Ultra Light DROP TABLE 文」 498 ページ

Ultra Light テーブル内の情報のブラウズ

Sybase Central または InteractiveSQL を使用して、Ultra Light データベースのテーブルに保持されているデータをブラウズできます。テーブルには、ユーザ・テーブルとシステム・テーブルがあります。現在のデータベースの表示でシステム・テーブルの表示と非表示を切り替えてテーブルをフィルタできます。Ultra Light には所有権の概念がないので、すべてのユーザがすべてのテーブルをブラウズできます。

Sybase Central

Sybase Central では、選択したデータベースを操作しながら、これらのタスクを実行できます。

♦ **Ultra Light テーブルをブラウズするには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. システム・テーブルが非表示になっているときに 1 つまたは複数のテーブルのデータをブラウズするには、**[コンテンツ]** ウィンドウ枠の空白スペースを右クリックし、**[システム・オブジェクトを表示]** を選択します。
3. テーブルのリストを表示するには、**[テーブル]** をダブルクリックします。
4. テーブル・データを表示するには、テーブルを右クリックして、右ウィンドウ枠の **[データ]** タブをクリックします。

♦ **Ultra Light のシステム・テーブルをフィルタするには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. 接続先のデータベースを右クリックし、**[システム・オブジェクトの非表示]** または **[システム・オブジェクトを表示]** を選択します。

Interactive SQL

Interactive SQL では、SELECT 文を使用してこれらのタスクを実行できます。

♦ Ultra Light のユーザ・テーブルをブラウズするには、次の手順に従います (Interactive SQL の場合)。

1. データベースに接続します。
2. SELECT 文を実行し、ブラウズするユーザ・テーブルを指定します。

♦ Ultra Light のシステム・テーブルをブラウズするには、次の手順に従います (Interactive SQL の場合)。

1. データベースに接続します。
2. SELECT 文を実行し、ブラウズするシステム・テーブルを指定します。

   たとえば、Interactive SQL で systable テーブルの内容を **[結果]** ウィンドウ枠の **[結果]** タブに表示するには、次のコマンドを実行します。

   ```
   SELECT * FROM SYSTABLE;
   ```

参照

- 「Ultra Light のシステム・テーブル」 305 ページ

Ultra Light データベースのデータのコピー・アンド・ペースト

Sybase Central では、コピー・アンド・ペーストおよびドラッグ・アンド・ドロップを使用できます。したがって、データベース内または複数のデータベース間でオブジェクトを共有または移動できます。コピー・アンド・ペーストまたはドラッグ・アンド・ドロップで、次の表に示すデータを共有できます。

ターゲット	結果
別の Ultra Light／SQL Anywhere データベース	新しいオブジェクトが作成され、元のオブジェクトのコードが新しいオブジェクトにコピーされます。
同じ Ultra Light データベース	オブジェクトのコピーが作成されるので、新しいオブジェクトの名前を変更します。

注意
Mobile Link で開いているデータベースのデータをコピーし、Ultra Light データベースにペーストできます。ただし、Ultra Light のデータを、Mobile Link で開いているデータベースにペーストすることはできません。

Sybase Central

Ultra Light プラグインでは、次のいずれかのオブジェクトをコピーすると、そのオブジェクトの SQL もクリップボードにコピーされます。この SQL は、Interactive SQL やテキスト・エディタなどの他のアプリケーションに貼り付けることができます。たとえば、Sybase Central でインデックスをコピーし、テキスト・エディタにペーストすると、そのインデックスの CREATE INDEX 文が表示されます。Ultra Light プラグインにある次のオブジェクトをコピーできます。

- アーティクル
- カラム
- 外部キー
- インデックス
- パブリケーション
- テーブル
- 一意性制約

Interactive SQL

Interactive SQL では、結果セットから別のオブジェクトにデータをコピーすることもできます。

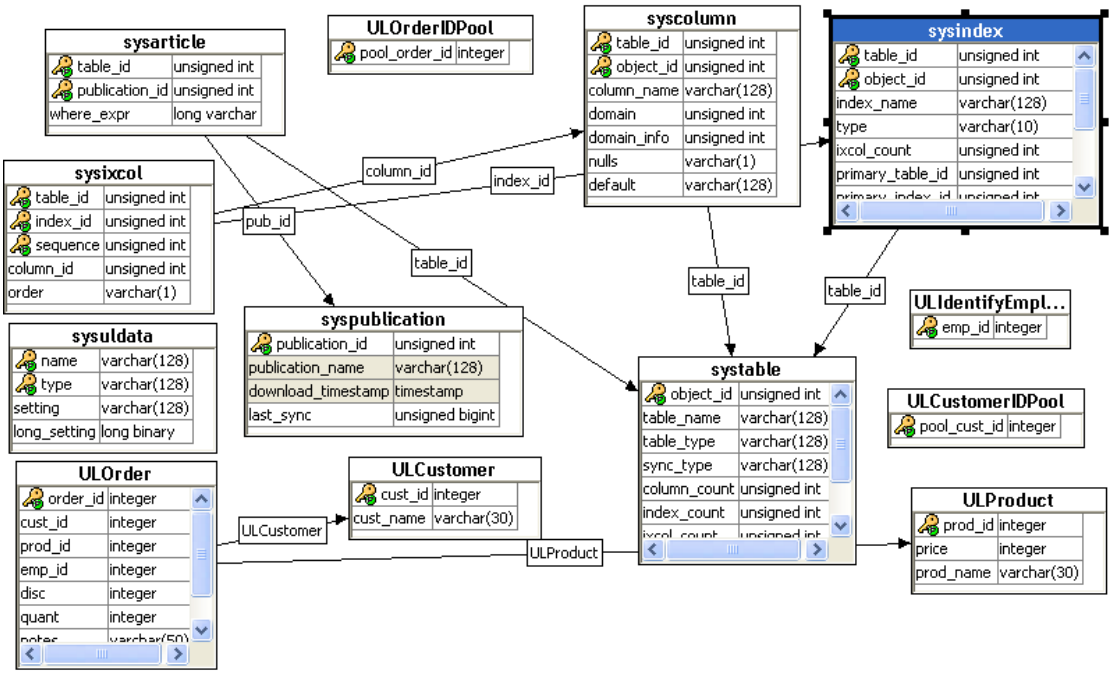
- 結果を指定のオブジェクトにコピーするには、SELECT 文を使用します。
- データベース内の別の場所にある 1 つのローまたは複数のローをテーブルに挿入するには、INSERT 文を使用します。

参照

- 「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』
- 「Ultra Light INSERT 文」 501 ページ
- 「Ultra Light SELECT 文」 507 ページ

Ultra Light プラグインから ER 図を表示する

Ultra Light プラグインからデータベースに接続している場合は、データベースにあるテーブルの ER 図を表示できます。データベースを選択している状態で、右ウィンドウ枠の **[ER 図]** タブをクリックして ER 図を表示します。



ER 図でオブジェクトを並べ替えると、変更は Sybase Central セッション間で保持されます。テーブルをダブルクリックすると、そのテーブルのカラム定義を表示できます。

参照

- 「SQL Anywhere でのデータベースの作成」 『SQL Anywhere サーバ - SQL の使用法』

Ultra Light のインデックスの操作

インデックスは、1 つまたは複数のカラムの値に基づいた、テーブルのローの順序 (昇順または降順) を示します。Ultra Light は、クエリを最適化するときに、既存のインデックスをスキャンして、クエリで指定されているテーブルにインデックスがあるかどうかを確認します。インデックスによってより短時間でローを返すことができる場合は、インデックスが使用されます。アプリケーションで Ultra Light のテーブル API を使用している場合は、ローをスキャンする順序を決定するインデックスを指定できます。

パフォーマンスに関するヒント

インデックスによって、特にテーブルが大きい場合は、クエリのパフォーマンスを向上できます。クエリで特定のインデックスが使用されているかどうかを確認するには、Interactive SQL で実行プランを確認します。

また、プランを返すメソッドがある PreparedStatement オブジェクトを Ultra Light アプリケーションに含めることもできます。

複合インデックスについて

マルチカラムのインデックスは複合インデックスとも呼ばれます。インデックスにカラムを追加すると検索対象を限定できますが、2 カラムのインデックスを使用することと 2 つの別個のインデックスを使用することは異なります。たとえば、次の文では 2 カラムの複合インデックスが作成されます。

```
CREATE INDEX name
ON Employees ( Surname, GivenName );
```

複合インデックスは、最初のカラムだけでは高い選択性が得られない場合に役立ちます。たとえば、Surname と GivenName に対する複合インデックスは、従業員の姓が同じ場合に便利です。各従業員はユニークな ID を持っており、カラム Surname は追加の選択性を提供しないため、EmployeeID と Surname の複合インデックスは役に立ちません。

参照

- 「インデックス・スキャンの使用」 114 ページ
- 「Ultra Light の実行プラン」 360 ページ
- 「複合インデックス」『SQL Anywhere サーバ - SQL の使用法』
- Ultra Light.NET :「テーブル API によるデータ・アクセスと操作」『Ultra Light - .NET プログラミング』
- Ultra Light.NET :「Prepare メソッド」『Ultra Light - .NET プログラミング』
- Ultra Light for C++ :「テーブル API を使用したデータへのアクセス」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++ :「UltraLite_PreparedStatement クラス」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「テーブル API を使用したデータ操作」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「PreparedStatement クラス」『Ultra Light - M-Business Anywhere プログラミング』

インデックスを使用する場合

インデックスは次の場合に使用します。

- **Ultra Light で参照整合性を保つ必要がある場合**　インデックスは Ultra Light に、テーブル内のローに一意性制約を強制するための手段も提供します。よく似ているデータにインデックスを追加する必要はありません。
- **特定のクエリのパフォーマンスがアプリケーションにとって重要である場合**　インデックスによってクエリのパフォーマンスが向上し、そのクエリのパフォーマンスがアプリケーションにとって重要であり、そのクエリを頻繁に使用する場合は、インデックスを使用します。テーブルが非常に小さい場合を除き、インデックスによって検索のパフォーマンスが大幅に向上します。データを頻繁に検索する場合は一般にインデックスを使用することをおすすめします。
- **複雑なクエリがある場合**　複雑なクエリ (たとえば、JOIN 句、GROUP BY 句、ORDER BY 句を使用するもの) は、インデックスを使用するとパフォーマンスが大幅に向上する可能性があります。ただし、パフォーマンスの向上の程度を確認するのは困難である場合があります。したがって、インデックスを使用した場合と使用しなかった場合でクエリをテストし、パフォーマンスを確認することをおすすめします。
- **Ultra Light テーブルのサイズが大きい場合**　ローを検索する平均時間は、テーブルのサイズに比例して長くなります。したがって、非常に大きなテーブルの検索効率を向上させるには、インデックスの使用を検討してください。インデックスを使用すると、インデックスが付いているカラムについては、Ultra Light でローが短時間で見つかります。インデックスがない場合、Ultra Light ではテーブル内のすべてのローを検索し、ローが探索条件を満たすかどうかを確認する必要があるので、大きなテーブルの場合は時間がかかります。
- **Ultra Light クライアント・アプリケーションで大量の挿入、更新、または削除の操作を行わない場合**　Ultra Light ではインデックスがデータ自体とともに保持されるので、これらの操作では、インデックスによってパフォーマンスが下がります。したがって、インデックスの使用は、前述のように頻繁に問い合わせるデータだけに制限してください。Ultra Light のデフォルトのインデックス (プライマリ・キーと一意性制約のインデックス) で十分である可能性があります。
- **WHERE 句か ORDER BY 句またはその両方を使用する場合**　これらの句の対象となるカラムにインデックスを使用すると、これらの句の評価を高速化できます。特にインデックスによってマルチカラムの ORDER BY 句を最適化できます。ただし、インデックスと ORDER BY 句でのカラムの配置がまったく同じである必要があります。

インデックス・タイプの選択

Ultra Light では、インデックスのタイプにユニーク・キー、ユニーク・インデックス、ユニークでないインデックスがあります。これらのタイプでは、インデックスで許可されることが異なります。

インデックスの特性	ユニーク・キー	ユニーク・インデックス	ユニークでないインデックス
インデックス・カラムに同じ値があるローの重複するインデックス・エントリを許可する	いいえ	いいえ	はい
インデックス・カラムに NULL 値を許可する	いいえ	はい	はい

注意
ユニーク・キーには外部キーを作成できますが、ユニーク・インデックスには作成できません。

また、自分でキー・カラムにインデックスを設定する必要はありません。Ultra Light によってユニーク・キーのインデックスが自動的に作成され、管理されます。

参照

- 「Ultra Light インデックスの追加」 86 ページ

Ultra Light インデックスの追加

Sybase Central または Interactive SQL のいずれかを使用して、このタスクを実行できます。

注意
Ultra Light では重複する、または冗長なインデックスは検出されません。インデックスはデータベース内のデータとともに保持されるので、インデックスは慎重に追加してください。

Sybase Central

Sybase Central では、選択したデータベースを操作しながら、このタスクを実行できます。

♦ 指定された Ultra Light テーブルに対する新しいインデックスを設定するには、次の手順に従います (Sybase Central の場合)。

1. Ultra Light データベースに接続します。
2. **[インデックス]** を右クリックし、**[新規]** - **[インデックス]** を選択します。
3. ウィザードの指示に従います。

Interactive SQL

Interactive SQL では、CREATE INDEX 文を使用してこのタスクを実行できます。

♦ 指定された Ultra Light テーブルに対する新しいインデックスを設定するには、次の手順に従います (Interactive SQL の場合)。

1. Ultra Light データベースに接続します。

2. CREATE INDEX 文を実行します。

   設定したデフォルトの最大ハッシュ・サイズでインデックスが作成されます。デフォルト以外の値を使用するインデックスを作成するには、WITH MAX HASH SIZE *value* 句を使用してこのインデックス・インスタンスの新しい値を設定します。「Ultra Light CREATE INDEX 文」 480 ページを参照してください。

   たとえば、従業員情報を追跡するデータベースでの従業員の姓の検索を高速化し、このインデックスに対するクエリのパフォーマンスをチューニングするために、次の文で EmployeeNames というインデックスを作成し、ハッシュ・サイズを 20 バイトに増加します。

   ```
   CREATE INDEX EmployeeNames
   ON Employees (Surname, GivenName)
   WITH MAX HASH SIZE 20;
   ```

参照

- 「Ultra Light CREATE INDEX 文」 480 ページ

インデックスの削除

インデックスを削除すると、データベースから削除されます。

Sybase Central または Interactive SQL のいずれかを使用して、このタスクを実行できます。

Sybase Central

Sybase Central では、選択したデータベースを操作しながら、このタスクを実行できます。

♦ **Ultra Light インデックスを削除するには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. 左ウィンドウ枠で、**[インデックス]** をダブルクリックします。
3. インデックスを右クリックして、**[削除]** を選択します。
4. **[はい]** をクリックします。

Interactive SQL

Interactive SQL では、テーブルの削除をテーブルのドロップとも呼びます。このタスクは DROP INDEX 文で実行できます。

♦ **Ultra Light インデックスを削除するには、次の手順に従います (Interactive SQL の場合)。**

1. データベースに接続します。
2. DROP INDEX 文を実行します。

   たとえば、次の文は、データベースから EmployeeNames インデックスを削除します。

   ```
   DROP INDEX EmployeeNames;
   ```

参照

- 「Ultra Light DROP INDEX 文」 495 ページ

Ultra Light のパブリケーションの操作

パブリケーションとは、同期されるデータを識別するデータベース・オブジェクトです。Ultra Light データベース内のすべてのテーブルと、そのテーブルのすべてのローを同期させる場合は、パブリケーションを作成しないでください。

パブリケーションはアーティクルのセットで構成されます。各アーティクルはテーブル全体、またはテーブル内のローです。WHERE 句を使用してこのローのセットを定義することができます （Palm OS 上の HotSync を除く）。

各データベースには、同期の論理によって、複数のパブリケーションを作成できます。たとえば、優先度の高いデータのパブリケーションを作成することができます。ユーザは高速無線ネットワークを経由してこのデータを同期できます。無線ネットワークには使用料がかかります。使用料を抑えるには、ビジネスに必須のデータのみを同期します。緊急でないデータは、後でクレードルから同期します。

Sybase Central または CREATE PUBLICATION 文を使用して、パブリケーションを作成します。Sybase Central では、[パブリケーション] フォルダにすべてのパブリケーションとアーティクルがあります。

使用上の注意

- Ultra Light パブリケーションでは、カラムのサブセットの定義と、SUBSCRIBE BY 句がサポートされていません。Ultra Light テーブルのカラムが SQL Anywhere 統合データベースのテーブルと正確に一致しない場合は、Mobile Link スクリプトを使用して、これらの違いを解消してください。
- パブリケーションはどのカラムが選択されているかは確認しますが、それらが送信される順序は確認しません。カラムは、CREATE TABLE 文で定義された順に常に送信されます。
- パブリケーションでテーブル同期順序を設定する必要はありません。配備においてテーブル順序が重要な場合は、Ultra Light データベースを同期するときに Table Order 同期パラメータを設定して、テーブル順序を設定できます。
- Ultra Light ではオブジェクトの所有権がサポートされていないので、すべてのユーザがパブリケーションを削除できます。

参照

- 「Ultra Light のテーブルの順序」 142 ページ
- 「データのパブリッシュ」 『Mobile Link - クライアント管理』
- 「Ultra Light での同期の設計」 139 ページ
- 「同期スクリプトの概要」 『Mobile Link - サーバ管理』

Ultra Light でのテーブル全体のパブリッシュ

作成できる最も簡単なパブリケーションは、単一のアーティクルで構成されます。このアーティクルは、テーブルのすべてのローとカラムで構成されます。

Sybase Central または Interactive SQL のいずれかを使用して、このタスクを実行できます。

Sybase Central

Sybase Central では、接続先のデータベースを操作しながら、このタスクを実行できます。

♦ Ultra Light のテーブル全体を 1 つ以上パブリッシュするには、次の手順に従います (Sybase Central の場合)。

1. Ultra Light データベースに接続します。
2. **[パブリケーション]** フォルダを右クリックし、**[新規]** - **[パブリケーション]** を選択します。
3. **[新しいパブリケーションの名前を指定してください。]** フィールドに、新しいパブリケーションの名前を入力します。**[次へ]** をクリックします。
4. **[テーブル]** タブで、**[使用可能なテーブル]** リストからテーブルを選択します。**[追加]** をクリックします。
5. **[完了]** をクリックします。

Interactive SQL

Interactive SQL では、CREATE PUBLICATION 文を使用してこのタスクを実行できます。

♦ Ultra Light のテーブル全体を 1 つ以上パブリッシュするには、次の手順に従います (Interactive SQL の場合)。

1. Ultra Light データベースに接続します。
2. 新しく作成するパブリケーションの名前とパブリッシュするテーブルの名前を指定して、CREATE PUBLICATION 文を実行します。

   たとえば、次の文は、customer テーブル全体をパブリッシュするパブリケーションを作成します。

```
CREATE PUBLICATION pub_customer (
  TABLE customer
);
```

参照

- 「Ultra Light CREATE PUBLICATION 文」 482 ページ
- 「Ultra Light クライアント」 131 ページ

Ultra Light テーブルのローのサブセットのパブリッシュ

パブリケーションには、特定のテーブル・ローだけが含まれます。Sybase Central または Interactive SQL では、WHERE 句によって、変更されたローのうち WHERE 句の探索条件に一致するローのみをアップロードするよう制限されます。

すべての変更されたローをアップロードする場合は、WHERE 句を指定しないでください。

Palm OS
このプラットフォームでは、CREATE PUBLICATION 文で WHERE 句を使用できません。

Sybase Central

Sybase Central では、接続先のデータベースを操作しながら、このタスクを実行できます。

♦ Ultra Light テーブル内の一部のローだけをパブリッシュするには、次の手順に従います (Sybase Central の場合)。

1. Ultra Light データベースに接続します。
2. **[パブリケーション]** フォルダを右クリックし、**[新規]** - **[パブリケーション]** を選択します。
3. **[新しいパブリケーションの名前を指定してください。]** フィールドに、新しいパブリケーションの名前を入力します。
4. **[次へ]** をクリックします。
5. **[使用可能なテーブル]** リストでテーブルを選択して、**[追加]** をクリックします。
6. **[WHERE 句]** タブで、**[アーティクル]** リストからテーブルを選択します。オプションで、[挿入] ウィンドウを使用して探索条件をフォーマットできます。
7. **[完了]** をクリックします。

Interactive SQL

Interactive SQL では、CREATE PUBLICATION 文を使用してこのタスクを実行できます。

♦ Ultra Light で WHERE 句を使用してパブリケーションを作成するには、次の手順に従います (Interactive SQL の場合)。

1. Ultra Light データベースに接続します。
2. パブリケーション対象のテーブルと WHERE 条件を含む CREATE PUBLICATION 文を実行します。

   たとえば、次の例は、単一のアーティクルで構成され、営業担当者 ID 番号 856 のすべての受注情報が含まれるパブリケーションを作成します。

```
CREATE PUBLICATION pub_orders_samuel_singer
 ( TABLE SalesOrders
    WHERE SalesRepresentative = 856 );
```

参照

- 「Ultra Light CREATE PUBLICATION 文」 482 ページ
- 「Ultra Light クライアント」 131 ページ

Ultra Light のパブリケーションの削除

パブリケーションは Sybase Central または Interactive SQL を使用して削除できます。

Sybase Central

Sybase Central では、接続先のデータベースを操作しながら、このタスクを実行できます。

♦ **パブリケーションを削除するには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. 左ウィンドウ枠で、**[パブリケーション]** フォルダをダブルクリックします。
3. パブリケーションを右クリックして、**[削除]** を選択します。
4. **[はい]** をクリックします。

Interactive SQL

Interactive SQL では、パブリケーションの削除をパブリケーションのドロップとも呼びます。このタスクは DROP PUBLICATION 文で実行できます。

♦ **パブリケーションを削除するには、次の手順に従います (Interactive SQL の場合)。**

1. Ultra Light データベースに接続します。
2. DROP PUBLICATION 文を実行します。

   たとえば、次の文は、パブリケーション pub_orders を削除します。

```
DROP PUBLICATION pub_orders;
```

参照

- 「Ultra Light DROP PUBLICATION 文」 496 ページ
- 「Ultra Light クライアント」 131 ページ

Ultra Light のユーザの操作

Ultra Light データベースではユーザ ID とパスワードは暗号化されているので、定義されているユーザのリストは Sybase Central だけで確認できます。

Ultra Light のユーザ ID は、Mobile Link のユーザ名や SQL Anywhere のユーザ ID とは別です。

制限事項

ユニークなユーザ ID を作成するときは、次の制限事項に注意してください。

- Ultra Light では、データベースごとに最大 4 つのユニークなユーザがサポートされます。
- ユーザ ID とパスワードの値はいずれも文字数制限が 31 文字です。
- パスワードは常に大文字と小文字が区別され、ユーザ ID は常に大文字と小文字が区別されません。パスワードは Sybase Central でいつでも変更できます。
- ユーザ ID に含まれる前後のスペースはすべて無視されます。ユーザ ID に、先頭の一重引用符 (')、先頭の二重引用符 (")、またはセミコロン (;) を含めることはできません。
- ユーザ ID は一度作成すると変更できません。変更する必要がある場合は、ユーザ ID を削除し、新しい ID を追加してください。
- パスワードは、Sybase Central を使用して変更できます。

新しい Ultra Light ユーザの追加

Ultra Light では、Interactive SQL でのユーザの作成がサポートされていません。ただし、次の方法でユーザを追加できます。

- Sybase Central を使用して [ユーザ] フォルダにユーザを追加する。
- Connection オブジェクトに GrantConnectTo 関数を使用して Ultra Light アプリケーションから新しいユーザを追加する。

♦ **新しい Ultra Light ユーザを作成するには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. **[ユーザ]** フォルダを右クリックし、**[新規]** - **[ユーザ]** を選択します。
3. ウィザードの指示に従います。さまざまなユーザ ID とパスワードの組み合わせが Ultra Light でどのように解釈されるかを理解している必要があります。「ユーザ ID とパスワードの組み合わせの解釈」 53 ページを参照してください。

参照

- Ultra Light.NET：「ユーザの認証」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「ユーザの認証」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「ユーザの認証」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「ユーザの認証」『Ultra Light - C/C++ プログラミング』

既存の Ultra Light ユーザの削除

Ultra Light では、SQL を使用したユーザの削除がサポートされていません。ただし、次の方法でユーザを削除できます。

- Sybase Central を使用して [ユーザ] フォルダからユーザを削除する。
- Connection オブジェクトに RevokeConnectFrom 関数を使用して Ultra Light アプリケーションからユーザを削除する。

♦ **既存の Ultra Light ユーザを削除するには、次の手順に従います (Sybase Central の場合)。**

1. Ultra Light データベースに接続します。
2. 左ウィンドウ枠で、**[ユーザ]** フォルダをダブルクリックします。
3. ユーザを右クリックし、**[削除]** を選択します。

参照

- Ultra Light.NET：「ユーザの認証」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「ユーザの認証」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「ユーザの認証」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「ユーザの認証」『Ultra Light - C/C++ プログラミング』

イベント通知の操作

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。通知は、イベントが発生したときに送信されるメッセージであり、パラメータに関する追加情報も示されます。Ultra Light にはシステム・イベントがありますが、イベントにはユーザ定義のものもあります。

イベント通知によって、接続間または同じデータベースに接続されているアプリケーション間での調整および通知を行うことができます。通知は、接続のデフォルト・キューまたは明示的に作成および命名されたキューのいずれかで管理されます。イベントの発生時に、通知が登録されたキュー (または接続) に送信されます。

それぞれの接続では、固有の通知キューが管理されます。名前付きのキューは、任意の接続に対して作成できます。

また、この機能では、事前に定義されたシステム・イベントを使用して、テーブルが変更された場合などのデータの変更に対する「トリガ」や同期が行われた場合の通知が提供されます。次のような事前に定義されたイベントがあります。

- Commit
- SyncComplete
- TableModified

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。

イベント

イベント	出現箇所
Commit	コミットの終了時に通知されます。
SyncComplete	同期の終了時に通知されます。
TableModified	テーブルのローが挿入、更新、または削除されたときにトリガされます。イベントに登録された要求によって影響を受けるローの数にかかわらず、要求ごとに 1 つのイベントが通知されます。 *object_name* パラメータは、モニタするテーブルを指定します。値 "*" は、データベース内のすべてのテーブルを意味します。 *table_name* 通知パラメータは、変更されたテーブルの名前です。

```
note_info.event_name = "SyncComplete";
note_info.event_name_len = 12;
note_info.parms_type = ul_ev_note_info::P_NONE;

note_info.event_name = "TableModified";
note_info.event_name_len = 13;
note_info.parms_type = ul_ev_note_info::P_TABLE_NAME;
note_info.parms = table->name->data;
note_info.parms_len = table->name->len;
```


キューの操作

キューは、作成および破棄することができます。

CreateNotificationQueue は、現在の接続のイベント通知キューを作成します。キュー名は、接続ごとにスコープされるため、別々の接続で同じ名前を持つキューを作成できます。イベント通知が送信されると、データベース内で一致する名前を持つすべてのキューが、個別のインスタンスの通知を受け取ります。名前では大文字と小文字が区別されません。デフォルトのキューは、キューが指定されていない場合に要求に応じて、各接続に対して作成されます。接続に対してその名前がすでに存在する場合や有効でない場合は、エラーが発生して呼び出しが失敗します。

DestroyNotificationQueue は、指定されたイベント通知キューを破棄します。キュー内に未読の通知が残っている場合は、警告が通知されます。未読の通知は破棄されます。接続のデフォルトのイベント・キューが作成されている場合、接続が閉じると破棄されます。

イベントの操作

DeclareEvent は、登録して、トリガできるイベントを宣言します。Ultra Light では、データベースまたは環境での操作によってトリガされるシステム・イベントの一部が事前に定義されています。イベント名は、ユニークにする必要があります。また、イベント名では大文字と小文字が区別されません。イベントが正常に宣言された場合は true、名前がすでに使用されているか、または無効な場合は false を返します。

RegisterForEvent は、イベントの通知を受信するキューを登録します。キュー名が指定されていない場合は、デフォルトの接続キューが暗黙で指定され、必要に応じて作成されます。特定のシステム・イベントでは、そのイベントが適用されるオブジェクト名を指定できます。たとえば、TableModified イベントではテーブル名を指定できます。SendNotification とは異なり、登録された特定のキューのみイベントの通知を受信します。別の接続の同じ名前を持つ他のキューは、明示的に登録されている場合を除き、イベントの通知を受信しません。正常に登録された場合は true、キューまたはイベントが存在しない場合は false を返します。

TriggerEvent は、イベントをトリガし、登録されているすべてのキューに通知を送信します。送信済みのイベント通知の数を返します。パラメータは、name=value; のペアとして指定できます。

通知の操作

SendNotification は、指定された名前と一致する、データベース内のすべてのキュー (現在の接続におけるキューを含む) に通知を送信します。この呼び出しはブロックしません。特別なキュー名の "*" を使用すると、すべてのキューに送信します。送信済み通知の数 (一致するキューの数) を返します。パラメータは、name=value; のペアとして指定できます。

GetNotification は、イベント通知を読み込みます。この呼び出しは、通知が受信されるまで、または指定された待機時間が経過するまでブロックします。待機をキャンセルするには、指定したキューに別の通知を送信するか、CancelGetNotification を使用します。通知を読み込んだら、ReadNotificationParameter を使用して追加のパラメータを取得します。イベントが読み込まれた場合は true、待機期間が終了したか、またはキャンセルされた場合は false を返します。

GetNotificationParameter は、GetNotification で読み込まれたイベント通知の名前付きパラメータを取得します。指定されたキューでの最後に読み込まれた通知のパラメータのみ取得できます。パラメータが見つかった場合は true、見つからなかった場合は false を返します。

CancelGetNotification は、指定された名前と一致する名前を持つすべてのキューで保留中の GetNotification 呼び出しをキャンセルします。影響を受けるキューの数を返します (必ずしも、ブロックされた読み込みの数ではありません)。

その他の考慮事項

- 通知キューおよびイベントの名前は 32 文字に制限されています。
- システム・リソースを管理するには、通知の数を制限します。この制限を超えると、SQLE_EVENT_NOTIFICATION_QUEUE_FULL が通知され、保留中の通知が廃棄されます。

SQL パススルーの Ultra Light サポート

SQL パススルー機能によって、統合データベースのパブリッシャは Ultra Light に SQL 文のスクリプトを送信して実行できます。SQL 文のスクリプトは、自動的に実行するか、または DBA 権限を持つユーザが手動で実行できます。

SQL 文のスクリプトは、同期中に Ultra Light に自動的にダウンロードされ、syssql テーブルに格納されます。ファイルベースのダウンロードまたは ping が実行されている場合やダウンロードが再起動された場合、SQL 文のスクリプトはダウンロードされません。

SQL 文のスクリプトは、次の場合を除き、次回データベースが起動したときに自動的に実行されます。

- 実行スクリプトで、**flags** パラメータが **manual** に設定されている。
- 接続パラメータ dont_run_scripts が設定されている。
- アップロードに失敗した。

実行スクリプトで **flags** パラメータが **manual** に設定されている場合は、次の方法を使用して、統合データベースから送信されたスクリプトを手動で適用できます。

- ExecuteSQLPassthroughScripts
- ExecuteNextSQLPassthroughScript
- GetSQLPassthroughScriptCount

同期中は、Ultra Light が、前回の同期以降に実行されたスクリプトのステータスをアップロードします。Ultra Light でスクリプトの実行エラーが発生した場合は、統合データベースに通知され、処理手順が送信されるまでスクリプトの実行は停止されます。実行されたスクリプトのステータスは、統合データベースの「ml_passthrough_status」『Mobile Link - サーバ管理』テーブルに格納されます。このテーブルを確認して、分散されたパススルー・スクリプトが成功したかどうかを判別します。

自動または手動で実行されたスクリプトの進行状況をモニタする場合は、observer コールバック関数を使用します。「ULRegisterSQLPassthroughCallback 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。

参照

- 「SQL パススルーの概要」『Mobile Link - クライアント管理』
- 「ExecuteSQLPassthroughScripts メソッド」『Ultra Light - .NET プログラミング』
- 「ExecuteNextSQLPassthroughScript メソッド」『Ultra Light - .NET プログラミング』
- 「GetSQLPassthroughScriptCount メソッド」『Ultra Light - .NET プログラミング』

Ultra Light CustDB サンプル

目次

CustDB サンプルは、SQL Anywhere とともにインストールされます。CustDB は、Mobile Link による SQL Anywhere 統合データベースとの同期を実行する多層データベース管理ソリューションです。

CustDB は、次のコンポーネントで構成されています。

- SQL Anywhere 統合データベース。このデータベースは販売管理データが事前に挿入されています。
- Ultra Light リモート・データベース。このデータベースは最初は空です。
- Ultra Light クライアント・アプリケーション。
- Mobile Link サーバ同期サンプル。同期スクリプトが事前に作成されています。

サポートされているプログラミング・インタフェースとプラットフォームごとに、複数のバージョンのアプリケーション・コードがあります。ただし、このチュートリアルでは、Windows デスクトップ用アプリケーションのコンパイル済みバージョンだけを使用します。各バージョンで Ultra Light の機能が実装されますが、各プラットフォームの規則に応じて機能が異なる場合があります。

注意
一度に実行できる CustDB のインスタンスは 1 つだけです。複数のインスタンスを実行すると、最初のインスタンスがフォアグランドに移されます。

CustDB を使用すると、販売担当者が取引を追跡、モニタできます。また、次の 2 種類のユーザからの情報が収集されます。

- ユーザ ID 51、52、 53 で認証された販売担当者
- ユーザ ID 50 で認証されたモバイル管理者

これらのユーザによって収集された情報は、統合データベースと同期できます。

各レッスンを終了したら、次の操作を行う方法を理解できます。

- Mobile Link サーバを実行して、統合データベースと Ultra Light リモートの間でデータ同期を行う
- Sybase Central を使用して Ultra Light リモートのデータをブラウズする
- Ultra Light のコマンド・ライン・ユーティリティを使用して Ultra Light データベースを管理する

参照

- 「CustDB サンプルファイルのロケーション」 101 ページ
- 「シナリオ」 『Mobile Link - クイック・スタート』
- 「CustDB サンプル内のユーザ」 『Mobile Link - クイック・スタート』
- 「CustDB データベース内のテーブル」 『Mobile Link - クイック・スタート』

CustDB サンプルファイルのロケーション

SQL Anywhere CustDB データベース

このデータベースは、統合データベースです。インストール中、このデータベース用に SQL Anywhere 11 CustDB という名前の ODBC データ・ソースが作成されます。

CustDB のインストール環境は、既存のサンプルを使用するか、新しいファイルを再作成するかによって異なります。

- 既存のサンプルを使用する場合：*samples-dir¥UltraLite¥CustDB¥custdb.db*
- 統合された *CustDB.db* ファイルに同期された変更内容を消去して、新しいバージョンを使用する場合： *samples-dir¥UltraLite¥CustDB¥newdb.bat*

このファイルのスキーマの詳細については、「Mobile Link CustDB サンプルの解説」『Mobile Link - クイック・スタート』を参照してください。

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。

Ultra Light CustDB データベース

情報のサブセットだけを含む、統合データベースのリモート・バージョンです。含まれる情報はデータベースを同期するユーザによって異なります。

ファイル名とロケーションは、プラットフォーム、プログラミング言語、またはデバイスによっても異なることがあります。

- Ultra Light.NET の場合：*samples-dir¥UltraLite.NET¥CustDB¥Common¥*
- その他すべてのプラットフォームおよび API の場合：*samples-dir¥UltraLite¥CustDB¥custdb.udb*

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。

RDBMS 固有の構築スクリプト

サポートされている任意の RDBMS 用に CustDB 統合データベースを再構築する SQL スクリプトです。

samples-dir¥MobiLink¥CustDB ディレクトリに次のファイルがあります。

- SQL Anywhere の場合：*custdb.sql*
- Adaptive Server Enterprise の場合：*custase.sql*
- Microsoft SQL Server の場合：*custmss.sql*
- Oracle の場合：*custora.sql*
- IBM DB2 の場合：*custdb2.sql*

統合データベースの設定の詳細については、「CustDB 統合データベースの設定」『Mobile Link - クイック・スタート』を参照してください。

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。

Ultra Light CustDB クライアント・アプリケーションと ReadMe ファイル

Ultra Light リモート・データベースへの使いやすいインタフェースを提供するエンド・ユーザ・ツールです。サポートされている API ごとにサンプル・クライアントがインストールされます。

各クライアント・アプリケーションには、*ReadMe.html* ファイルまたは *ReadMe.txt* ファイルが含まれています。ファイルの内容はそれぞれ異なります。ただし、一部のファイルには、サンプルの構築と実行に必要な手順の概略が記載されています。

アプリケーションと ReadMe のロケーションは、環境変数によって異なります。「レッスン 1：CustDB アプリケーションの構築と実行」 103 ページを参照してください。

SQL 同期論理

Ultra Light データベースの情報を問い合わせて修正するための SQL 文と、統合データベースとの同期を開始するために必要な呼び出しです。

samples-dir¥UltraLite¥CustDB¥custdb.sqc

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。

参照

- 「Mobile Link 統合データベース」『Mobile Link - サーバ管理』

レッスン 1：CustDB アプリケーションの構築と実行

CustDB アプリケーションは、数多くの開発環境用に構築されます。すべての環境に適用される一般的な手順については、次の項を参照してください。

♦ **CustDB アプリケーションを構築して実行するには、次の手順に従います。**

1. CustDB アプリケーションを構築します。
   a. 適切な環境で CustDB プロジェクト・ファイルを開きます。
   b. ソース・コードをコンパイルします。
2. CustDB アプリケーションを実行します。
   a. CustDB 実行ファイルをモバイル・デバイスに配備します。
   b. Ultra Light CustDB データベースをモバイル・デバイスに配備します。
   c. CustDB 実行ファイルを実行します。

Windows 32 ビット・デスクトップ用 Ultra Light

CustDB アプリケーションを実行する前に構築する必要はありません。

CustDB 実行ファイルは *install-dir¥UltraLite¥win32¥386* ディレクトリにあります。

Ultra Light for C/C++

- **C/C++ のすべてのバージョン**　C/C++ には数多くの開発環境が存在するので、C/C++ CustDB プロジェクト・ファイルは複数のバージョンが用意されています。ほとんどのバージョンでは、汎用ファイルを使用しています。汎用ファイルは、*samples-dir¥UltraLite¥Custdb* ディレクトリにあります。

  C/C++ CustDB アプリケーションのすべてのバージョンについては、*samples-dir¥UltraLite¥Custdb¥readme.txt* を参照してください。

- **CodeWarrior for Palm OS**　プロジェクト・ファイルは *samples-dir¥UltraLite¥CustDB¥cwcommon* ディレクトリと *samples-dir¥UltraLite¥CustDB¥cw* ディレクトリにあります。

  C/C++ CustDB アプリケーションの構築の詳細については、「CodeWarrior を使用した CustDB サンプル・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。

- **Visual Studio**　プロジェクト・ファイルは、Visual Studio のバージョンに応じて、*samples-dir¥UltraLite¥CustDB¥vs7* ディレクトリまたは *samples-dir¥UltraLite¥CustDB¥vs8* ディレクトリにあります。CustDB アプリケーションを構築して実行するには、レッスン冒頭で説明している手順に従ってください。

Ultra Light for Embedded SQL

Embedded Visual C++ 用のプロジェクト・ファイルは、Embedded Visual C++ のバージョンに応じて、*samples-dir¥UltraLite¥CustDB¥EVC* ディレクトリまたは *samples-dir¥UltraLite¥CustDB¥EVC40* ディレクトリにあります。

Windows Mobile 用の Embedded SQL CustDB アプリケーションを Embedded Visual C++ を使用して構築する方法の詳細については、「CustDB サンプル・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。

Ultra Light.NET

Visual Studio.NET 用のプロジェクト・ファイルは *samples-dir¥UltraLite.NET¥CustDB* ディレクトリにあります。

Visual Studio.NET を使用して CustDB アプリケーションを Windows Mobile 用に構築する手順については、*samples-dir¥UltraLite.NET¥CustDB¥ce¥ReadMe.html* を参照してください。

Microsoft Windows デスクトップの配備ディレクトリ情報と、追加の Ultra Light.NET サンプルのダウンロード先に関する情報については、*samples-dir¥UltraLite.NET¥CustDB¥Desktop¥ReadMe.html* を参照してください。

Ultra Light for M-Business Anywhere

M-Business Anywhere 用のプロジェクト・ファイルは *samples-dir¥UltraLiteForMBusinessAnywhere¥CustDB* ディレクトリにあります。

M-Business Anywhere を使用して CustDB アプリケーションを構築する手順の詳細については、「Ultra Light for M-Business Anywhere クイック・スタート」『Ultra Light - M-Business Anywhere プログラミング』を参照してください。Windows Mobile、Windows、および Palm OS に適用される手順が用意されています。

レッスン 2：Ultra Light リモート・データベースへのログインとデータ移植

このレッスンでは、次の作業の方法について説明します。

- サンプルの Mobile Link サーバの起動
- サンプルの Ultra Light クライアント・アプリケーションの起動
- Ultra Light へのログイン

ここでは、サンプル・アプリケーションは Mobile Link サーバと同じデスクトップ・コンピュータ上で動作しています。ただし、クライアント・アプリケーションをデバイスに配備しても結果は同じです。

♦ **サンプル・アプリケーションを起動して同期するには、次の手順に従います。**

1. **[スタート] - [プログラム] - [SQL Anywhere 11] - [Mobile Link] - [同期サーバのサンプル]** を選択します。または、次のコマンドを実行します。

   ```
   mlsrv11 -c "DSN=SQL Anywhere 11 CustDB" -zu+ -vcrs
   ```

   コマンド・プロンプトが開き、Mobile Link サーバのステータスを示すメッセージが表示されます。

2. **[スタート] - [プログラム] - [SQL Anywhere 11] - [Ultra Light] - [Windows サンプル・アプリケーション]** を選択します。

3. **[Employee ID]** フィールドに、**50** と入力します。[Enter] を押します。

   アプリケーションが同期し、Mobile Link サーバのメッセージ・ウィンドウに、同期が行われたことを示すメッセージが表示されます。

   デフォルトの同期スクリプトによって、ユーザ 50 がログインしたときにアプリケーションにダウンロードされる顧客、製品、注文のサブセットが決まります。ここでは、承認されていない注文のみがダウンロードされます。

4. アプリケーション・ウィンドウに会社名とサンプル注文情報が表示されることを確認してください。

レッスン 3：CustDB クライアント・アプリケーションの使用

統合データベースとリモート・データベースの両方に ULOrder というテーブルがあります。統合データベースにはすべての注文 (承認済みと未承認) が含まれますが、Ultra Light リモートには、認証されたユーザに応じて、カラムのサブセットだけが表示されます。

テーブル内のカラムは、クライアント・アプリケーションでフィールドとして表示されます。注文を追加するときは、[Customer]、[Product]、[Quantity]、[Price]、[Discount] の各フィールドに入力する必要があります。[Status] や [Notes] などその他の情報を追加することもできます。タイムスタンプ・カラムにより、ローの同期が必要かどうかが判断されます。

♦ 注文をブラウズするには、次の手順に従います。

注文のブラウズは、Ultra Light クライアント・アプリケーションの各バージョンで同様の方法で行います。

注文をブラウズするときは、ローカルの Ultra Light データベースのデータをスクロールします。顧客はアルファベット順でソートされているので、簡単にリストをスクロールして顧客の名前を見つけることができます。

1. 顧客のリストを下にスクロールするには、**[Next]** をクリックします。
2. 顧客のリストを上にスクロールするには、**[Previous]** をクリックします。

♦ 注文情報を追加するには、次の手順に従います。

注文の追加は、Ultra Light クライアント・アプリケーションの各バージョンで同様の方法で行います。

これで、ローカルの Ultra Light データベースでデータが修正されました。このデータは、同期が行われるまで統合データベースとは共有されません。

1. **[Order]** - **[New]**を選択します。
2. **[Customer]** リストで、**Basements R Us** を選択します。
3. **[Product]** リストで、**Screwmaster Drill** を選択します。この製品の価格は **[Price]** フィールドに自動的に設定されます。
4. **[Quantity]** フィールドに **20** と入力します。
5. **[Discount]** フィールドに、**5** (パーセント) と入力し、[Enter] を押します。

♦ 注文の承認、拒否、削除を行うには、次の手順に従います。

ユーザ ID 50 として認証されたユーザはマネージャで、販売担当者と同じ操作に加えて、注文を承認または拒否する権限があります。注文を承認または拒否する場合、注文のステータスを変更し、販売担当者の確認用にメモを追加します。ただし、統合データベースのデータは、同期が行われるまで変更されません。

1. **Apple Street Builders** からの注文を承認します。

a. **[Previous]** をクリックしてこの顧客を探します。
b. **[Order] - [Approve]** をクリックして、注文を承認します。
c. **[Note]** リストで、**Good** を選択します。
d. [Enter] を押します。
注文情報には [Approved] のステータスが表示されます。

2. **Art's Renovations** からの注文を拒否します。
a. リストの次の項目である **Art's Renovations** からの注文を表示します。
b. **[Order] - [Deny]** をクリックして、注文を拒否します。
c. **[Note]** リストで、**[Discount Is Too High]** を選択します。
d. [Enter] を押します。
注文情報には [Denied] のステータスが表示されます。

3. **Awnings R Us** からの注文を削除します。
a. リストの次の項目である Awnings R Us からの注文を表示します。
b. **[Order]** - **[Delete]** を選択してこの注文を削除します。
[OK] をクリックして、削除を確認します。
注文に削除済みのマークが付けられます。ただし、変更内容を統合データベースと同期するまでは、現在のデータが Ultra Light リモートに残ります。

参照

- 「CustDB データベース内のテーブル」 『Mobile Link - クイック・スタート』

レッスン 4：CustDB 統合データベースとの同期

同期するには、Mobile Link サーバが実行されている必要があります。Mobile Link サーバを停止した場合は、サーバを再起動する必要があります。「レッスン 2：Ultra Light リモート・データベースへのログインとデータ移植」 105 ページを参照してください。

このサンプル・アプリケーションで同期を行うと、承認された注文情報はデータベースから削除されます。

Interactive SQL または Sybase Central を使用して統合データベースに接続し、変更内容が同期されたことを確認します。

♦ Ultra Light リモートを同期するには、次の手順に従います。

1. **[ファイル] - [同期]** を選択して、データを同期します。
2. 同期されたことを確認します。
   - リモート・データベースでは、**Apple Street Builders** の承認済み注文が削除されたことを確認することで、必要なトランザクションがすべて完了したことを確認できます。このエントリがないことを確認するには、注文をブラウズします。
   - 統合データベースでは、データを確認して、必要な処理がすべて行われたことを確認できます。

♦ 同期を確認するには、次の手順に従います (Sybase Central の場合)。

1. **[スタート] - [プログラム] - [SQL Anywhere 11] - [Sybase Central]** を選択します。
2. **[接続] - [SQL Anywhere 11 に接続]** を選択します。
3. **[ODBC データ・ソース名]** をクリックします。
4. **[参照]** をクリックし、**[SQL Anywhere 11 CustDB]** を選択します。
5. **[OK]** をクリックします。
6. **[OK]** をクリックします。
7. **[テーブル]** をダブルクリックします。
8. **[ULOrder (DBA)]** をダブルクリックします。
9. **[データ]** タブをクリックし、注文 5100 が承認され、注文 5101 が拒否され、注文 5102 が削除されていることを確認します。

♦ 同期を確認するには、次の手順に従います (Interactive SQL の場合)。

1. Interactive SQL から統合データベースに接続します。
   a. **[スタート] - [プログラム] - [SQL Anywhere 11] - [Interactive SQL]** を選択します。
   b. **[ODBC データ・ソース名] - [SQL Anywhere 11 CustDB]** を選択します。
2. 承認や拒否が同期されたことを確認するため、次の文を実行します。


```
SELECT order_id, status
FROM ULOrder
WHERE status IS NOT NULL;
```

この文の結果として、注文 5100 は承認されており、5101 は拒否されたことがわかります。

3. 削除された注文情報の order_id は 5102 です。次のクエリを実行してもローは返りません。これは、その注文情報がシステムから削除されたことを示します。

```
SELECT *
FROM ULOrder
WHERE order_id = 5102;
```

レッスン 5：Mobile Link 同期スクリプトのブラウズ

CustDB の同期論理は、Mobile Link 同期スクリプトとして統合データベースに格納されています。同期論理によって、統合データベースでダウンロードまたはアップロードの対象となる容量を特定できます。タイムスタンプベースの同期やスナップショット同期などの方法で、テーブル全体またはテーブルの一部 (ローまたはカラムのサブセット) をダウンロードできます。

Sybase Central を使用すると、テーブル、ユーザ、パブリケーションの他に、統合データベースに格納されている同期スクリプトをブラウズできます。Sybase Central は、データベースにこれらのスクリプトを追加するためのプライマリ・ツールです。

custdb.sql ファイルは、ml_add_connection_script または ml_add_table_script を呼び出して、各同期スクリプトを統合データベースに追加します。接続スクリプトは、特定のテーブルに関連付けられていない高いレベルのイベントを制御します。これらのイベントは、各同期の処理中に必要な全般的なタスクを実行するときに使用します。テーブル・スクリプトによって、ローのアップロードの開始や終了、競合の解決、ダウンロードするローの選択など、特定のテーブルの同期に関する特定のイベントでのアクションを実行できます。

CustDB で使用されている同期論理の詳細については、「同期論理のソース・コード」 『Mobile Link - クイック・スタート』を参照してください。

CustDB への同期の実装の詳細については、「同期の設計」 『Mobile Link - クイック・スタート』を参照してください。

♦ **同期スクリプトをブラウズするには、次の手順に従います。**

1. **[スタート] - [プログラム] - [SQL Anywhere 11] - [Sybase Central]** を選択します。
2. **[接続] - [Mobile Link 11 に接続]** を選択します。
3. **[ODBC データ・ソース名]** をクリックします。
4. **[参照]** をクリックし、**[SQL Anywhere 11 CustDB]** を選択します。
5. **[OK]** をクリックします。
6. **[OK]** をクリックします。
7. **[接続スクリプト]** をダブルクリックします。

   右ウィンドウ枠には、同期スクリプトと、それらが関連付けられているイベント・セットがリストされています。Mobile Link サーバが同期処理を実行すると、一連のイベントがトリガされます。このときに、イベントに関連付けられた同期スクリプトが実行されます。同期スクリプトを作成し、同期イベントを割り当てることによって、同期の際に実行されるアクションを制御できます。
8. **[同期テーブル]** をクリックします。
9. 右ウィンドウ枠で、**[ULCustomer]** をダブルクリックします。

   このテーブル固有のスクリプト・セットと、対応するイベントが表示されます。これらのスクリプトは、ULCustomer テーブルのデータがリモート・データベースと同期される方法を制御します。

参照

- 「同期スクリプトの作成」 『Mobile Link - サーバ管理』
- 「Ultra Light クライアント」 131 ページ
- 「接続スクリプト」 『Mobile Link - サーバ管理』
- 「テーブル・スクリプト」 『Mobile Link - サーバ管理』

独自のアプリケーションの構築

サポートされているいずれかのインタフェースを使用して、独自のアプリケーションを構築します。詳細については、次の項を参照してください。

- Ultra Light for C/C++：「チュートリアル：C++ API を使用したアプリケーションの構築」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「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』
- Ultra Light for M-Business Anywhere：「チュートリアル：M-Business Anywhere 用のサンプル・アプリケーション」『Ultra Light - M-Business Anywhere プログラミング』

Ultra Light のパフォーマンスと最適化

目次

Ultra Light では、優れた SQL クエリ・パフォーマンスを提供します。インデックス・スキャン、ダイレクト・ページ・スキャン、テンポラリ・テーブルは、この製品から最高のパフォーマンスを引き出すことができる内部的な最適化の方法です。こうした機能は、実行するクエリ・パフォーマンス・テストの結果に応じて調整することも可能です。

インデックス・スキャンの使用

インデックスとは、1 つ以上のテーブル・カラムに含まれているデータ値の順序に基づく、テーブル内のローのポインタ・セットです。インデックスは、データベース・オブジェクトです。インデックスが作成されると、Ultra Light によって自動的に保持されます。1 つ以上のインデックスを作成すると、クエリのパフォーマンスが向上します。インデックスのタイプによっては、ローの値がユニークであることを保証できます。

インデックスは、一部またはすべてのカラムの値を基に、テーブルのローに順序を付けます。インデックスを作成する場合、インデックス付けするカラムを指定する順序が、インデックスでカラムが実際に出現する順序になります。したがって、インデックスを計画的に使用すると、それによってインデックス・カラムの検索パフォーマンスが大幅に向上します。

Ultra Light では、次のインデックスをサポートしています。インデックスは、シングルカラムとマルチカラム (複合インデックス) が可能です。LONG VARCHAR カラムや LONG BINARY カラムをインデックス付けすることはできません。

インデックス	特性
プライマリ・キー	必須。ユニーク・キーのインスタンスです。プライマリ・キーは 1 つだけ持つことができます。インデックス付けされたカラムの値はユニークである必要があります。NULL は不可です。
外部キー [1]	省略可。インデックス付けされたカラムの値は重複していてかまいません。NULL 入力可かどうかは、NULL を許可するようにカラムが作成されたかどうかで決まります。外部キー・カラムの値は、参照されているテーブルに存在している必要があります。
ユニーク・キー [2]	省略可。インデックス付けされたカラムの値はユニークである必要があります。NULL は不可です。
ユニークでないインデックス	省略可。インデックス付けされたカラムの値は重複していてかまいません。NULL を指定できます。
ユニーク・インデックス	省略可。インデックス付けされたカラムの値が重複していてはいけません。NULL を指定できます。

[1] 外部キーは、プライマリ・キーまたはユニーク・キーを参照できます。

[2] 一意性制約と同じです。

パフォーマンスに関するヒント

- 次のようなカラムにインデックスを作成してください。
  - 定期的に検索する値
  - クエリでテーブルのジョインに使用するカラム
  - ORDER BY 句、GROUP BY 句、または WHERE 句で頻繁に使用するカラム

- 複合インデックスを作成する場合、インデックスの最初のカラムは、クエリの述部で最も頻繁に使用されるカラムにしてください。
- インデックスにより生じる更新管理オーバヘッドがデバイスのメモリに負荷をかけすぎていないことを確認してください。
- 不要なインデックスを作成または保持しないでください。インデックスは、カラムのデータが変更されたときに更新する必要があります。したがって、挿入、更新、削除の各操作はインデックスにも実行されます。
- 大規模なテーブルでインデックスを作成してください。
- 冗長なインデックスは作成しないでください。たとえば、カラム (x、y) に対するインデックスをテーブル T に作成した場合に、カラム (x、y、z) に対する別の既存インデックスが T にあると、冗長的になります。

参照

- 「テンポラリ・テーブルの管理」 123 ページ
- 「ダイレクト・ページ・スキャンの使用」 123 ページ
- 「Ultra Light の実行プランの表示」 360 ページ
- 「複合インデックスについて」 84 ページ
- 「EXPLANATION 関数 [その他]」 402 ページ
- Ultra Light for C/C++ :「GetPlan 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「getPlan メソッド」 『Ultra Light - M-Business Anywhere プログラミング』
- 「Ultra Light page_size 作成パラメータ」 206 ページ

オプティマイザで使用するアクセス方法の決定

Ultra Light オプティマイザは、クエリの最適化に使用するインデックスを選択する際に、優れた最適化方式を取ります。単純なクエリの場合を除き、オプティマイザがクエリ・パフォーマンスの最適化に使用するインデックスをあらかじめ決めることや、そもそもインデックスを使用するかどうかを決めることは、簡単ではありません。複雑さが増すにつれて、選択するインデックスはクエリで必要な句によって変わってきます。場合によっては、FOR READ ONLY 句が存在することで、オプティマイザが、クエリ・パフォーマンスの向上のためにインデックスではなくダイレクト・テーブル・スキャンを選択することもあります。

クエリを最適化する場合、オプティマイザはクエリの要件を確認し、パフォーマンスの向上に使用できるインデックスがないかどうかをチェックします。どのインデックスを使用してもパフォーマンス向上が望めない場合、オプティマイザはインデックスをスキャンせず、テンポラリ・テーブルかダイレクト・ページ・スキャンを使用します。したがって、インデックスを使用して実験し、生成された実行プランを頻繁にチェックして、次の事項を確認する必要があります。

- オプティマイザによって使用されないインデックスを保持していないこと。
- 作成されるテンポラリ・テーブルの数を最小限に抑えていること。「テンポラリ・テーブルの管理」 123 ページを参照してください。

複雑なクエリの場合には、使用されるインデックスを予測するのはさらに困難です。たとえば、クエリに WHERE 述部があり、ORDER BY 句に加えて GROUP BY 句が存在する場合、インデックスが 1 つだけではクエリの探索条件を満たさない可能性があります。したがって、WHERE 述部の選択性要件を満たすインデックスを作成していても、オプティマイザがそれを使用しない可能性があります。代わりに、オプティマイザは、最も多くの処理を必要とする ORDER BY 条件でのパフォーマンスが最も優れているインデックスを使用する可能性があります。

実行プランのチェック

実行プランは、API 呼び出しを使用してプログラムでチェックしたり、Interactive SQL のプラン・ビューワでチェックしたりできます。

- **インデックスが使用されていない場合**　実行プランは次のように表示されます。

  ```
  scan(T)
  ```

- **テンポラリ・テーブルが使用されている場合**　実行プランは次のように表示されます。

  ```
  temp [scan(T)]
  ```

- **インデックスが使用されている場合**　インデックス名が実行プランに含まれている場合は次のように表示されます。

  ```
  scan (T, index_name)
  ```


インデックスのハッシュによるクエリ・パフォーマンスのチューニング

「ハッシュ」の上限として特定のサイズを選択することでクエリのパフォーマンスをチューニングできます。ハッシュ・キーは、インデックス付けされたカラムの実際の値を表します。インデックスのハッシュの目的は、インデックスされた値を特定するためのローの検索、ロード、アンパックという負荷の高い処理を避けることです。必要なだけの実際のロー・データを 1 つのロー ID に含めることで、これらの処理を回避します。

ロー ID を使用することで、Ultra Light がデータベース・ファイル内の実際のロー・データを見つけることができます。ハッシュ・サイズを 0 に設定する (インデックスのハッシュを無効にする) と、インデックス・エントリにはこのロー ID だけが含まれます。ハッシュ・サイズを 0 以外の値に設定すると、ハッシュ・キーも使用されます。ハッシュ・キーには、そのローの変換されたデータすべてまたはその一部を含めることが可能であり、ロー ID とともにインデックスのページに格納されます。

ハッシュ・キーに含まれるロー・データの量は、次の要素によって決まります。

- 設定した最大ハッシュ・サイズ・プロパティ。「最適なハッシュ・サイズの選択」 119 ページを参照してください。
- カラムのデータ型に実際に必要とされるデータ・サイズ。

ハッシュの例

インデックスのハッシュの値は、インデックス付けされたカラムの実際のロー・データの順序を保持しています。たとえば、Employees テーブルの LastName カラムにインデックスを作成した場合、4 つの名前が次の順序で表示されます。

Anders

Anderseck

Andersen

Anderson

最初の 6 文字をハッシュした場合、これらのロー値のハッシュ・キーは次のようになります。

Anders

Anders

Anders

Anders

これらのエントリは同じに見えますが、リストの最初の Anders は実際のロー値 **Anders** を表すために使用されます。リストの最後の Anders は実際のロー値 **Anderson** を表すために使用されます。

ここで、次の文を考えてみます。

```
SELECT *
FROM Employees
WHERE LastName = 'Andersen';
```

Employees テーブルに Andersen に似た名前ばかりが数多く含まれている場合、ハッシュ・キーではパフォーマンスが向上するほどの一意性がありません。この場合、Ultra Light はこの文の条件に実際に一致するハッシュ・キーを決定できません。インデックスのハッシュ・キーの重複があると、Ultra Light が次の処理を行う必要が生じます。

1. 目的のロー ID に一致するテーブル・ローを探す。
2. データをロードしてアンパックし、値を評価する。

パフォーマンスが向上するのは、Ultra Light が適正な数のユニークなハッシュを認識可能であり、クエリ条件の評価がそのままインデックス自体につながる場合のみです。たとえば、Employees テーブルに何千という名前があっても、6 文字によるハッシュで十分なパフォーマンス向上が得られます。ただし、Employees テーブルに Anders* で始まる名前が突出して含まれている場合は、少なくとも 7 文字でハッシュして、ユニークなキーの程度を向上させる必要があります。これにより、この例の冒頭に挙げた 4 つの名前は、次のハッシュ・キーで表されるようになります。

Anders

Anderse

Anderse

Anderso

これにより、4 つローの値のうち、アンパックして評価する必要があるのが、4 つではなく 2 つになります。

参照

- 「Ultra Light max_hash_size 作成パラメータ」 201 ページ
- 「最適なハッシュ・サイズの選択」 119 ページ
- 「Ultra Light のパフォーマンスと最適化」 113 ページ
- 「Ultra Light インデックスの追加」 86 ページ

最適なハッシュ・サイズの選択

Ultra Light のデフォルトの最大ハッシュ・サイズである 4 バイトは、ほとんどの場合に適切であるように選ばれています。値を大きくしてロー ID とともに含めるデータを増やすことができます。ただし、そうすると、インデックスのサイズが大きくなり、複数ページに断片化される場合があります。この変更によって、結果的にデータベースのサイズが大きくなる可能性があります。最大ハッシュ・サイズが大きくなることによる影響は、テーブルに含まれるローの数によって異なります。たとえば、ローの数が少なければ、インデックスのハッシュ・キーは、大きくてもインデックス・ページに収まる可能性があります。その場合、インデックスの断片化は起こりません。

最適なハッシュ・サイズを選択するには、データ型、ローのデータ、データベースのサイズ (特にテーブルに含まれるローの数が多い場合) を考慮してください。

最適なハッシュ・サイズを選択したかどうかを確認する唯一の方法は、ターゲット・デバイスで Ultra Light クライアント・アプリケーションに対してベンチマーク・テストを実行することです。さまざまなハッシュ・サイズによって、アプリケーションとクエリのパフォーマンスにどのように影響するか、さらにデータベースのサイズ自体がどう変化するかを確認する必要があります。

データ型

カラム内の値全体をハッシュする場合は、次の表で、データ型ごとに必要なサイズを確認してください。Ultra Light では、最大ハッシュ・サイズは本当に必要になった場合にのみ使用し、指定した最大ハッシュ・サイズを超えることはありません。カラムのデータ型がバイト制限いっぱいを使用しないかぎり、常により値の小さいハッシュ・サイズを使用します。

データ型	値全体のハッシュに使用されるバイト数
FLOAT、DOUBLE、REAL	ハッシュなし
BIT と TINYINT	1
SMALL INT と SHORT	2
INTEGER、LONG、DATE	4
DATETIME、TIME、TIMESTAMP、BIG	8

データ型	値全体のハッシュに使用されるバイト数
CHAR と VARCHAR	文字列全体をハッシュするには、バイト単位の最大ハッシュ・サイズが、カラムの宣言されたサイズと一致する必要があります。UTF-8 でエンコードされたデータベースでは、宣言されたサイズを常に 2 倍にします。ただし、最大値は 32 バイトです。 たとえば、UTF-8 でエンコードされていないデータベースでカラムを VARCHAR(10) と宣言した場合は、必要なサイズは 10 バイトです。しかし、同じカラムを UTF-8 でエンコードされたデータベースで宣言した場合は、文字列全体のハッシュに使用されるサイズは 20 バイトです。
BINARY	バイト単位の最大ハッシュ・サイズは、カラムの宣言されたサイズと一致する必要があります。 たとえば、カラムを BINARY(30) と宣言した場合は、必要なサイズは 30 バイトです。
UUID	16

たとえば、INTEGER と BINARY (20) と宣言した 2 カラムの複合インデックスに対して最大ハッシュ・サイズを 6 バイトに設定した場合、データ型のサイズの要件に基づいて、次の処理が行われます。

- INTEGER カラムのローの値全体がハッシュされ、インデックスに格納されます。これは、整数のデータ型のハッシュに必要なのは 4 バイトだけだからです。
- ハッシュの最初の 4 バイトは INTEGER カラムで使用されるので、BINARY カラムは最初の 2 バイトだけがハッシュされ、インデックスに格納されます。残りの 2 バイトで、BINARY カラムの適切な量がハッシュされない場合は、最大ハッシュ・サイズを大きくしてください。

ローのデータ

データベースに格納されているデータのローの値も、インデックスのハッシュの有効性に影響します。

たとえば、特定のカラムのエントリ間で共通のプレフィクスを共有している場合は、プレフィクスだけがハッシュされるサイズを選択するとハッシュの効果がなくなる可能性があります。この場合、共通のプレフィクスがハッシュされる以上のサイズを選択する必要があります。共通のプレフィクスが長い場合、値をハッシュしないことも検討してください。

ユニークでないインデックスに重複する値が数多く含まれており、Ultra Light が値全体をハッシュできない場合は、ハッシュによるパフォーマンスの向上は期待できません。

データベース・サイズ

各インデックス・ページには固定のオーバヘッドがありますが、ページのほとんどの領域は実際のインデックス・エントリに使用されます。ハッシュ・サイズを大きくすると、各インデックス・エントリが大きくなり、1 ページに収まるエントリの数が少なくなります。大規模なテーブルになると、大規模なハッシュを使用するインデックスの方が、小規模のハッシュを使用する

テーブルやハッシュを使用しないテーブルより、使用するページ数が多くなります。必要なページ数が増えると、データベース・サイズが増加し、パフォーマンスが低下します。通常、このパフォーマンス低下が生じるのは、キャッシュで保持できるページ数が固定であり、Ultra Light でページのスワップが発生するためです。

次の表に、インデックスに含まれるデータの格納に必要なページ数にハッシュ・サイズがどう影響するかの概要を示します。

テーブル	ページ・サイズ	ハッシュ・サイズ	エントリ数	必要なページ
テーブル A	4 KB	0	1200	3 ページ
テーブル B	4 KB	32 バイト	116	3 ページ
テーブル C	4 KB	32 バイト	1200 エントリ	11 ページ

参照

- 「Ultra Light max_hash_size 作成パラメータ」 201 ページ
- 「Ultra Light のパフォーマンスと最適化」 113 ページ
- 「Ultra Light インデックスの追加」 86 ページ
- 「Ultra Light のデータ型」 328 ページ

最大ハッシュ・サイズの設定

最大ハッシュ・サイズは次の 2 つの方法で設定できます。

- データベースでの最大サイズの**デフォルト**を格納するには、データベースの作成時に max_hash_size 作成プロパティを設定します。デフォルトでインデックスをハッシュしない場合は、この値を 0 に設定します。インデックスをハッシュする場合は、この値を 32 バイト以内の任意の値に変更するか、Ultra Light のデフォルト値 4 バイトを使用できます。
- デフォルトを**変更**する場合は、新しいインデックスを作成するときに特定のハッシュ・サイズを設定します。次のいずれかを行います。
  - Sybase Central で、新しいインデックスを作成するときに [最大ハッシュ・サイズ] プロパティを設定します。
  - SQL で、CREATE TABLE 文または CREATE INDEX 文で WITH MAX HASH SIZE 句を使用します。

参照

- 「Ultra Light CREATE INDEX 文」 480 ページ
- 「Ultra Light CREATE TABLE 文」 489 ページ

テンポラリ・テーブルの管理

一般的に、オプティマイザは、クエリ結果を返すことを目的としたテンポラリ・テーブル作成を回避しようとします。これは、最初のローを返す前に、テンポラリ・テーブル全体を移植する必要があるからです。インデックスが存在する場合、オプティマイザはまずこのインデックスを使用しようとします。テンポラリ・テーブルを作成するのは最後の手段としてのみです。

作成したインデックスでテンポラリ・テーブルの作成を回避できるかどうかを予測することは困難です。そのため、クエリ・プランは必ずチェックし、作成したインデックスが Ultra Light クエリ・オプティマイザによって実際に使用されているかどうかを確認する必要があります。

参照

- 「Ultra Light のテンポラリ・テーブル」 11 ページ
- 「オプティマイザで使用するアクセス方法の決定」 116 ページ
- 「Ultra Light の実行プランの解釈」 361 ページ

ダイレクト・ページ・スキャンの使用

Ultra Light では、データベース・ページから直接情報にアクセスする方が効率的な場合、インデックス・スキャンの代わりにダイレクト・ページ・スキャンを使用します。ダイレクト・ページ・スキャンが使用されるのは、オプティマイザが次の条件を確認した場合のみです。

- 結果を効率的に返すことができる既存のインデックスがない。
- 更新の実行にクエリが使用されない。たとえば、文が FOR READ ONLY (FOR 句が指定されていない場合のデフォルト設定) であると宣言されている場合、またはデータが更新されないことが明らかなようにクエリが記述されている場合です。

Ultra Light では、ローが格納されているページからローを直接読み込むので、クエリ結果は順序に関係なく返されます。その後のクエリ結果の順序は予想できません。ローの順序を予測可能で決定的にするには、ORDER BY 句を使用して一貫性のある順序で結果を取得する必要があります。一方、順序が重要ではない場合は、ORDER BY 句を省略することで、クエリのパフォーマンスを向上させることができます。

注意
アプリケーションのプログラムにテーブル API を使用している場合、ダイレクト・ページ・スキャンは使用できません。

Ultra Light がいつページを直接スキャンするか、または結果を返すのにどのインデックスが使用されたかを確認できます。「オプティマイザで使用するアクセス方法の決定」 116 ページを参照してください。

プライマリ・キー・インデックス順序の復元

10.0.0 以前のバージョンの Ultra Light では、Ultra Light のオプティマイザによって他のインデックスが使用されていない場合には、結果を返すのにプライマリ・キーが使用されていました。その結果、ローはプライマリ・キー・インデックスの順序で並びました。

結果をプライマリ・キーの順序で並べる必要がある場合は、クエリを書き直して ORDER BY 句を含める必要があります。この句を使用してローを並べ替えることが現実的ではない場合は、ORDERED_TABLE_SCAN 接続パラメータの使用を検討してください。

ヒント
できるだけ ORDER BY 句を使用することをおすすめします。

参照

- 「Ultra Light ORDERED_TABLE_SCAN 接続パラメータ (旧式)」 250 ページ
- 「インデックス・スキャンの使用」 114 ページ
- 「Ultra Light SELECT 文」 507 ページ

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

コミットされたトランザクションのフラッシュを遅延させることで、Ultra Light でのリカバリ・ポイントを選択できます。リカバリ・ポイントを使用すると、Ultra Light がコミットを記憶領域に解放したときに、トランザクションに含まれている SQL 文のサブセットがトリガする追加操作オーバヘッドを制御できます。

デフォルトでは、Ultra Light は操作ベースのデフォルト処理を使用しており、トランザクションはコミットされるとすぐに、個別に記憶領域にフラッシュされます。配備環境によっては、このような頻度の高い処理では負荷が重くなり、トランザクション・スループットが制限される可能性があります。このデフォルト処理によるパフォーマンスの低下を防ぐには、ステータスベースの方法を選択できます。特にオートコミット操作に依存しているアプリケーションでは、この方法によって、コミットされたトランザクションの記憶領域へのフラッシュに必要な追加オーバヘッドを遅延させます。

- **チェックポイントでの制御**　独自のチェックポイントを設定し、これを使用して、それまでの期間に実行された作業を解放します。チェックポイントは、単一トランザクション内にも、複数のトランザクションにわたるものでも、必要に応じていくつでも使用できます。
- **グループ化して制御**　トランザクション・カウント・スレッショルドかタイムアウト・スレッショルドまたはその両方を使用して、実行した作業を解放できます。

ステータスに応じてコミットのフラッシュを遅延させる方が、パフォーマンスへの影響が少なく、アプリケーションの設計もすっきりします。これは、アプリケーションが Ultra Light からの応答を待機する必要がないためです。コミット・フラッシュを遅延させることで、作業が完全には完了していないデータへのきめ細かい制御が可能になり、トランザクションへの影響が最小限に抑えられます。たとえば、販売アプリケーションでは、注文が、すべての項目が追加されたり承認されたりする前に、別のアプリケーションに対しても使用可能になります。

ただし、コミット・フラッシュが遅延されるトランザクションのリカバリ性を考慮しておくことが重要です。解放されていないトランザクションはリカバリできません。したがって、アプリケーションのデータ整合性とパフォーマンスとの間のトレードオフを評価しておく必要があります。

参照

- 「Ultra Light COMMIT_FLUSH 接続パラメータ」 239 ページ
- 「Ultra Light commit_flush_count オプション [テンポラリ]」 228 ページ
- 「Ultra Light commit_flush_timeout オプション [テンポラリ]」 230 ページ
- 「Ultra Light CHECKPOINT 文」 478 ページ
- Ultra Light for Embedded SQL：「ULCheckpoint 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「Checkpoint 関数」 『Ultra Light - C/C++ プログラミング』

データベースの暗号化と難読化によるパフォーマンスへの影響

データベースを暗号化すると、Ultra Light に格納される情報のセキュリティを強化できます。ただし、これによってオーバヘッドが 5 ～ 10% 増えるためパフォーマンスが低下することに注意してください。パフォーマンスにどの程度影響するかは、キャッシュのサイズによって異なります。キャッシュがきわめて小さい場合は、暗号化によって著しいオーバヘッドが加わる可能性があります。一方、キャッシュが十分に確保されていれば、まったく違いを感じない場合もあります。シナリオに最適なキャッシュ・サイズを決めるには、ベンチマーク・テストを実行してデータベースのパフォーマンスをグラフで表します。

キャッシュにストレスをかける

さまざまなキャッシュ・サイズでベンチマーク・テストを実行し、パフォーマンスの急激な変化を観察できます。キャッシュは、十分なページのワーキング・セットを保持できる大きさにする必要があります。キャッシュにストレスをかけるには、次の項目を検討してください。

- テーブルに複数のインデックスを作成し、外部キーを追加する。
- ローをランダムに (インデックスの順序とは異なる順で) 挿入する。
- データベース・ページ・サイズの 25% 以上となる多数のローを作成する。
- インデックスのハッシュを 0 以外の値に設定する。このようにサイズを増やすと、必要とされるページ・アクセスも増えます。
- 最小のキャッシュ・サイズからパフォーマンスのグラフ作成を開始する。たとえば、Windows NT の場合は 256 KB (このプラットフォームの最小許容キャッシュ)、その他のすべてのプラットフォームの場合は 64 KB。

キャッシュを大きくしても暗号化されたデータベースのパフォーマンスが向上しない場合は、データを暗号化する代わりに難読化を検討します。難読化すると、パフォーマンスが向上するだけでなく、セキュリティ面での利点もあります。難読化アルゴリズムでは、強力な暗号化と比較して使用するコードが少なく、実行する計算が減ります。単純暗号化のパフォーマンスは、暗号化をまったく行わない場合よりも若干遅いだけです。ただし、最終的には、強力な暗号化を使用するかどうかを決めるときには、セキュリティの要件に従う必要があります。

参照

- 「Ultra Light のパフォーマンスと最適化」 113 ページ
- 「Ultra Light page_size 作成パラメータ」 206 ページ
- 「Ultra Light fips 作成パラメータ」 199 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」 236 ページ
- 「Ultra Light CREATE INDEX 文」 480 ページ

Ultra Light の最適化方法

Ultra Light の配備環境によっては、ニーズに応じてデータベースをチューニングするための専用の設定が必要になる場合があります。

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 11 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。

プラットフォームごとの方法　対応策として、次の 2 つの方法があります。

- Palm OS 5.x にアップグレードする。この方法を推奨します。
- Ultra Light をバージョン 9.0.x にダウングレードする。

データベースの最適化方法　次の事項を行っていることを確認してください。

- **CACHE_SIZE 作成パラメータ**　接続の確立にこのパラメータを使用せず、Ultra Light によるデフォルト値の設定を受け入れてください。独自の値を設定すると、同期のための動的メモリ要件に関連して予期しない問題が発生する可能性があります。
- **page_size 作成データベース・プロパティ**　アプリケーションに最低限必要なページ・サイズを使用する新しいデータベースを作成してください。最小ページ・サイズは 1 KB で、Palm OS 用として一般的に使用されているページ・サイズは 2 KB です。

Mobile Link クライアントとしての Ultra Light

この項では、Mobile Link 同期のために Ultra Light クライアントを設定し実行する方法について説明します。

Ultra Light クライアント

目次

Ultra Light のランタイムには、現場や移動中の社員と企業のバックエンド・システムを結ぶ双方向の同期フレームワークが組み込まれているので、Ultra Light データの同期は、他のリモート・クライアントほど複雑ではありません。組み込みのフレームワークによって、Ultra Light データベース内のデータはすべてデフォルトで自動的に同期されます。Mobile Link 同期を初めて行うときには、このデフォルトの動作を使用できます。業務上、必要になれば、統合データベースと同期する Ultra Light データを変更するように同期をカスタマイズできます。

Ultra Light の詳細については、「Ultra Light の概要」 1 ページを参照してください。SQL Anywhere データベースを Mobile Link クライアントとして使用する方法については、「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。

ヒント
Ultra Light アプリケーションをバックアップする最適な方法として、統合データベースとの同期を行うことが挙げられます。Ultra Light データベースをリストアするには、空のデータベースを作成し、統合データベースと同期してデータを移植します。

ヒント
配備するファイルが複数ある場合や、配備するファイルのバージョンがユーザ ID ごとに異なる場合は、Mobile Link サーバを使用してファイルを転送できます。「Mobile Link ファイル転送の使用」 146 ページを参照してください。

Ultra Light の組み込みの同期機能

Ultra Light では、データ管理レイヤに Mobile Link 同期テクノロジが含まれます。SQL Anywhere リモートとは異なり、同期機能を追加するために Ultra Light の占有容量を増やす必要がありません。

Ultra Light のランタイムに組み込まれている重要な同期機能には、ローのステータスを追跡するメカニズムと、進行状況のカウンタがあります。

ローのステータスを追跡するメカニズム

テーブルとローのステータスの追跡は、データ同期時に特に重要です。Ultra Light データベースの各ローには、ローのステータスを記録する 1 バイトのマーカがあります。Ultra Light では、ローのステータスは、同期だけでなく、トランザクション処理とデータ・リカバリの制御にも使用されます。「Ultra Light でのローのステータス」 13 ページを参照してください。

進行状況のカウンタ

Ultra Light は、進行状況カウンタを使用して堅牢な同期を行います。各アップロードには、識別のためのユニークな ID が付けられます。この ID により、通信エラーが発生した場合にアップロードが正常に行われたかどうかを判別できます。

新しいデータベースを作成したときは、同期進行状況カウンタは必ず 0 に設定されます。進行状況のカウンタの値 0 は、データベースが新しい Ultra Light データベースなので、Mobile Link サーバでこのクライアントのステータス情報をリセットする必要があることを示します。

警告
Ultra Light は、同期が発生すると進行状況カウンタを 1 つずつ増やすので、Ultra Light データベースを別の統合データベースに同期することはできません。進行状況カウンタの値が 0 ではなく、統合データベースに格納されているシーケンス番号と一致しない場合は、Mobile Link 同期はオフセット不一致をレポートし、同期は失敗します。

参照

- 「ml_subscription」 『Mobile Link - サーバ管理』

Ultra Light クライアントの同期の動作のカスタマイズ

Ultra Light にカスタムの同期のサポートを追加するには、次の 3 つの作業を行います。

- **複数のリモート・クライアントを含む同期モデル内でプライマリ・キーの一意性を維持する。** 必須。同期システムでは、プライマリ・キーは、異なるデータベース (リモートと統合) 内の同じローを識別する唯一の方法であり、競合を検出する唯一の方法です。したがって、複数のクライアントがある場合は、以下の規則に従う必要があります。

  ○ 同期される各テーブルには、プライマリ・キーが存在する必要がある。
  ○ プライマリ・キーの値は更新しない。
  ○ プライマリ・キーは、同期されるすべてのデータベース間でユニークでなければならない。

  「ユニークなプライマリ・キーの管理」『Mobile Link - サーバ管理』と「Ultra Light でのプライマリ・キーの一意性」 135 ページを参照してください。

- **小数データが失われないようにデータ・カラムを設定する。** SQL Anywhere 統合データベースの場合は、これは問題ではありません。ただし、Oracle などのデータベースでは、互換性の問題を考慮する必要がある場合があります。たとえば、Ultra Light と Oracle のデータベースでは、タイムスタンプ精度が同じである必要があります。また、Oracle データベースに TIMESTAMP を追加して、Ultra Light リモート・データベースから統合データベースにデータをアップロードするときに秒の小数点以下のデータが失われないようにする必要があります。「Oracle 統合データベース」『Mobile Link - サーバ管理』と「Ultra Light precision 作成パラメータ」 208 ページを参照してください。

- **統合データベースにアップロードするデータ・サブセットを指定する。** オプション。デフォルトですべてのデータを同期しない場合にのみ行います。同期するデータを指定するには、1 つまたは複数のサブセット方法を使用します。「Ultra Light での同期の設計」 139 ページを参照してください。

  たとえば、優先度の高いデータのパブリケーションを作成することができます。ユーザは高速無線ネットワークを経由してこのデータを同期できます。無線ネットワークには使用コストがかかるので、このコストは、業務に必要なものに制限する必要があります。緊急でないデータは後でクレードルから同期できます。

- **Ultra Light アプリケーションから同期を初期化し、セッションに関するパラメータを指定する。** 必須。同期のプログラミング時には、セッションの設定を行ってから、同期操作を初期化します。

  セッションの設定を行うには、同期の通信ストリーム (ネットワーク・プロトコル) とそのストリームのパラメータを選択し、同期スクリプトのバージョンを設定し、Mobile Link ユーザを割り当てます。ただし、ほかにも設定できるパラメータがあります。たとえば、upload_only パラメータと download_only パラメータを使用して同期の方向をデフォルトの双方向から一方向に変更できます。「Ultra Light アプリケーションへの同期の追加」 143 ページを参照してください。

その他の重要な同期の動作は、Mobile Link 同期スクリプトで制御します。各 Mobile Link リモート・データベースは、異なるデータのサブセットを統合データベース内に持つことができるので、複数のスクリプトが必要です。

これらのコードを以下に示します。

- Ultra Light リモート・データベース内のテーブルに更新としてダウンロードするデータ
- リモート・データベースからアップロードされた変更内容に必要な処理

つまり、自分専用の同期スクリプトを作成して、適切な方法でリモート・データベース間でデータを「分割」できます。

参照

- 「Mobile Link 統合データベース」『Mobile Link - サーバ管理』
- 「同期スクリプトの概要」『Mobile Link - サーバ管理』
- 「ダイレクト・ロー・ハンドリング」『Mobile Link - サーバ管理』
- 「リモート・データベース間でのローの分割」『Mobile Link - サーバ管理』

Ultra Light でのプライマリ・キーの一意性

Ultra Light では、Mobile Link でサポートされている任意の方法でプライマリ・キーの一意性を維持できます。「ユニークなプライマリ・キーの管理」 『Mobile Link - サーバ管理』を参照してください。

方法の一つとして、グローバル ID を使用できます。グローバル ID は、GLOBAL AUTOINCREMENT カラムの値が生成される方法に影響します。Mobile Link サーバで複数のクライアントの同期を管理する必要がある場合は、GLOBAL AUTOINCREMENT と宣言したカラムを使用する必要があります。

GLOBAL AUTOINCREMENT は AUTOINCREMENT と同じですが、ドメインが分割されます。Ultra Light では、データベースのグローバル ID に割り当てられた分割からのみカラムの値が設定されます。

参照

- 「Ultra Light global_database_id オプション」 231 ページ

Ultra Light での GLOBAL AUTOINCREMENT の使用

Ultra Light データベースにあるカラムのデフォルト値は、GLOBAL AUTOINCREMENT 型として宣言できます。ただし、これらのカラム ID を自動的に増分するには、その前に Ultra Light リモートのグローバル ID を設定する必要があります。

警告

Mobile Link 同期を介してダウンロードされた GLOBAL AUTOINCREMENT カラムの値は、GLOBAL AUTOINCREMENT 値のカウンタを更新しません。したがって、ある Mobile Link クライアントによって別のクライアントの分割に値が挿入された場合は、エラーが発生する可能性があります。この問題を回避するため、Ultra Light アプリケーションのコピーでは、そのコピー自体の分割にだけ値が挿入されるようにします。

♦ Ultra Light データベースで GLOBAL AUTOINCREMENT カラムを宣言するには、次の手順に従います。

1. データベースの各コピーにユニークなグローバル ID 番号を割り当てます。

   global_database_id データベース・オプションは、Ultra Light データベースに値を設定します。Ultra Light を配備するときには、各データベースに対して必ず異なる ID 番号を割り当てる必要があります。「Ultra Light global_database_id オプション」 231 ページを参照してください。

2. Ultra Light は、Ultra Light のデータベース番号でユニークに識別された分割を使用して、カラムにデフォルト値を設定します。Ultra Light は、次の規則に従います。

   - カラムに現在の分割の値が含まれていない場合、最初のデフォルト値は $pn + 1$ である。ここで、p は分割サイズ、n はグローバル ID 番号を表します。

- カラムに現在の分割の値が含まれていても、そのすべてが $p(n+1)$ 未満であれば、この範囲内でこれまで使用した最大値より 1 大きい値が次のデフォルト値になる。
- デフォルトのカラム値は、現在の分割以外のカラムの値の影響を受けない。つまり、$pn+1$ より小さいか $p(n+1)$ より大きい数には影響されない。Mobile Link 同期を介して別のデータベースからレプリケートされた場合に、このような値が存在する可能性があります。

  たとえば、Ultra Light データベースにグローバル ID 番号 1 を割り当て、分割サイズが 1000 の場合、データベースのデフォルト値は 1001 ～ 2000 の範囲から選択されます。このデータベースの別のコピーで、ID 番号 2 が割り当てられたデータベースからは、2001 ～ 3000 の範囲で同一カラムのデフォルト値が指定されます。
- グローバル ID 番号は負の値に設定できないので、Ultra Light で GLOBAL AUTOINCREMENT カラムに選択される値は常に正の数です。ID 番号の最大値を制限するのは、カラムのデータ型と分割サイズだけです。
- グローバル ID の値を設定しないか、分割内の値がなくなった場合は、カラムに NULL 値が挿入されます。NULL 値が許可されていない場合にローを挿入しようとすると、エラーが発生します。

3. GLOBAL AUTOINCREMENT と宣言されたカラムに使用可能な値がなくなったか、なくなりそうになったら、新しいグローバル ID を設定する必要があります。GLOBAL AUTOINCREMENT の値は、グローバル ID 番号によって識別される分割から、最大値に達するまで選択されます。値を超えると、NULL 値が生成され始めます。新しいグローバル ID 番号を割り当てることで、Ultra Light で別の分割から適切な値を設定できるようになります。

   新しいグローバル ID を選択する方法の一つとして、未使用のグローバル ID の値のプールを管理できます。このプールは、プライマリ・キー・プールと同じ方法で管理されます。「プライマリ・キー・プールの使用」 『SQL Remote』を参照してください。

ヒント
Ultra Light の API を使うと、使用済みの番号の割合を取得できます。戻り値は、これまでに使用された値のパーセンテージを表す 0 ～ 100 の範囲の SHORT 型です。たとえば、値 99 は、未使用の値がほとんど残っていないので、データベースに新しい ID 番号を割り当てる必要があることを示します。この ID 番号の設定方法は、使用するプログラミング・インタフェースによって異なります。

参照

- 「オートインクリメント・カラムの分割サイズの上書き」 138 ページ
- 「Ultra Light global_database_id オプション」 231 ページ
- Ultra Light.NET：「Connection プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「Synchronize 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「setDatabaseID メソッド」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「ULSetDatabaseID 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「GlobalAutoIncrementUsage プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「GetSynchResult 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「getGlobalAutoIncrementUsage メソッド」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「ULGlobalAutoincUsage 関数」 『Ultra Light - C/C++ プログラミング』

最後に割り当てられた GLOBAL AUTOINCREMENT の値の割り出し

最後の挿入操作中に選択された GLOBAL AUTOINCREMENT の値は、取り出すことができます。これらの値はプライマリ・キーで頻繁に使用されるので、生成された値がわかると、最初のローのプライマリ・キーを参照するローをより簡単に挿入できます。次のいずれかを使用して値を確認できます。

- **Ultra Light for C/C++** ULConnection オブジェクトの GetLastIdentity 関数を使用します。「GetLastIdentity 関数」 『Ultra Light - C/C++ プログラミング』を参照してください。
- **Ultra Light.NET** ULConnection クラスの LastIdentity プロパティを使用します。「LastIdentity 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 Anywhere** Connection クラスの GetLastIdentity メソッドを使用します。「getLastIdentity 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。

戻り値は、符号なし 64 ビット整数であるデータベース・データ型 UNSIGNED BIGINT です。この文では最後に割り当てられたデフォルト値がわかるだけなので、間違った結果を取らないために INSERT 文を実行した直後にこの値を取り出してください。

注意
ときには、1 つの INSERT 文に GLOBAL AUTOINCREMENT 型のカラムが複数含まれていることがあります。この場合、戻り値は生成されたデフォルト値のいずれか 1 つですが、そのうちのどの値であるかを判別する信頼できる方法はありません。このため、このような状況を回避するようなデータベースの設計と INSERT 文の記述を行ってください。

オートインクリメント・カラムの分割サイズの上書き

この分割サイズには任意の正の整数を設定できますが、通常、分割サイズは、サイズの値がすべての分割で不足しないように選択されます。

カラムの型が INT または UNSIGNED INT である場合、デフォルトの分割サイズは $2^{16} = 65536$ です。他の型のカラムの場合、デフォルトの分割サイズは $2^{32} = 4294967296$ です。これらのデフォルト値は適切でないことがあるため、分割サイズを明示的に指定するのが最も賢明です。

Ultra Light アプリケーションと SQL Anywhere データベースでは、一部のデータ型のデフォルト分割サイズが異なります。他のデータベースの一貫性を保つ場合は、分割サイズを明示的に宣言します。

♦ Ultra Light 分割値を上書きするには、次の手順に従います (Sybase Central の場合)。

1. Ultra Light データベースに接続します。
2. 選択されたカラムを右クリックして、**[プロパティ]** を選択します。
3. **[値]** タブをクリックします。
4. **[分割サイズ]** フィールドには任意の正の整数を入力します。

♦ Ultra Light でオートインクリメント・カラムを宣言するには、次の手順に従います (Interactive SQL の場合)。

1. Ultra Light データベースに接続します。
2. 分割サイズをカッコで指定した DEFAULT GLOBAL AUTOINCREMENT 句を使用して、CREATE TABLE または ALTER TABLE 文を実行します。「Ultra Light CREATE TABLE 文」 489 ページと「Ultra Light ALTER TABLE 文」 474 ページを参照してください。

   たとえば、次の文では 2 つのカラム (顧客 ID 番号を保持する整数カラム、顧客名を保持する文字列カラム) を持つ簡単なリファレンス・テーブルが作成されます。このテーブルの分割サイズには、5000 を設定する必要があります。

```
CREATE TABLE customer (
  id   INT         DEFAULT GLOBAL AUTOINCREMENT (5000),
  name VARCHAR(128) NOT NULL,
  PRIMARY KEY (id)
);
```

参照

- Ultra Light.NET：「GetColumnPartitionSize メソッド」 『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「GetGlobalAutoincPartitionSize 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「getColumnPartitionSize メソッド」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「ULGlobalAutoincUsage 関数」 『Ultra Light - C/C++ プログラミング』

Ultra Light での同期の設計

Ultra Light データベース内のすべてのデータは、デフォルトで同期されます。Mobile Link リモート・データベースとして Ultra Light を初めて配備する場合、最初は Ultra Light リモート全体を同期することを検討します。

処理に慣れてきたら、同期操作の動作をカスタマイズして、より複雑なビジネス論理を展開できます。カスタム同期動作を設計するときは、次の点を確認する必要があります。業務上の要件が単純である場合は、単一の同期機能だけを使用できます。ただし、非常に複雑な配備では、複数の同期機能を使用すると、必要な同期動作を設定できます。

設計時の確認事項	「はい」の場合の処理
同期からテーブルを除外するかどうか。	テーブル名サフィックス nosync を使用して、同期しないテーブルを指定できます。「Ultra Light の nosync テーブル」 140 ページを参照してください。
データが変更されてないときも含め、常にテーブル全体を同期するかどうか。	テーブル名サフィックス allsync を使用すると、変更が検出されなくてもテーブル全体を同期できます。「Ultra Light の allsync テーブル」 140 ページを参照してください。
テーブル全体を同期するか、特定の条件を満たすローだけを同期するか。重要度または緊急度により、同期の優先度が高いデータがあるかどうか。	パブリケーションには、同期が必要なテーブルをリストするアーティクルが含まれます。アーティクルには、ローが定義した基準を満たすかどうかに基づいてアップロードするローを指定する WHERE 句を含めることができます。 複数のパブリケーションを使用すると、特定の Ultra Light データを他のデータよりも前にアップロードする必要がある場合の優先度の問題に対処できます。「Ultra Light のパブリケーション」 141 ページを参照してください。
外部キーの循環があるので、同期するテーブルの順序を指定する必要があるかどうか。	Table Order 同期パラメータを使用すると、外部キーの循環があるときの同期操作の順序を指定できます。ただし、外部キーの循環は一般に Ultra Light にはおすすめしません。「Ultra Light のテーブルの順序」 142 ページを参照してください。
同期の動作を制御するかどうか。たとえば、アップロードと同時にダウンロードを行うかどうか。また、同期の方向を双方向から一方向に変更するかどうか。	次の場所で適切な同期パラメータを使用します。 ● アプリケーションの同期構造体 (または同期列挙) ● ulsync ユーティリティの -e オプション 「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」 157 ページを参照してください。

設計時の確認事項	「はい」の場合の処理
同期は、時間ベース (スケジュールが組まれている) で行うのか、クレードルでトリガするのか、ユーザが開始するか。または、これらを組み合わせて同期するか。	プログラムで適切なインタフェースを使用して異なる動作を実行できます。場合によっては、HotSync または ActiveSync で同期処理を管理できます。「Ultra Light アプリケーションへの同期の追加」 143 ページを参照してください。
Ultra Light クライアントで TLS を有効にするかどうか。	選択した暗号化アルゴリズムにより、デバイスで実行するプラットフォームに従ってデバイスを設定する方法が決まります。「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページを参照してください。

参照

- 「同期処理」 『Mobile Link - クイック・スタート』
- 「アップロードとダウンロード」 『Mobile Link - クイック・スタート』

Ultra Light の nosync テーブル

テーブル名に **_nosync** サフィックスを追加することで、テーブル全体をアップロード操作から除外することを指定できます。これらの非同期テーブルは、統合データベースでは不要なクライアント固有の永続的なデータ用に使用できます。同期からは除外されていますが、テーブルの使用方法自体は Ultra Light データベース内の他のテーブルとまったく同じです。

_nosync サフィックスを含むテーブルを作成し、名前を変更する場合は、_nosync サフィックスが保持されるようにする必要があります。たとえば、次の RENAME 句を指定した ALTER DBSPACE 文は、変更した名前の末尾が nosync ではなくなるので使用できません。

```
ALTER TABLE purchase_comments_nosync
RENAME comments;
```

修正するには、次のようにこのサフィックスを含む文に書き換えます。

```
ALTER TABLE purchase_comments_nosync
RENAME comments_nosync;
```

またはパブリケーションを使用して同じ効果を上げることができます。「Ultra Light のパブリケーション」 141 ページを参照してください。

Ultra Light の allsync テーブル

テーブル名に **_allsync** サフィックスを追加することで、アップロード時に、前回の同期セッション後に変更されたデータがない場合にも、すべてのテーブル・データを同期するように同期の動作を変更することを指定できます。

一部の Ultra Light アプリケーションでは、allsync テーブルに保管できるユーザ固有またはクライアント固有のデータが必要です。したがって、統合データベースのテンポラリ・テーブルにこのテーブルのデータをアップロードすると、他のスクリプトで同期を制御するのにそのデータを使用できます。そのため、統合データベースにそのデータを保持する必要はありません。たとえば、Ultra Light アプリケーションを使用してチャネルやトピックの中から関連のあるものを示し、この情報を使って適切なローをダウンロードする場合などが該当します。

Ultra Light のパブリケーション

パブリケーションは、同期するデータを指定するアーティクルを定義します。通常は、各アーティクルは 1 つのテーブルの全体を構成します。または、テーブル内にデータのサブセットを定義できます。特定のテーブルのローのサブセットを定義する場合は、オプションで述部 (WHERE 句) を含めることができます。

パブリケーションは、_nosync テーブル・サフィックスの方法よりも柔軟なので、より詳細な制御に使用できます。Ultra Light データベースのデータ・サブセットを**別々**に同期させるには、複数のパブリケーションを使用します。パブリケーションをアップロード専用またはダウンロード専用の同期パラメータと組み合わせると、優先度の高い変更を効率よく同期することが可能です。

パブリケーションの追加

パブリケーションは、Sybase Central を使用して Ultra Light データベースに追加したり、Interactive SQL から追加したりできます。Ultra Light 同期では、パブリケーション内の各アーティクルに、完全なテーブルか WHERE 句のいずれかを入れることができます (Palm OS の HotSync を除く)。

注意

Ultra Light パブリケーションでは、カラムのサブセットの定義と、SUBSCRIBE BY 句がサポートされていません。Ultra Light テーブル内のカラムが SQL Anywhere 統合データベース内のテーブルと完全に一致しない場合は、Mobile Link スクリプトを使用してその違いを解決します。

パブリケーションでテーブル同期順序を設定する必要はありません。配備においてテーブル順序が重要な場合は、Ultra Light データベースを同期するときに Table Order 同期パラメータを設定して、テーブル順序を設定できます。

♦ Ultra Light のデータベースからデータをパブリッシュするには、次の手順に従います (Sybase Central の場合)。

1. Ultra Light プラグインを使用して、Ultra Light データベースに接続します。
2. **[パブリケーション]** フォルダを右クリックし、**[新規]** - **[パブリケーション]** を選択します。
3. 新しいパブリケーションの名前を入力します。**[次へ]** をクリックします。
4. **[テーブル]** タブで、**[一致するテーブル]** リストからテーブルを 1 つ選択します。**[追加]** をクリックします。右側の **[選択したテーブル]** リストに、そのテーブルが表示されます。
5. テーブルを追加します。

6. 必要な場合は、**[WHERE]** タブをクリックし、パブリケーションに含めるローを指定します。カラムのサブセットは指定できません。HotSync 同期を使用している場合は、WHERE 句を指定しないでください。
7. **[完了]** をクリックします。

♦ Ultra Light のデータベースからデータをパブリッシュするには、次の手順に従います (Interactive SQL の場合)。

1. Ultra Light データベースに接続します。
2. CREATE PUBLICATION 文を実行して、新しく作成するパブリケーションの名前とパブリッシュするテーブルの名前を指定します。

参照

- 「Download Only 同期パラメータ」 163 ページ
- 「Upload Only 同期パラメータ」 178 ページ
- 「Additional Parameters 同期パラメータ」 159 ページ
- 「データのパブリッシュ」 『Mobile Link - クライアント管理』
- 「Ultra Light CREATE PUBLICATION 文」 482 ページ
- 「Ultra Light のパブリケーションの操作」 89 ページ
- 「同期スクリプトの概要」 『Mobile Link - サーバ管理』

Ultra Light のテーブルの順序

Table Order 同期パラメータを設定することで、同期操作の順序を制御できます。同期のテーブル順序を指定する場合は、Table Order パラメータをプログラムで使用するか、テスト中に ulsync ユーティリティの一部として使用します。Table Order パラメータは、アップロードするテーブルの順序を指定します。「Additional Parameters 同期パラメータ」 159 ページを参照してください。

テーブル順序は、Ultra Light データベースに次のいずれかがある場合にのみ明示的に設定します。

- 外部キー循環。循環に含まれるすべてのテーブルをリストする必要があります。
- 統合データベースとは異なる外部キー関係

外部キー循環に関連する同期の問題の防止

テーブル順序は、外部キー循環を使用する Ultra Light データベースに特に重要です。循環が形成されるようにテーブルをリンクさせると、循環が発生します。ただし、統合データベースと Ultra Light リモート間の循環が異なる場合は外部キー循環は複雑になるのでおすすめしません。

外部キー循環がある場合は、プライマリ・テーブルへの操作が先に実行され、関連する外部テーブルへの操作が後に実行される順序にテーブルを並べてください。Table Order パラメータによって、外部テーブルへの挿入が外部キーの参照整合性制約を満たしていることが保証されます (削除などの他の操作でも同様です)。

テーブル順序とは別に、同期の問題を防ぐ方法として、参照整合性を確認してから操作をコミットできます。統合データベースが SQL Anywhere データベースの場合は、1 つの外部キーを **check on commit** (コミット時にチェック) に設定します。このように設定すると、外部キーの参照整合性が、操作の開始時ではなくコミット・フェーズでチェックされます。次に例を示します。

```
CREATE TABLE c (
   id INTEGER NOT NULL PRIMARY KEY,
   c_pk INTEGER NOT NULL
);
CREATE TABLE p (
   pk INTEGER NOT NULL PRIMARY KEY,
   c_id INTEGER NOT NULL,
   FOREIGN KEY p_to_c (c_id) REFERENCES c(id)
);
ALTER TABLE c
  ADD FOREIGN KEY c_to_p (c_pk)
  REFERENCES p(pk)
  CHECK ON COMMIT;
```

他のデータベース・ベンダーの統合データベースを使用している場合は、参照総合性のチェックが同様の方法で行われているかどうかを確認します。同じであれば、この方法を実装します。方法が異なる場合は、すべての外部キー循環をなくすようにテーブル関係を再設計してください。

参照

- 「参照整合性と同期」 『Mobile Link - クイック・スタート』

Ultra Light アプリケーションへの同期の追加

Ultra Light では、設定されている通信ストリーム (ネットワーク・プロトコル) で Mobile Link サーバとの特定の接続を開始することで同期が開始されます。ネットワークに直接接続している場合の同期のサポートに加えて、Palm OS デバイスでは HotSync 同期が、Windows Mobile デバイスでは ActiveSync 同期がサポートされます。

接続の定義

各 Ultra Light リモートは、ネットワーク・プロトコルによって Mobile Link サーバと同期します。ネットワーク・プロトコルは同期ストリーム・パラメータを使用して設定します。サポートされているネットワーク・プロトコルには、TCP/IP、HTTP、HTTPS、TLS があります。選択するプロトコルについて、Mobile Link サーバのホストやポートなどその他の必要な接続情報を定義するストリーム・パラメータを指定する必要があります。

また、user_name パラメータと version パラメータで Mobile Link のユーザ情報と同期スクリプトのバージョンを指定する必要もあります。

同期動作の定義

同期動作は、さまざまな同期パラメータを設定することで制御できます。パラメータの設定方法は、使用している Ultra Light インタフェースによって異なります。

検討する必要がある重要な動作は、次のとおりです。

- **同期の方向**　デフォルトでは、同期は双方向です。一方向だけの同期を行うには、upload_only パラメータまたは download_only パラメータを使用します。一方向の同期を実行することによって、同期に必要な時間を最小限に抑えることができます。また、ダウンロード専用同期では、Ultra Light データベースに対してあらかじめすべての変更をコミットする必要はありません。コミットしなかった変更は同期に含まれず、アップロードされません。したがって、不完全なトランザクションでも問題は起こりません。

  ダウンロード専用同期を使用する場合は、ダウンロードと重なっているローがローカル・レベルで変更されていないことを確認してください。ローカル・レベルでデータが変更されていると、SQLE_DOWNLOAD_CONFLICT エラーによって、Ultra Light アプリケーションの同期は失敗します。

- **同期中の変更**　アップロード・フェーズでは、Ultra Light アプリケーションは、読み込み専用として Ultra Light データベースにアクセスできます。ダウンロード・フェーズでは、読み込み／書き込みアクセスが許可されますが、アプリケーションがローを変更してからダウンロードによってこのローが変更されようとすると、ダウンロードが失敗し、ロールバックが行われます。disable_concurrency 同期パラメータを設定することで、同期中のデータへの同時アクセスを無効にすることができます。

♦ Ultra Light アプリケーションに同期コードを追加するには、次の手順に従います。

1. 同期情報の構造体のフィールドとして、セッションに必要な同期パラメータとプロトコル・オプションを指定します。

   たとえば、C/C++ API を使用する場合は、ul_synch_info 構造体に適切な値を設定して、Ultra Light アプリケーションに同期を追加します。

   ```
   ul_synch_info info;
       // define a sync structure named "info"
        ULEnableTcpipSynchronization( &sqlca );
       // use a TCP/IP stream
       conn->InitSynchInfo( &info );
       // initialize the structure
       info.stream = ULSocketStream();
       // specify the Socket Stream
       info.stream_parms= UL_TEXT( "host=myMLserver;port=2439"  );
       // set the MobiLink host information
       info.version = UL_TEXT( "custdb 11.0" );
       // set the MobiLink version information
       info.user_name = UL_TEXT( "50" );
       // set the MobiLink user name
       info.download_only =ul_true;
       // make the synchronization download-only
   ```

2. 同期を初期化します。

   TCP/IP を使用して直接同期する場合は、API 固有の同期関数を呼び出します。これらの関数は、同期操作の成功または失敗を示すブール値を返します。同期が失敗した場合は、別の構造体で詳細なエラー・ステータスのフィールドを検証して、追加のエラー情報を取得できます。

   HotSync 同期の場合は、引数として ul_synch_info 構造体を指定して ULSetSynchInfo 関数を使用する必要があります。ul_synch_info 構造体を指定すると、以後の同期で使用できるように、この構造体が現在のデータベースにアタッチされます。

ActiveSync 同期の場合は、ActiveSync プロバイダからの同期メッセージを取得し、DoSync 関数を使用して ULSynchronize を呼び出す必要があります。

3. 同期の進行状況をユーザにレポートする場合は、observer コールバック関数を使用します。

ヒント
DLL が非常に大きいか、ネットワーク接続の信頼性が低いため、DLL を正常に実行できない場合は、再開可能なダウンロードを実装できます。「失敗したダウンロードの処理」『Mobile Link - サーバ管理』と「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。

参照

- 「Ultra Light での ActiveSync と HotSync の使用」 149 ページ
- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link - サーバ管理』
- 「アップロードとダウンロード」『Mobile Link - クイック・スタート』
- 「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」 157 ページ
- Ultra Light.NET：「Ultra Light アプリケーションの同期」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「データの同期」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C/C++：「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』
- Ultra Light for C/C++：「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』
- Ultra Light for M-Business Anywhere：「データの同期」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「アプリケーションへの同期の追加」『Ultra Light - C/C++ プログラミング』

Mobile Link ファイル転送の使用

M-Business Anywhere を除くすべての Ultra Light ライブラリで、Mobile Link サーバからのファイル転送がサポートされています。M-Business Anywhere には、ファイルの配備または転送の独自のメカニズム (「チャネル同期」) があるので、この機能は必要ありません。

その他のすべての API については、次の場合に Mobile Link のファイル転送メカニズムを使用します。

- 特にセキュリティ対策に企業のファイアウォールを使用していて、複数のファイルを複数のデバイスに配備する必要がある場合。Mobile Link は、企業のファイアウォールを介して同期を処理するように設定済みなので、MLFileTransfer メカニズムにより、アップグレードや他のタイプのファイル転送のためのデバイスのプロビジョニングが非常に便利になります。
- 特定の Mobile Link ユーザ ID を対象とするファイルがある場合。必要なユーザ ID ごとに、Mobile Link サーバで 1 つまたは複数のユーザ固有のディレクトリを作成する必要があります。ファイルのバージョンが 1 つだけの場合は、デフォルトのディレクトリを使用できます。

ファイル転送のしくみ

Mobile Link から開始する 2 種類のファイル転送方法のどちらかを使用して、ファイルをデバイスにダウンロードできます。デスクトップ転送用の mlfiletransfer ユーティリティを実行する方法か、使用する API に適切な関数を呼び出して Ultra Light アプリケーションをコーディングする方法です。どちらの方法でも次の処理が必要です。

1. 転送先を指定します。

   デスクトップから mlfiletransfer ユーティリティを使用するか、API に適切な関数を使用するかに関係なく、転送先のデバイスまたはデスクトップ・コンピュータにあるファイルのローカル・パスとファイル名を設定する必要があります。アプリケーションで、またはエンド・ユーザがこの情報を指定しなかった場合、ファイルは転送元のファイル名で現在のディレクトリに保存されます。

   ターゲットの転送先ディレクトリは、次のようにデバイスのオペレーティング・システムによって異なります。

   - Palm OS では、転送先が外部記憶メディアの場合は、ローカル・パスの前に **vfs:** を付ける必要があります。

     転送先が NULL の場合、mlfiletransfer では、ダウンロードする必要があるファイルが、デバイスのレコード・ストアへの Palm のレコード・データベース (**.pdb* ファイル) であると見なされます。

     ファイル名は、Symbian OS のファイルの命名規則に従っている必要があります。「Palm OS」 50 ページを参照してください。
   - Windows Mobile では、転送先が NULL の場合、ファイルはルート・ディレクトリ (¥) に格納されます。

     ファイル名は、Windows Mobile のファイルの命名規則に従っている必要があります。「Windows Mobile」 50 ページを参照してください。

- デスクトップ・コンピュータでは、転送先が NULL の場合、ファイルは現在のディレクトリに格納されます。

  ファイル名は、デスクトップ・システムのファイルの命名規則に従っている必要があります。「Windows デスクトップ」 50 ページを参照してください。

2. ユーザが識別され、正しいファイルがダウンロードされるように、ユーザ認証を設定します。

   このユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するために使用されます。

3. 使用するストリーム・タイプを設定し、必要なストリームのパラメータを定義します。このパラメータは、Ultra Light で Mobile Link 同期用にサポートされているパラメータと同じです。「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」 157 ページを参照してください。

   ほとんどの同期ストリームでは、Mobile Link サーバのアドレスを識別したり、その他の動作を制御したりするパラメータが必要です。ストリーム・タイプを、プラットフォームに適さない無効な値に設定すると、ストリーム・タイプは TCP/IP に設定されます。

4. 転送メカニズムに必要な動作を指定します。

   たとえば、転送先にファイルが既に存在し、変更されていない場合でも、ダウンロードを強制したり、部分的なダウンロードを再開したりするようにプロパティを設定できます。また、ダウンロードの進行状況をモニタし、レポートするかどうかも設定できます。

5. Mobile Link サーバが実行中であり、-ftr オプションを指定して起動されたことを確認します。

6. 転送を開始し、必要な場合はダウンロードの進行状況をモニタします。

   ダウンロードの進行状況を表示することで、ユーザはダウンロードをキャンセルし、後で再開できます。

参照

- 「-ftr オプション」 『Mobile Link - サーバ管理』
- 「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 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』
- Ultra Light for M-Business Anywhere：サポートされていません。

Ultra Light での ActiveSync と HotSync の使用

目次

Palm OS の HotSync

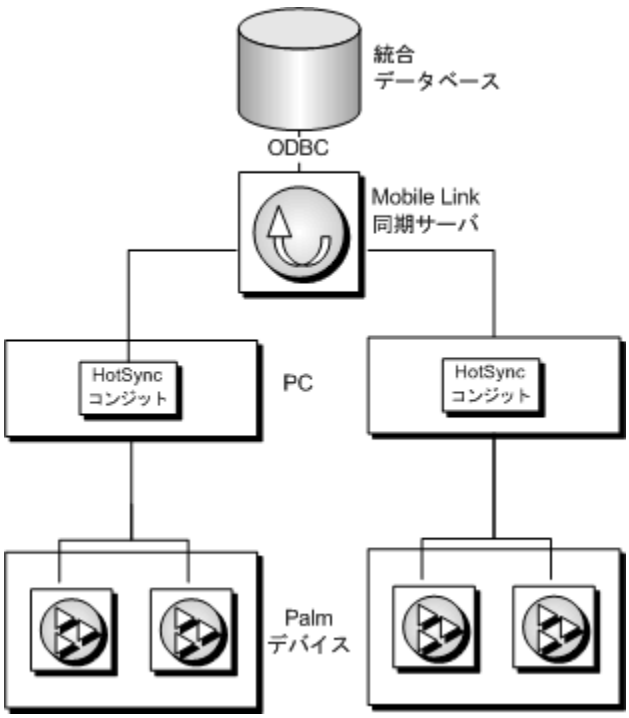
Palm OS デバイスのデータは、Ethernet、Wi-Fi、RAS 接続を介して同期することもできますが、この項では HotSync 同期用にデスクトップとデバイスを設定する方法について説明します。

HotSync 同期では、Palm デバイス上のすべての Ultra Light データベースをユーザのデスクトップから同時に管理できます。HotSync 同期は、デバイスをデスクトップに接続したときに HotSync によって外部的に開始されます。これを行うには、アプリケーションを HotSync 同期用に設定しておく必要があります。「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。

HotSync 同期を使用せずに Ultra Light をアプリケーションから直接同期する場合、エンド・ユーザは各アプリケーションを個別に開いて、各データベースを順番に同期する必要があります。これをプログラムで処理するよう実装するには、API 固有の同期機能を使用して同期を初期化します。「Palm アプリケーションへの TCP/IP、HTTP、HTTPS 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。

HotSync アーキテクチャ

HotSync アーキテクチャには、リモート・データベースに送受信されるデータを同期した統合データベースが必要です。Mobile Link サーバは、これらのデータベース間の同期イベントを管理します。Ultra Light HotSync コンジットは、エンド・ユーザのデスクトップで同期イベントをローカルに管理します。



HotSync コンジット・インストール・ユーティリティ (ulcond11) を使用して、コンジットをインストールし、各 Ultra Light データベースを登録してください。コンジットを使用して各データベースを登録する場合、Ultra Light HotSync コンジットは次のような設定になります。

- アプリケーションに関連付けられたすべてのデータベースの HotSync 同期と登録された作成者 ID を管理する。該当する作成者 ID ごとにコンジットをインストールしてください。
- 適切な接続文字列を使用して各データベースに接続する。複数のデータベースの登録が必要な場合は、ulcond11 ユーティリティで -a オプションを使用します。このオプションでは、各接続文字列を -c オプションで定義した接続文字列に追加します。

これにより、デスクトップから HotSync で同期を初期化することで、すべてのデータベースを同時に同期できます。

参照

- 「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)」 278 ページ
- Ultra Light for C/C++ :「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』
- Ultra Light for Embedded SQL :「アプリケーションへの同期の追加」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「データの同期」 『Ultra Light - M-Business Anywhere プログラミング』

HotSync 同期の概要

1. ul_synch_info 構造体を使用して、必要な同期パラメータをアプリケーションで指定したことを確認してください。SetSynchInfo 関数を呼び出すと、指定した同期パラメータは HotSync 同期用に設定されます。同期パラメータには、Mobile Link サーバとの通信に必要なネットワーク・プロトコル・オプションも含まれています。「Mobile Link 同期のプロトコル・オプションの設定」 152 ページを参照してください。
2. Ultra Light アプリケーションを閉じると、Ultra Light アプリケーションの状態は、データベースとは別のテンポラリファイルに保存されます。「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』を参照してください。
3. Palm デバイスの同期時に、HotSync は特定の作成者 ID について、デスクトップの Ultra Light HotSync コンジットを呼び出します。
4. Ultra Light HotSync コンジットは、適切な接続文字列を使用して登録されたすべてのデータベースに接続し、それらを同期します。SetSynchInfo を適切に呼び出さないデータベースでは、同期が失敗します。
5. アプリケーションの起動時に、直前に保存された Ultra Light アプリケーションの状態がロードされます。「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』を参照してください。

参照

- 「ul_synch_info_a 構造体」 『Ultra Light - C/C++ プログラミング』
- 「SetSynchInfo 関数」 『Ultra Light - C/C++ プログラミング』

Mobile Link 同期のプロトコル・オプションの設定

プロトコル・オプションは、Ultra Light HotSync コンジットから Mobile Link サーバへの接続を定義します。通常は、この情報をアプリケーションの同期コードの一部として追加します。ただし、必要なパラメータは HotSync や ulcond11 から入力することもできます。

ul_synch_info 構造体を使用している場合、引数は次の形式になります。

```
stream=name;parameters
```

stream=name は、ネットワーク・プロトコルのタイプを示します。*stream=name* のデフォルト値は TCPIP です。

parameters は、コンジットが使用するネットワーク・プロトコル・オプションのセットであり、使用するプロトコルの **stream_parms** 引数と同じ形式です。指定されたストリームに対して、*parameters* は、プロトコルの **stream_parms** 引数と同じデフォルトを使用します。

注意

プロトコル・オプションを指定しない場合、コンジットは、ユーザがレジストリで ulcond11 の実行時に入力した可能性のあるプロトコル名とプロトコル・オプションを探します。

有効なネットワーク・プロトコルが見つからないと、デフォルトのプロトコルとプロトコル・オプションが使用されます。このデフォルト・ストリーム・パラメータの設定は、次のとおりです。

```
stream=tcpip;host=localhost
```

参照

- 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 182 ページ
- Ultra Light.NET：「ULStreamType 列挙体」 『Ultra Light - .NET プログラミング』
- Ultra Light for Embedded SQL：「ULSetSynchInfo 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C/C++：「SetSynchInfo 関数」 『Ultra Light - C/C++ 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：「setStream メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

HotSync 操作のデバッグ

HotSync ログ・ファイルは、*Palm-install¥User-dir* ディレクトリに保存されます。デフォルトでは、このファイルには次の情報が含まれます。

- 同期イベントの実行時刻
- 登録された各コンジットのステータス

UL_DEBUG_CONDUIT_LOG 環境変数を設定すると、詳細なデバッグ情報をファイルに取得することができます。情報量のレベルは 2 つあり、取得する必要がある情報量に応じて選択できます。

UL_DEBUG_CONDUIT_LOG = 1　基本的な情報を記録します。たとえば、同期パラメータやレジストリの場所などです。

UL_DEBUG_CONDUIT_LOG = 2　その他の Ultra Light の詳細情報 (入出力操作を含む) を記録します。

HotSync を再起動して、新しい設定を有効にします。

参照

- 「SQL Anywhere 環境変数の概要」『SQL Anywhere サーバ - データベース管理』

Windows Mobile の ActiveSync

Windows Mobile デバイスのデータは、Ethernet や Wi-Fi 接続を介して同期することもできますが、この項では ActiveSync 同期用にデスクトップとデバイスを設定する方法について説明します。別の方法を使用して同期を直接行う場合は、適切な同期機能を使用してアプリケーションのプログラムを設定する必要があります。

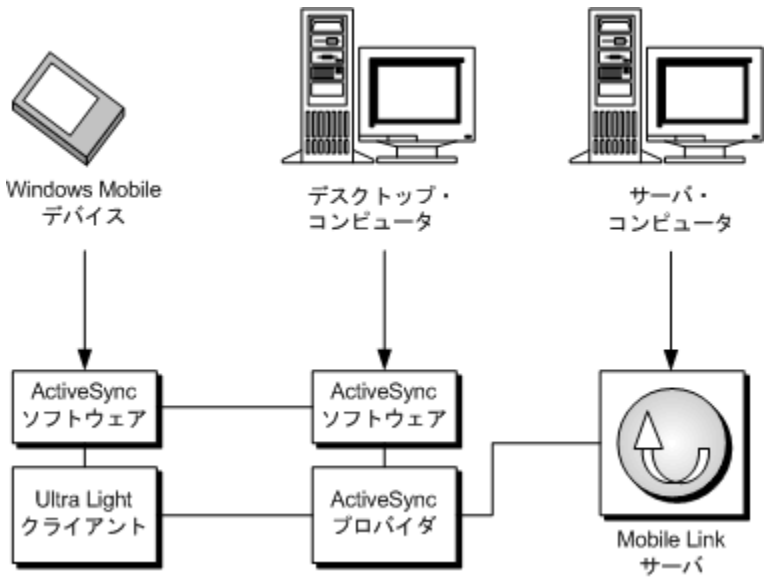
ActiveSync 起動の同期を使用するには、次のことが必要です。

- ActiveSync を使用して ActiveSync 起動の同期を使用するのに必要なすべてのアプリケーションを登録する。
- ActiveSync プロバイダをデスクトップにインストールし、デバイスに配備する。

  プロバイダがサポートされているプラットフォームを確認するには、http://www.ianywhere.jp/developers/technotes/os_components_1101.html を参照してください。

ActiveSync アーキテクチャ

次の図は、ActiveSync アーキテクチャで必要なコンピューティング・レイヤを示しています。



ActiveSync プロバイダは、デスクトップだけでなくデバイスにもインストールしてください。1 台のコンピュータでは 1 つの ActiveSync プロバイダのみ使用できます。ただし、複数の Ultra Light アプリケーションを Windows Mobile デバイスにインストールしている場合は、アプリケーションを同じプロバイダに登録して同時に同期が行われるようにすることができます。

参照

- Ultra Light for C/C++：「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』
- Ultra Light for Embedded SQL：「アプリケーションへの同期の追加」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「データの同期」『Ultra Light - M-Business Anywhere プログラミング』

ActiveSync 同期の概要

1. ActiveSync によって同期セッションが開始されます。
2. ActiveSync プロバイダは、同期の通知メッセージをデバイス上の最初に登録されているアプリケーションに送信します。アプリケーションが実行中でない場合は、起動されます。
3. 登録されたアプリケーションごとに WndProc が起動されます。
4. アプリケーションは、メッセージが ActiveSync からの同期通知メッセージであると判断すると、ULIsSynchronizeMessage を呼び出してデータベース同期プロシージャを起動します。
5. 同期処理が完了すると、アプリケーションは ULSignalSyncIsComplete を呼び出して、同期が終了したことをプロバイダに通知します。
6. プロバイダに登録されている各アプリケーションについて、手順 2 ～ 5 が繰り返されます。

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

目次

Ultra Light の同期パラメータ

同期パラメータは、Ultra Light データベースと Mobile Link サーバ間の同期を制御します。パラメータの設定方法は、使用している Ultra Light インタフェースによって異なります。この章では、パラメータの影響について説明し、これらの設定方法が説明された他の項へのリンクを示します。

注意
この章で説明するパラメータが適用されるのは、Ultra Light リモート・データベースだけです。SQL Anywhere リモート・データベースを同期させる場合は、「Mobile Link SQL Anywhere クライアント・ユーティリティ (dbmlsync)」『Mobile Link - クライアント管理』を参照してください。

API 固有の詳細情報については、次の項を参照してください。

- Ultra Light for Embedded SQL：「ULSynchronize 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「ULSyncParms クラス」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「Synchronize 関数」『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』

必須パラメータ

次のパラメータは必須です。

- **Stream Type** 「Stream Type 同期パラメータ」 175 ページを参照してください。
- **User Name** 「User Name 同期パラメータ」 180 ページを参照してください。
- **Version** 「Version 同期パラメータ」 181 ページを参照してください。

これらのパラメータが設定されていない場合、同期関数は例外 (SQLCode.SQLE_SYNC_INFO_INVALID またはこれと同じもの) をスローします。

パラメータの競合

次のパラメータは、いずれか 1 つだけ指定できます。

- **Download Only** 「Download Only 同期パラメータ」 163 ページを参照してください。
- **Ping** 「Ping 同期パラメータ」 169 ページを参照してください。
- **Upload Only** 「Upload Only 同期パラメータ」 178 ページを参照してください。

これらのパラメータが複数 true に設定されると、同期関数は例外 (たとえば SQLCode.SQLE_SYNC_INFO_INVALID またはこれと同じもの) をスローします。

Additional Parameters 同期パラメータ

この同期パラメータを使用すると、他の定義済みパラメータでは簡単に指定できない追加パラメータをアプリケーションで指定できます。このパラメータ・フィールドには、あまり使用されないパラメータが指定されます。

追加パラメータは、複数の keyword=value 設定をセミコロンで区切った文字列として指定されます。

構文

構文は、使用する API によって異なります。

指定可能な値

次のプロパティを追加パラメータ設定の一部として指定できます。

プロパティ名	説明
AllowDownloadDupRows	同期時に、ダウンロードされたローにプライマリ・キーの重複が検出された場合にエラーにしないようにします。 このプロパティを **0** に設定すると、エラーが発生し、ダウンロードをロールバックします。**1** に設定すると、警告が発生し、ダウンロードを続行します。 このプロパティは Ultra Light C/C++ でのみ使用できます。
CheckpointStore	同期中にデータベースのチェックポイントを追加して、同期プロセス中のデータベースの拡張を制限します。 このプロパティを **1** に設定するとこの機能が有効になります。多くの更新を伴う大量のダウンロードには最適ですが、同期処理に時間がかかるようになります。デフォルトでは **0** に設定されています。
DisableConcurrency	同期のアップロード・フェーズ中は、他のスレッドからのデータベースへのアクセスを禁止します。 このプロパティを **0** に設定すると、データベースへの同時アクセスが許可されます。許可しないためには **1** に設定します。デフォルトでは、このプロパティは **0** に設定されています。

プロパティ名	説明
TableOrder	Ultra Light のデフォルトのテーブルの順序が適切でない場合に、優先同期に必要なテーブルの順序を設定します。 このプロパティを設定して、テーブル名をアップロード順に並べたリストを作成します。Ultra Light ではカンマで区切られたリストを使用し、ulsync ではセミコロンで区切られたリストを使用します。デフォルトでは、外部キーの関係に基づいた順序で並べられます。統合データベースの外部キーが Ultra Light リモートに一致し、外部キー循環がない場合は、通常、デフォルトの順序を使用できます。 一重または二重引用符でテーブル名を囲みます。たとえば、"Customer,Sales" と 'Customer,Sales' の両方が Ultra Light でサポートされています。 同期に含まれていないテーブルを指定すると、そのテーブルは無視されます。リストしないテーブルは、リモート・データベースで定義した外部キーに基づいて適切に保存されます。 ダウンロードのテーブルの順序は、アップロードで定義した順序と同じです。 Ultra Light テーブルが次のいずれかに該当する場合は、テーブル順序を明示的に設定するだけですみます。 ● 外部キー循環に含まれる場合。循環に含まれるすべてのテーブルをリストする必要があります。 ● 統合データベース内の外部キーの関係が異なる場合。

参照

- 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.NET :「AdditionalParms プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「setAdditionalParms メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のように追加パラメータを設定できます。

```
ul_synch_info info;
// ...
info.additional_parms = UL_TEXT(
   "AllowDownloadDupRows=1;
   CheckpointStore=1;
   DisableConcurrency=1;
   TableOrder=Customer,Sales"
);
```


Authentication Parameters 同期パラメータ

Mobile Link イベントの認証パラメータにパラメータを渡します。

構文

構文は、使用する API によって異なります。

備考

パラメータには、ユーザ名やパスワードなどがあります。

このパラメータを使用する場合、パラメータの数も指定してください。「Number of Authentication Parameters パラメータ」 166 ページを参照してください。

指定可能な値

文字列の配列。文字列の値として NULL は使用できませんが、空の文字列は指定できます。

参照

- 「Number of Authentication Parameters パラメータ」 166 ページ
- 「認証パラメータ」 『Mobile Link - サーバ管理』
- 「authenticate_parameters 接続イベント」 『Mobile Link - サーバ管理』
- 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.NET :「AuthenticationParms プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「setAuthenticationParms メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_char * Params[ 3 ] = { UL_TEXT( "parm1" ),
                          UL_TEXT( "parm2" ),
                          UL_TEXT( "parm3" ) };
// ...
info.num_auth_parms = 3;
info.auth_parms = Params;
```

Authentication Status 同期パラメータ

このフィールドは同期によって設定され、Mobile Link のユーザ認証のステータスをレポートします。Mobile Link サーバが、この情報をクライアントに提供します。

構文

構文は、使用する API によって異なります。

指定可能な値

使用できる値は、インタフェース固有の列挙に格納されています。たとえば、C/C++ アプリケーションの列挙は次のようになります。

定数	値	説明
UL_AUTH_STATUS_UNKNOWN	0	認証ステータスが不明です。接続がまだ同期していない可能性があります。
UL_AUTH_STATUS_VALID	1	ユーザ ID とパスワードは、同期時には有効でした。
UL_AUTH_STATUS_VALID_BUT_EXPIRES_SOON	2	ユーザ ID とパスワードは、同期時には有効でしたが、まもなく有効期限が切れます。
UL_AUTH_STATUS_EXPIRED	3	認証に失敗しました。ユーザ ID またはパスワードの有効期限が切れています。
UL_AUTH_STATUS_INVALID	4	認証に失敗しました。不正なユーザ ID またはパスワードです。
UL_AUTH_STATUS_IN_USE	5	認証に失敗しました。ユーザ ID はすでに使用されています。

備考

統合データベースでカスタム **authenticate_user** 同期スクリプトが異なる値を返す場合は、authenticate_user 接続イベントで指定されているルールに従って値が解釈されます。「authenticate_user 接続イベント」『Mobile Link - サーバ管理』を参照してください。

カスタム認証スキームを実装している場合、authenticate_user または authenticate_user_hashed 同期スクリプトは、このパラメータの有効値の 1 つを返します。

このパラメータは Mobile Link サーバによって設定されるため、読み込み専用です。

参照

- 「Mobile Link ユーザ」『Mobile Link - クライアント管理』
- 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.NET：「AuthStatus プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_synch_info info;
// ...
returncode = info.auth_status;
```


Authentication Value 同期パラメータ

このフィールドは同期によって設定され、カスタム Mobile Link のユーザ認証スクリプトの結果をレポートします。Mobile Link サーバが、この情報をクライアントに提供します。

構文

構文は、使用する API によって異なります。

備考

デフォルトの Mobile Link ユーザ認証メカニズムによって設定される値については、authenticate_user 接続イベントと Authentication Status 同期パラメータの項で説明します。

このパラメータは Mobile Link サーバによって設定されるため、読み込み専用です。

参照

- 「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』
- 「Authentication Status 同期パラメータ」 161 ページ
- 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.NET：「AuthValue プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_synch_info info;
// ...
returncode = info.auth_value;
```

Download Only 同期パラメータ

この同期中は、Ultra Light データベースから変更がアップロードされないようにします。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

競合するパラメータ

ping とアップロード専用

備考

ダウンロード専用同期で同期されたリモートがある場合、ダウンロード専用同期がスキャンするログの量を減らすために、定期的に完全な同期を行ってください。そうしないと、ダウンロード専用同期が完了するのに次第に時間がかかるようになります。

ulsync の場合　ダウンロード専用同期が発生する場合、ulsync はデータの変更をアップロードしません。このとき、次の処理が行われます。

- スキーマに関する情報と進行状況のカウンタに格納されている値をアップロードします。
- ダウンロード専用同期中に、リモートでの変更が上書きされないようにします。

これらの処理は、ulsync で Ultra Light データベース・ログをスキャンし、統合データベースへの操作が保留中のローを追跡することで行われます。ulsync で競合が検出されると、データベースがロールバックされ、同期は失敗します。この競合を解決するには、完全な同期 (アップロードとダウンロード) を行う必要があります。

参照

- 「Upload Only 同期パラメータ」 178 ページ
- 「進行状況のカウンタ」 132 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET：「DownloadOnly プロパティ」 『Ultra Light - .NET 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』

例

ulsync では、拡張同期パラメータとしてこのパラメータがサポートされています。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;DownloadOnly=ON;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.download_only = ul_true;
```

Ignored Rows 同期パラメータ

このフィールドは同期によって設定され、同期中にスクリプトがないために Mobile Link サーバによってローが無視されたことを示します。

構文

構文は、使用する API によって異なります。

指定可能な値

ブール式

備考

このパラメータは読み込み専用です。

参照

- 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.NET：「IgnoredRows プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_synch_info info;
// ...
res = info.ignored_rows;
```

Keep Partial Download 同期パラメータ

同期時の通信エラーが原因でダウンロードが失敗したときに、変更をロールバックしないで部分的なダウンロードを保持するかどうかを制御します。

構文

構文は、使用する API によって異なります。

デフォルト

False (ダウンロードが失敗したときにすべての変更をロールバック)

指定可能な値

ブール式

参照

- 「失敗したダウンロードの再開」『Mobile Link - サーバ管理』
- 「Resume Partial Download 同期パラメータ」 171 ページ
- 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.NET：「KeepPartialDownload プロパティ」『Ultra Light - .NET 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』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.keep_partial_download = ul_true;
```

New Password 同期パラメータ

ユーザ名に対する新しい Mobile Link パスワードを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

備考

このパラメータはオプションです。

参照

- 「Mobile Link ユーザ」 『Mobile Link - クライアント管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET :「NewPassword プロパティ」 『Ultra Light - .NET 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』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;NewMobiLinkPwd=mynewpassword;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.new_password = UL_TEXT( "mlnewpass" );
```

Number of Authentication Parameters パラメータ

Mobile Link イベントの認証パラメータに渡す認証パラメータの数を指定します。

構文

構文は、使用する API によって異なります。

デフォルト

カスタム認証スクリプトに渡されるパラメータはありません。

備考

Authentication Parameters とともにパラメータを使用して、カスタム認証スクリプトに情報を提供します。

参照

- 「Authentication Parameters 同期パラメータ」 161 ページ
- 「authenticate_parameters 接続イベント」 『Mobile Link - サーバ管理』
- 「認証パラメータ」 『Mobile Link - サーバ管理』
- 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.NET :「AuthenticationParms プロパティ」 『Ultra Light - .NET 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』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.num_auth_parms = 3;
```

Observer 同期パラメータ

同期をモニタするコールバック関数またはイベント・ハンドラへのポインタを指定します。

構文

構文は、使用する API によって異なります。

参照

- 「User Data 同期パラメータ」 179 ページ
- 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.NET :「ULSyncProgressListener メンバ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「synchronizeWithParm メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.observer=callfunction;
```

Partial Download Retained 同期パラメータ

このフィールドは同期によって設定され、同期時の通信エラーが原因でダウンロードが失敗したときに、変更をロールバックしないで、ダウンロードされたこの変更が適用されたかどうかを示します。

構文

構文は、使用する API によって異なります。

指定可能な値

ブール式

備考

ダウンロード・エラーが発生して部分的なダウンロードが保持された場合、同期時にパラメータが設定されます。

部分的ダウンロードが保持されるのは、Keep Partial Download が true に設定されている場合のみです。「Keep Partial Download 同期パラメータ」 165 ページを参照してください。

参照

- 「失敗したダウンロードの再開」 『Mobile Link - サーバ管理』
- 「Resume Partial Download 同期パラメータ」 171 ページ
- 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.NET：「PartialDownloadRetained プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

このパラメータにアクセスするには、次のように入力します。

```
ul_synch_info info;
// ...
returncode=info.partial_download_retained;
```

Password 同期パラメータ

ユーザ名に対する Mobile Link パスワードを指定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

備考

このパラメータはオプションです。

この Mobile Link のユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するためだけに使用されます。「User Name 同期パラメータ」 180 ページを参照してください。

Mobile Link クライアントにパスワードが設定されている場合は、New Password パラメータを使用してパスワードを変更します。「New Password 同期パラメータ」 166 ページを参照してください。

参照

- 「Mobile Link ユーザ」 『Mobile Link - クライアント管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET :「Password プロパティ」 『Ultra Light - .NET 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』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;MobiLinkPwd=mypassword;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.password = UL_TEXT( "mypassword" );
```

Ping 同期パラメータ

Ultra Light クライアントと Mobile Link サーバ間の通信を確認します。このパラメータが true に設定されている場合は、同期は行われません。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

備考

Mobile Link サーバは、ping を受信すると、統合データベースに接続し、ユーザを認証し、ユーザ認証ステータスと値をクライアントに送信します。

ping に成功した場合、Mobile Link サーバは情報メッセージを発行します。ping に失敗した場合は、エラー・メッセージを発行します。

Mobile Link サーバがコマンド・ライン・オプション -zu+ を指定して実行されていると、Mobile Link ユーザ ID が ml_user システム・テーブルに見つからない場合は Mobile Link サーバがユーザを ml_user に追加します。

Mobile Link サーバに次のスクリプトが存在する場合、ping 要求に対してこれらのスクリプトを実行できます。

- begin_connection
- authenticate_user
- authenticate_user_hashed
- authenticate_parameters
- end_connection

参照

- 「-pi オプション」 『Mobile Link - クライアント管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET :「PingOnly プロパティ」 『Ultra Light - .NET 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』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;Ping=True;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.ping = ul_true;
```

Publications 同期パラメータ

同期させるパブリケーションを指定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync で使用することもできます。

デフォルト

すべてのパブリケーションを同期します。

備考

C/C++ での同期時には、publications 同期パラメータを「パブリケーション・リスト」、つまり、パブリケーション名をカンマで区切ったリストに設定します。

参照

- 「Ultra Light のパブリケーション」 141 ページ
- 「Ultra Light のパブリケーションの操作」 89 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET :「ULPublicationSchema クラス」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「PublicationSchema クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;Publications=UL_PUB_MYPUB1,UL_PUB_MYPUB2;Stream=ht
tp"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.publications = UL_TEXT( "Pubs1,Pubs3" );
```

Resume Partial Download 同期パラメータ

失敗したダウンロードを再開します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

備考

同期によって変更はアップロードされず、失敗したダウンロードで変更のみがダウンロードされます。

参照

- 「失敗したダウンロードの再開」 『Mobile Link - サーバ管理』
- 「Keep Partial Download 同期パラメータ」 165 ページ
- 「Partial Download Retained 同期パラメータ」 168 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET：「ResumePartialDownload プロパティ」 『Ultra Light - .NET 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』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.resume_partial_download = ul_true;
```

Send Column Names 同期パラメータ

アップロード時にカラム名が送信されるように指定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

備考

このオプションは、Mobile Link サーバによってダイレクト・ロー・ハンドリングに使用されます。ダイレクト・ロー・ハンドリングを使用する場合は、このオプションを有効にする必要があります。それ以外の場合、このオプションは効果がありません。「ダイレクト・ロー・ハンドリング」 『Mobile Link - サーバ管理』を参照してください。

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET：「SendColumnNames プロパティ」 『Ultra Light - .NET 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』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;SendColumnNames=true;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.send_column_names = ul_true;
```

Send Download Acknowledgement 同期パラメータ

クライアントがダウンロード確認を提供することを Mobile Link サーバに指示します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET :「SendDownloadAck プロパティ」 『Ultra Light - .NET 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』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;SendDownloadACK=true;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.send_download_ack = ul_true;
```

Stream Error 同期パラメータ

通信エラー・レポート情報を保持する構造体です。

構文

構文は、使用する API によって異なります。

適用対象

このパラメータは、C/C++ インタフェースにのみ適用されます。

指定可能な値

このパラメータにはデフォルト値がないので、サポートされているフィールドの一つを使用して明示的に設定する必要があります。ul_stream_error のフィールドは、次のとおりです。

- **stream_error_code**　必須ではありません。値は常に 0 です。

  エラー番号のリストについては、「Mobile Link 通信エラー・メッセージ」『エラー・メッセージ』を参照してください。エラー・コードのサフィックスについては、*install-dir¥SDK¥Include¥sserror.h* を参照してください。

- **system_error_code**　システム固有のエラー・コード。エラー・コードの詳細については、プラットフォームのマニュアルを参照してください。Windows プラットフォームの場合は、Microsoft Developer Network (MSDN) マニュアルを参照してください。

  Windows における一般的なシステム・エラーを次に示します。

  - **10048 (WSAADDRINUSE)**　アドレスがすでに使用されています。
  - **10053 (WSAECONNABORTED)**　ソフトウェアが接続のアボートを引き起こしました。
  - **10054 (WSAECONNRESET)**　通信の他方の側がソケットを閉じました。
  - **10060 (WSAETIMEDOUT)**　接続がタイムアウトしました。
  - **10061 (WSAECONNREFUSED)**　接続が拒否されました。通常、Mobile Link サーバが稼働していないか、指定されたポートで受信していません。Microsoft Developer Network Web サイトを参照してください。

- **error_string**　アプリケーションが提供するエラー・メッセージ。文字列は空の場合もあれば、空でない場合もあります。空でない error_string は、stream_error_code とともに情報を提供します。たとえば、書き込みエラー (エラー・コード 9) の場合、エラー文字列は、書き込もうとしたバイト数を示します。

- **error_string_length**　廃止予定。エラー文字列バッファのサイズ

備考

Ultra Light C++ コンポーネント以外の Ultra Light アプリケーションは、Sync Result パラメータの一部として通信エラー情報を受け取ります。「Sync Result 同期パラメータ」 177 ページを参照してください。

stream_error フィールドは ul_stream_error 型の構造体です。

```
typedef struct {
    ss_error_code stream_error_code;
    asa_uint16    alignment;
    asa_int32     system_error_code;
    char          error_string[UL_STREAM_ERROR_STRING_SIZE];
} ul_stream_error, * p_ul_stream_error;
```


この構造体は、*install-dir¥SDK¥Include¥sserror.h* で定義されています。

SQLE_COMMUNICATIONS_ERROR をチェックします。

```
Connection conn;
ul_synch_info info;
...
conn.InitSynchInfo( &info );
info.stream_error.error_string = error_buff;
info.stream_error.error_string_length =
                sizeof( error_buff );
if( !conn.Synchronize( &synch_info ) ){
   if( SQLCODE == SQLE_COMMUNICATIONS_ERROR ){
      printf( error_buff );
   // more error handline here
```

参照

- 「ul_synch_info_a 構造体」 『Ultra Light - C/C++ プログラミング』

Stream Type 同期パラメータ

同期に使用する Mobile Link ネットワーク・プロトコルを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

備考

このパラメータは必須です。デフォルト値はありません。

ほとんどのネットワーク・プロトコルでは、Mobile Link サーバ・アドレスや他の動作を識別するプロトコル・オプションが必要です。これらのオプションは、Stream Parameters パラメータで指定します。「Stream Parameters 同期パラメータ」 176 ページを参照してください。

ネットワーク・プロトコルがオプションを必要とする場合は、Stream Parameters を使用してそのオプションを渡すか、あるいは Stream Parameters パラメータを NULL に設定します。

ストリームのタイプには次の種類がありますが、すべてのターゲット・プラットフォームですべての関数を使用できるわけではありません。

ネットワーク・プロトコル	説明
HTTP	HTTP を介して同期します。
HTTPS	HTTPS を介して同期します。 HTTPS プロトコルは、基本のセキュリティ・レイヤとして TLS を使用します。TCP/IP を介して機能します。
TCP/IP	TCP/IP を介して同期します。

ネットワーク・プロトコル	説明
TLS	トランスポート・レイヤ・セキュリティ (TLS) 付きの TCP/IP を介して同期します。TLS は、デジタル証明書とパブリック・キー暗号方式を使用して、クライアント／サーバ通信をセキュリティ保護します。

サポートされる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。

参照

- 「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』
- 「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』
- 「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』
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 182 ページ
- 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.NET :「Stream プロパティ」 『Ultra Light - .NET 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』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定します。

```
Connection conn;
ul_synch_info info;
...
conn.InitSynchInfo( &info );
info.stream = "http";
```

Stream Parameters 同期パラメータ

ネットワーク・プロトコルを設定するオプションを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

NULL

指定可能な値

文字列

備考

このパラメータはオプションです。このパラメータは、セミコロンで区切られたネットワーク・プロトコル・オプションのリストを受け入れます。各オプションは *keyword=value* の形式で、許可されるキーワード・セットはネットワーク・プロトコルによって異なります。

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 182 ページ
- 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.NET :「StreamParms プロパティ」 『Ultra Light - .NET 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』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.stream_parms= UL_TEXT( "host=myserver;port=2439" );
```

Sync Result 同期パラメータ

同期のステータスをレポートします。

構文

構文は、使用する API によって異なります。

備考

このパラメータは Ultra Light によって設定され、読み込み専用です。

C/C++ インタフェースでは、この情報を ul_synch_info 構造体の中で複数の別個のパラメータで受け取ります。それ以外の場合は、この情報は、さまざまな情報がそれぞれ別個のフィールドに格納された複合パラメータとして定義されます。

- **Authentication Status** 認証の成功または失敗をレポートします。「Authentication Status 同期パラメータ」 161 ページを参照してください。
- **Ignored Rows** 無視されるローの数をレポートします。「Ignored Rows 同期パラメータ」 164 ページを参照してください。
- **Stream Error 情報** Stream Error 情報には、Stream Error Code、Stream Error Context、Stream Error ID、Stream Error System が含まれています。「Stream Error 同期パラメータ」 173 ページを参照してください。
- **Upload OK** アップロード・フェーズの成功または失敗をレポートします。「Upload OK 同期パラメータ」 178 ページを参照してください。

参照

- Ultra Light.NET：「ULSyncParms クラス」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「ul_synch_result 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』

Upload OK 同期パラメータ

このフィールドは同期によって設定され、Mobile Link サーバにアップロードされたデータのステータスをレポートします。

構文

構文は、使用する API によって異なります。

備考

このパラメータは Ultra Light によって設定されるため、読み込み専用です。

同期後、このパラメータには、アップロードが成功した場合は **true**、それ以外の場合は **false** が入ります。同期エラーがあったかどうかについてこのパラメータを確認することによって、エラーが発生する前にデータが正常にアップロードされたかどうかを確認できます。

参照

- 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.NET：「UploadOK プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_synch_info info;
// ...
returncode = info.upload_ok;
```

Upload Only 同期パラメータ

現在の同期中にダウンロードは発生しないことを示します。これにより、特に低速の通信リンクでは、通信時間を節約できます。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

Boolean

競合するパラメータ

Download Only、Ping、Resume Partial Download

備考

true に設定すると、クライアントは Mobile Link サーバからのアップロード確認を待ってから、同期セッションを正常終了します。

参照

- 「Ultra Light での同期の設計」 139 ページ
- 「Download Only 同期パラメータ」 163 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET：「UploadOnly プロパティ」『Ultra Light - .NET 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』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;UploadOnly=True;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.upload_only = ul_true;
```

User Data 同期パラメータ

アプリケーション固有の情報を同期 observer で使用できるようにします。

適用対象

C/C++ アプリケーションのみ。Ultra Light.NET などのコンポーネントは、ユーザ・データの処理に別個のパラメータを必要としないので、User Data パラメータはありません。

構文

構文は、使用する API によって異なります。

備考

同期 observer コールバック関数またはイベント・ハンドラの実装時に、User Data パラメータを使用して情報を指定することによって、アプリケーション固有の情報を使用可能にできます。

参照

- 「Observer 同期パラメータ」 167 ページ
- Ultra Light for C/C++：「ul_synch_info_a 構造体」 『Ultra Light - C/C++ プログラミング』

User Name 同期パラメータ

必須。Mobile Link サーバが認証用に使用する文字列。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

備考

このパラメータは必須です。空の文字列と NULL 文字列は一般に拒否されます。

パラメータにはデフォルト値がないので、明示的に設定してください。

リモート ID が使用されている場合は、ユーザ名はユニークである必要はありません。「リモート ID」 『Mobile Link - クライアント管理』を参照してください。

この Mobile Link のユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するためだけに使用されます。「Password 同期パラメータ」 168 ページを参照してください。

同期システムに含まれるユーザの場合は、Mobile Link サーバを使用してユーザ名を登録する必要があります。ユーザ名は、統合データベースの ml_user Mobile Link システム・テーブルの名前カラムに格納されます。

参照

- 「Mobile Link ユーザ」 『Mobile Link - クライアント管理』
- 「Ultra Light ユーザ認証」 『Mobile Link - クライアント管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET：「UserName プロパティ」 『Ultra Light - .NET 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』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。


```
ul_synch_info info;
// ...
info.user_name= UL_TEXT( "remoteA" );
```

Version 同期パラメータ

統合データベースのバージョンを定義します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

備考

このパラメータは必須です。空の文字列と NULL 文字列は一般に拒否されます。

統合データベースの同期スクリプトは、それぞれバージョン文字列で区別されます。たとえば、異なるバージョン文字列によって特定される 2 つの download_cursor スクリプトが存在する場合があります。

参照

- 「スクリプト・バージョン」 『Mobile Link - サーバ管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページ
- 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.NET :「Version プロパティ」 『Ultra Light - .NET 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』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info info;
// ...
info.version = UL_TEXT( "default" );
```


Ultra Light 同期ストリームのネットワーク・プロトコルのオプション

アプリケーションでネットワーク・プロトコルを設定する必要があります。各 Ultra Light データベースは、ネットワーク・プロトコルによって Mobile Link サーバと同期します。使用可能なネットワーク・プロトコルには、TCP/IP、HTTP 、HTTPS、TLS があります。また、Palm OS での HotSync 同期、Windows Mobile での ActiveSync 通知がサポートされています。

設定したネットワーク・プロトコルでは、対応するプロトコル・オプションのセットから選択することによって、Ultra Light アプリケーションが Mobile Link 同期サーバを特定して正しく通信できるようにします。ネットワーク・プロトコルのオプションは、アドレス情報 (ホストとポート) やプロトコル固有の情報などを提供します。使用しているストリーム・タイプに使えるオプションを判別するには、下記を参照してください。

プロトコル・オプションのリストについては、「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。

参照

- 「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』
- 「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページ
- 「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』
- 「Stream Parameters 同期パラメータ」 176 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 292 ページの -x オプション

同期ストリームとオプションの設定

Stream Parameters パラメータを設定することによって、アプリケーション内で Mobile Link サーバを特定するのに必要な情報を提供できます。「Stream Parameters 同期パラメータ」 176 ページを参照してください。

ただし、HotSync を使用している場合、Stream Parameter の値を指定しない場合、または値を NULL として指定した場合は、必要なパラメータを HotSync マネージャから入力できます。「Mobile Link 同期のプロトコル・オプションの設定」 152 ページを参照してください。

Ultra Light 同期の呼び出しでストリーム・パラメータを指定する方法については、次の項を参照してください。

- Ultra Light for C/C++ :「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』
- Ultra Light for Ultra Light.NET :「StreamParms プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「setStreamParms メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

zlib 圧縮に関する注意
zlib 圧縮は、Palm OS ではサポートされていません。

Ultra Light データベースのリファレンス

この項は、Ultra Light データベースのプロパティ、オプション、接続パラメータ、ユーティリティのリファレンスです。

Ultra Light 作成パラメータ

目次

Ultra Light データベースを新規作成する際、設定に作成パラメータを使用します。設定を変更する唯一の方法は、データベースを作成し直すことです。

データベースの作成時に作成パラメータを指定するには、ulcreate、ulinit、または ulload の各ユーティリティを使用するか、サポートされているクライアント・アプリケーションから指定します。

ブール作成パラメータは、YES、Y、ON、TRUE、T、1 のいずれかによってオンになり、NO、N、OFF、FALSE、F、0 のいずれかによってオフになります。パラメータは、大文字と小文字を区別しません。

Ultra Light 作成パラメータは、セミコロンで区切った文字列で指定します。次に例を示します。

```
ulcreate -o "case=respect;utf_encoding=1" -y test.udb
```

または、複数の -o オプションを指定することもできます。

参照

- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light case 作成パラメータ

Ultra Light データベースの文字列を比較するときに大文字と小文字を区別するかどうかを設定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o case=***value*;...

指定可能な値

Ignore、Respect

デフォルト

Ignore

備考

データの大文字と小文字を区別するかしないかは、テーブル、インデックスなどに反映されます。Ultra Light データベースは、デフォルトでは、データは常に入力されたとおりの大文字と小文字で保持されていますが、比較において大文字と小文字を区別しません。識別子 (テーブル名、カラム名など) とユーザ ID は、データベースの大文字と小文字の区別に関係なく、常に大文字と小文字が区別されません。データベースの大文字と小文字の区別の設定に関係なく、パスワードは常に大文字と小文字が区別されます。「Ultra Light の文字列」 320 ページを参照してください。

文字列の比較結果とソート順は、データベースで大文字と小文字を区別するかどうかによって異なります。

ただし、一部の照合では、識別子の大文字と小文字の区別を前提とする場合、特別な注意が必要です。特に、トルコ語の照合では、予期できない複雑なエラーが発生するような大文字と小文字の変換動作があります。最も一般的なエラーは、i または I という文字を含むシステム・オブジェクトが見つからないというものです。

既存のデータベースの大文字と小文字を区別するかどうかは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードで大文字と小文字を区別するかどうかを設定できます。**[新しいデータベースの照合と文字セット]** ページで **[文字列の比較で大文字と小文字を区別する]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light checksum_level 作成パラメータ

データベースのチェックサム検証のレベルを設定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o checksum_level=***value*;...

指定可能な値

0, 1, 2

デフォルト

0

備考

チェックサムは、ディスク、フラッシュ、またはメモリに保管されているページのオフラインの破損を検出するために使用します。チェックサムによって、重要なページの破損が原因で他のデータが破損するのを防ぐことができます。選択するレベルによって、Ultra Light によって各データベース・ページのチェックサムが計算、記録されてから、ページが記憶領域に書き込まれます。

計算されたチェックサムが、記憶領域から読み取られたページのチェックサムと一致しない場合は、ページの保管または取り出し中にページが変更されたか、壊れた可能性があります。チェックサムが一致しなかった場合、データベースでページがロードされるときに、Ultra Light によってデータベースが停止され、致命的なエラーがレポートされます。このエラーは解決できません。Ultra Light データベースを再作成し、データベースのエラーを iAnywhere にレポートしてください。

チェックサムが有効になっている Ultra Light データベースをアンロードしてから再ロードした場合、チェックサムのレベルは保持され、リストアされます。

checksum_level では、次の値がサポートされています。

- **0** データベース・ページにチェックサムを追加しない。
- **1** インデックスや同期ステータスのページなど、重要なデータベース・ページにチェックサムを追加し、ロー・ページには追加しない。
- **2** すべてのデータベース・ページにチェックサムを追加する。

Sybase Central では、データベースを作成するウィザードでチェックサムの使用を設定できます。**[新しいデータベースの記憶領域設定]** ページで **[データベース・ページのチェックサム・レベル]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- checksum_level データベース・プロパティ：「Ultra Light データベース・プロパティ」 221 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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 の最適化方法」 127 ページ
- 「Ultra Light のパフォーマンスと最適化」 113 ページ
- 「Ultra Light page_size 作成パラメータ」 206 ページ
- 「Ultra Light データベースへの接続」 45 ページ

Ultra Light collation 作成パラメータ

データベースの照合を設定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o collation=***value*;...

指定可能な値

文字列

デフォルト

1252Latin1

備考

Ultra Light でサポートされている照合のリストについては、「Ultra Light でサポートされている照合」 39 ページを参照してください。

また、次のコマンドを実行して、Ultra Light でサポートされている照合のリストを表示することもできます。

```
ulcreate -l
```

Sybase Central では、データベースを作成するウィザードで照合を設定できます。**[新しいデータベースの照合と文字セット]** ページで、デフォルトの照合 (1252Latin1) を選択するか、リストから別の照合を選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- 「Ultra Light 文字セット」 38 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light date_format 作成パラメータ

データベースから取り出した日付のフォーマットを設定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o date_format=***value*;...

指定可能な値

文字列

デフォルト

YYYY-MM-DD (ISO 日付フォーマット仕様に準拠)

備考

データ型が DATE の値は、date_format 作成パラメータで設定されているフォーマットで表されます。日付の値は、文字列で表すこともできます。値は、文字列に割り当ててから取り出します。

Ultra Light では日付の単位から日付が構築されます。日付の単位には、年、月、日、曜日、通し日数、時、分、秒 (その要素) があります。

デフォルトの日付のフォーマットと順序は ISO (YYYY-MM-DD) です。たとえば、2006 年 1 月 7 日はこのフォーマットでは 2006-01-07 になります。デフォルトの ISO の日付のフォーマットと順序を使用しない場合は、これらの日付の単位に別のフォーマットと順序を指定します。

フォーマットは次の記号を組み合わせた文字列です。

シンボル	説明
yy	2 桁の年度。
yyyy	4 桁の年度。
mm	2 桁の月、または 2 桁の分 (*hh*:*mm* のように、コロンの後ろに続く場合)。
mmm[*m*...]	月を示す略式文字で、文字数は "m" の数に一致。大文字の M の場合、出力も大文字になります。
d	1 桁の曜日 (0 = 日曜日、6 = 土曜日)。
dd	2 桁の指定月の日。先頭の 0 は必要ありません。
ddd[*d*...]	曜日を示す略式文字。大文字の D の場合、出力も大文字になります。
hh	2 桁の時間。先頭の 0 は必要ありません。
nn	2 桁の分。先頭の 0 は必要ありません。

シンボル	説明
ss[*.ss..*]	秒と秒の少数位。
aa	12 時間表記。正午前の時刻は AM で表します。
pp	12 時間表記。正午後の時刻は PM で表します。
jjj	指定年の日 (1 ～ 366)。

既存のデータベースの日付フォーマットは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

指定可能な値は、上の表に示す記号から構成されます。各記号は、フォーマットされる日付のデータで置き換えられます。

略式文字については、指定した文字数が数えられます。必要な場合は、A.M. または P.M. のインジケータ (ローカライズ可能) が、指定した文字数に対応するバイト数までトランケートされます。

出力の大文字と小文字の制御　文字データを表す記号 (*mmm* など) では、出力の文字を次のように制御できます。

- 記号を大文字で入力すると、フォーマットが大文字で表記されます。たとえば、MMM と入力すると、JAN と表記されます。
- 記号を小文字で入力すると、フォーマットが小文字で表記されます。たとえば、mmm と入力すると、jan と表記されます。
- 大文字と小文字を混ぜて入力すると、使用される言語に適切な文字が Ultra Light により選択されます。たとえば、Mmm と入力すると、英語では May、フランス語では mai と表記されます。

0 埋め込みの制御
数値データを表す記号では、記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

- 記号をすべて大文字または小文字 (MM や mm など) で入力すると、0 埋め込みが行われます。たとえば、yyyy/mm/dd と入力すると、2002/01/01 と表記されます。
- 大文字と小文字を混ぜて入力すると (Mm など)、0 の埋め込みは行われません。たとえば、yyyy/Mm/Dd と入力すると、2002/1/1 と表記されます。

Sybase Central では、データベースを作成するウィザードで日付フォーマットを設定できます。**[新しいデータベース作成パラメータ]** ページで **[日付形式]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light date_order 作成パラメータ」 197 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

例

次の表は、date_format の設定と、2001 年 5 月 21 日 (木) に実行された SELECT CURRENT DATE 文からの出力を示します。

使用した date_format の構文	返された結果
yyyy/mm/dd/ddd	2001/05/21/thu
jjj	141
mmm yyyy	may 2001
mm-yyyy	05-2001

Ultra Light date_order 作成パラメータ

日付フォーマットの解釈を制御します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o date_order=***value*;...

指定可能な値

MDY、YMD、DMY

デフォルト

YMD (ISO 日付フォーマット仕様に準拠)

備考

データ型が DATE の値は、date_format 作成パラメータで設定されているフォーマットで表されます。日付の値は、文字列で表すこともできます。値は、文字列に割り当ててから取り出します。

Ultra Light では日付の単位から日付が構築されます。日付の単位には、年、月、日、曜日、通し日数、時、分、秒 (その要素) があります。

デフォルトの日付のフォーマットと順序は ISO (YYYY-MM-DD) です。たとえば、2006 年 1 月 7 日はこのフォーマットでは 2006-01-07 になります。デフォルトの ISO の日付のフォーマットと順序を使用しない場合は、これらの日付の単位に別のフォーマットと順序を指定します。

既存のデータベースの日付順は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードで日付順を設定できます。**[新しいデータベース作成パラメータ]** ページで **[日付順]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light date_format 作成パラメータ」 194 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

例

値によって、日付 10/11/12 の変換方法が異なります。

使用される構文	変換
MDY	1912 年 10 月 11 日
YMD	1910 年 11 月 12 日
DMY	1912 年 11 月 10 日

Ultra Light fips 作成パラメータ

新しいデータベースで AES または AES_FIPS の強力な暗号化方式のどちらを使用するかを制御します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o fips=***value*;**KEY=***value*;...

指定可能な値

Yes (AES_FIPS を使用)、No (AES を使用)

デフォルト

Yes

備考

データベースの暗号化タイプを変更するには、fips 作成パラメータまたは obfuscate 作成パラメータを使用してデータベースを作成し直す必要があります。データベースの暗号化キーを変更するには、新しい暗号化キーを Connection オブジェクトで指定します。データベースに接続するユーザは、接続するたびにキーを指定する必要があります。

Sybase Central では、データベースを作成するウィザードで強力な暗号化を設定できます。**[新しいデータベースの記憶領域設定]** ページで **[AES FIPS アルゴリズム]** オプションを選択します。暗号化キーを設定し、確認する必要もあります。

Palm OS では、ULEnableFipsStrongEncryption メソッドを使用します。

その他すべてのプラットフォームでは、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

FIPS 対応のデータベースを配備するには、プラットフォームに適切なライブラリをすべてコピーします。「AES_FIP データベース暗号化を使用した Ultra Light の配備」 61 ページを参照してください。

参照

- 「強力な暗号化」『SQL Anywhere サーバ - データベース管理』
- 「Ultra Light データベースの保護」 42 ページ
- 「Ultra Light obfuscate 作成パラメータ」 205 ページ
- 「Ultra Light DBKEY 接続パラメータ」 244 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- 「Palm OS の暗号化キーの保存、取り出し、クリア」『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C/C++：「ULChangeEncryptionKey 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」『Ultra Light - .NET プログラミング』
- Ultra Light.NET：「ChangeEncryptionKey メソッド」『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 M-Business Anywhere：「ChangeEncryptionKey メソッド」『Ultra Light - M-Business Anywhere プログラミング』
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light max_hash_size 作成パラメータ

デフォルトの最大インデックス・ハッシュ・サイズをバイト単位で設定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o max_hash_size=***value*;...

指定可能な値

0 ～ 32 バイト

デフォルト

4 バイト

備考

ハッシュは、インデックス・エントリのオプション部分で、インデックスのページに格納されます。ハッシュによって、インデックス・カラムの実際のローの値が、インデックスの順序を保持したまま、それに相当する数値 (キー) に変換されます。キーのサイズと、実際の値がハッシュされる長さは、設定するハッシュのサイズで決まります。

Ultra Light では、ロー ID によって、テーブル内の実際のデータのローを見つけることができます。ロー ID は、常にインデックス・エントリの一部です。ハッシュ・サイズを 0 に設定する (インデックスのハッシュを無効にする) と、インデックス・エントリにはこのロー ID だけが含まれます。ハッシュ・サイズを 0 以外に設定すると、そのローの変換されたデータすべてまたはその一部を含むハッシュ・キーがロー ID とともにインデックスのページに格納されます。Ultra Light で必ずしもデータを検索、ロード、アンパックしてから実際のローの値を比較する必要がないため、これらのインデックス・カラムに対するクエリのパフォーマンスを向上できます。

データベースのデフォルトのハッシュ・サイズを決定するには、クエリの効率とデータベースのサイズのトレードオフを評価する必要があります。最大ハッシュ値が大きいほど、データベースのサイズが大きくなります。

Ultra Light では、このパラメータで指定されている最大値を上限として、カラムのデータ型に必要なバイト数のみが使用されます。デフォルトのハッシュ・サイズは、インデックスを作成するときにサイズを設定しなかった場合にだけ使用されます。デフォルトのハッシュ・サイズを 0 に設定すると、Ultra Light では、ローの値はハッシュされません。

既存のインデックスのハッシュ・サイズは変更できません。Sybase Central で Ultra Light の**インデックス作成ウィザード**を使用して、または CREATE INDEX 文か CREATE TABLE 文で WITH MAX SIZE 句を使用して新しいインデックスを作成するときにデフォルト値を変更できます。

カラムを DOUBLE、FLOAT、または REAL として宣言した場合は、ハッシュは使用されません。ハッシュ・サイズは常に無視されます。

Sybase Central では、データベースを作成するウィザードで最大ハッシュ・サイズを設定できます。**[新しいデータベースの記憶領域設定]** ページで **[インデックスの最大ハッシュ・サイズ]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light のパフォーマンスと最適化」 113 ページ
- 「Ultra Light のインデックスの操作」 84 ページ
- 「最適なハッシュ・サイズの選択」 119 ページ
- 「Ultra Light CREATE INDEX 文」 480 ページ
- 「Ultra Light CREATE TABLE 文」 489 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light nearest_century 作成パラメータ

文字列から日付への変換で、2 桁の年の解釈を制御します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o nearest_century=***value*;...

指定可能な値

整数 (0 ～ 100)

デフォルト

50

備考

Ultra Light では、期待値が日付の値であれば、年が文字列内で 2 桁だけで表されている場合でも、文字列を日付に自動的に変換します。年が 2 桁の場合は、適切なロールオーバ値を設定する必要があります。この値より小さい 2 桁の年は 20*yy* に変換され、この値以上の年は 19*yy* に変換されます。

適切なロールオーバ値を決定するには、次の点を考慮します。

- **2 桁の年の使用**　年が 2 桁ではない場合は、基準年への変換は適用されません。設定した nearest_century 値よりも小さい 2 桁の年は 20*yy* に変換され、nearest_century 値以上の年は 19*yy* に変換されます。

  4 桁の年を格納して、間違った変換の問題を防ぐことをおすすめします。「あいまいさのない日付と時刻の使用」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

- **統合データベースの互換性**　たとえば、従来の SQL Anywhere では、年に 1900 を加算していました。Adaptive Server Enterprise では基準年を使用するので、*yy* が 50 より小さい場合は 20*yy* になります。

- **日付の意味：過去または未来**　誕生年は、過去のことなので、小さいロールオーバ値が必要です。したがって、*yy* が 20 未満の年は 20*yy* に設定してください。これに対して、日付が有効期限の場合は、未来のことなので、大きい値の方が理にかなっています。

既存のデータベースの基準年は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードで基準年を設定できます。**[新しいデータベース作成パラメータ]** ページで**[基準年]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「あいまいな文字列から日付への変換」『SQL Anywhere サーバ - SQL リファレンス』
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」『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』
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light obfuscate 作成パラメータ

データベース内のデータの難読化を制御します。難読化は単純暗号化です。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o obfuscate=***value*;...

指定可能な値

ブール式

デフォルト

0 (データベースの難読化なし)

備考

単純暗号化は、難読化と同じです。これにより第三者は、ディスク・ユーティリティを使用してファイルを表示し、データベースのデータを解読することが困難になります。単純暗号化では、データベースの暗号化のためのキーは不要です。

正しい暗号化キーがないとデータベースにアクセスできないようにするには、強力な暗号化を使用してください。「Ultra Light fips 作成パラメータ」 199 ページを参照してください。

Sybase Central では、データベースを作成するウィザードで難読化を設定できます。**[新しいデータベースの記憶領域設定]** ページで **[簡易暗号化を使用 (obfuscation)]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「単純暗号化」 『SQL Anywhere サーバ - データベース管理』
- 「Ultra Light データベースの保護」 42 ページ
- 「Ultra Light fips 作成パラメータ」 199 ページ
- 「Ultra Light DBKEY 接続パラメータ」 244 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light page_size 作成パラメータ

データベース・ページ・サイズを定義します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o page_size=***size*[**k**];...

指定可能な値

1k、2k、4k、8k、16k

デフォルト

4k

備考

Ultra Light データベースはページ単位で保管され、I/O 操作はすべてページ単位で行われます。選択したページ・サイズが、データベースのパフォーマンスまたはサイズに影響することがあります。

前述の値以外の値を使用すると、サイズはその値より大きい次の値に変更されます。単位を指定しない場合、デフォルトはバイトです。

プラットフォームの動的メモリの制約が大きい場合は、ページ・サイズを小さくして、同期メモリ要件による影響を低減させることを検討してください。

ページ・サイズを選択するときは、次のガイドラインに従ってください。

- **データベース・サイズ**　大規模なデータベースでは、通常、より大きなページ・サイズの方が有利です。より大きなページではより多くの情報を保持できるため、特に、ページの半分以上のサイズのローを挿入する場合に領域を効率よく使用できます。ページが大きいほど、必要なページのスワップが少なくなります。
- **ローの数**　ロー (BLOB を除く) はページに収まる必要があるので、ページ・サイズによって、パックされたローの最大サイズと、各ページに格納できるローの数が決まります。1 ページを読み込んで 1 つのローの値を取得すると、次のいくつかのローの内容をメモリにロードできるという 2 次的な効果がある場合があります。「ローのパックとテーブル定義」 74 ページを参照してください。
- **クエリのタイプ**　通常、ページ・サイズが小さいと、ランダムな場所から比較的少ない数のローを取り出すクエリに有利な傾向があります。これに対して、ページ・サイズが大きいと、テーブルの逐次スキャンを実行するクエリに有利です。
- **キャッシュ・サイズ**　ページ・サイズが大きいと、必要なキャッシュ・サイズも大きくなる場合があります。キャッシュに十分なページを格納できない場合、Ultra Light は頻繁に使用されるページをディスクにスワップし始めるため、パフォーマンスが低下します。「Ultra Light CACHE_SIZE 接続パラメータ」 236 ページを参照してください。
- **インデックス・エントリ**　ページ・サイズは、インデックスにも影響を与えます。データベースのページが大きいほど、保持できるインデックス・エントリ数が多くなります。「Ultra Light のインデックスの操作」 84 ページを参照してください。

- **デバイス・メモリ**　特に、メモリが限られている小型のデバイスでデータベースを実行する場合に便利です。たとえば、1 MB のメモリには、1 ページを 1 KB とすると 1000 ページ格納できますが、1 ページ 4 KB であれば 250 ページしか格納できません。

既存のデータベースのページ・サイズは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードでページ・サイズを設定できます。[新しいデータベースの記憶領域設定] ページで、適切なバイト値を選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light の最適化方法」 127 ページ
- 「ローのパックとテーブル定義」 74 ページ
- 「Ultra Light case 作成パラメータ」 189 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」 236 ページ
- 「Ultra Light RESERVE_SIZE 接続パラメータ」 256 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」『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』
- 「作成パラメータの値へのアクセス」 35 ページ

例

データベースのページ・サイズを 8 KB に設定するには、**page_size=8k** または **page_size=8192** を指定します。

```
ulcreate test.udb -o page_size=8k
```


Ultra Light precision 作成パラメータ

計算結果が小数の場合の最大桁数を指定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o precision=***value*;...

指定可能な値

整数 (1 ～ 127)

デフォルト

30

備考

小数点の位置は数値の精度 (precision) と位取り (scale) で決まります。精度は小数点の左右の合計桁数です。位取りは、計算結果が最大精度にトランケートされるときの小数点以下の最小桁数です。

適切な小数点の位置を決定するには、次の点を考慮します。

- **実行する計算のタイプ**　掛け算、割り算、足し算、引き算、集合関数はすべて結果が最大精度を超える可能性があります。

  たとえば、DECIMAL(8,2) と DECIMAL(9,2) を掛けると、結果には DECIMAL(17,4) が必要です。precision が 15 の場合、15 桁のみが結果に保持されます。scale が 4 の場合、結果は DECIMAL(15,4) になります。scale が 2 の場合、結果は DECIMAL(15.2) になります。どちらの場合も、オーバフロー・エラーの可能性があります。

- **scale と precision の値の関係**　scale は、小数点以下の桁数を設定し、負の数や precision より大きい値にはできません。

既存のデータベースの精度は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

統合データベースとして Oracle データベースを使用している場合、すべての Ultra Light リモートと Oracle 統合データベースが精度として同じ値を使用する必要があります。

Sybase Central では、データベースを作成するウィザードで精度を設定できます。**[新しいデータベース作成パラメータ]** ページで **[精度]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light scale 作成パラメータ」 210 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light scale 作成パラメータ

計算結果が最大 precision にトランケートされる場合の、小数点以下の最小桁数を指定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o scale=***value*;...

指定可能な値

整数 (0 ～ 127)

デフォルト

6

備考

小数点の位置は数値の精度 (precision) と位取り (scale) で決まります。精度は小数点の左右の合計桁数です。位取りは、計算結果が最大精度にトランケートされるときの小数点以下の最小桁数です。

適切な小数点の位置を決定するには、次の点を考慮します。

- **実行する計算のタイプ**　掛け算、割り算、足し算、引き算、集合関数はすべて結果が最大精度を超える可能性があります。

  たとえば、DECIMAL(8,2) と DECIMAL(9,2) を掛けると、結果には DECIMAL(17,4) が必要です。precision が 15 の場合、15 桁のみが結果に保持されます。scale が 4 の場合、結果は DECIMAL(15,4) になります。scale が 2 の場合、結果は DECIMAL(15.2) になります。どちらの場合も、オーバフロー・エラーの可能性があります。

- **scale と precision の値の関係**　scale は、小数点以下の桁数を設定し、負の数や precision より大きい値にはできません。

既存のデータベースの位取りは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードで位取りを設定できます。**[新しいデータベース作成パラメータ]** ページで **[位取り]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light precision 作成パラメータ」 208 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

例

DECIMAL(8,2) と DECIMAL(9,2) を掛けると、結果には DECIMAL(17,4) が必要です。precision が 15 の場合、15 桁のみが結果に保持されます。scale が 4 の場合、結果は DECIMAL(15,4) になります。scale が 2 の場合、結果は DECIMAL(15.2) になります。どちらの場合も、オーバフローが発生する可能性があります。

Ultra Light time_format 作成パラメータ

データベースから取り出した時刻のフォーマットを設定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o time_format=***value*;...

指定可能な値

文字列 (下記の記号の組み合わせ)

デフォルト

HH:*NN*:*SS*.*sss*

備考

Ultra Light では time_format 作成パラメータで設定した時間の単位を使用して時間が記述されます。時間の単位には、時、分、秒、ミリ秒があります。

時間の値は、文字列で表すこともできます。時間の値は、文字列変数に割り当ててから取り出します。

ISO (HH:MM:SS) がデフォルトの時間フォーマットです。たとえば、「真夜中」はこのフォーマットでは 00:00:00 になります。デフォルトの ISO の時間フォーマットを使用しない場合は、これらの時間の単位に別のフォーマットと順序を指定します。

フォーマットは次の記号を組み合わせた文字列です。

シンボル	説明
HH	2 桁の時間 (24 時間表記)。
NN	2 桁の分。
MM	コロンの後の場合は、2 桁の分 (*hh*:*mm* など)。
SS[.s...]	2 桁の秒と、省略可能な秒の小数位。

既存のデータベースの時間フォーマットは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

各記号は、フォーマットしようとする日付のデータで置き換えられます。数字の出力ではなく文字を表すフォーマット記号は、大文字で入力でき、その場合、置き換えられる文字も大文字になります。フォーマット文字列で大文字と小文字を混ぜて使用すると、数字の前に 0 が付きません。

記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

- 記号をすべて大文字または小文字 (HH や hh など) で入力すると、0 埋め込みが行われます。たとえば HH:NN:SS と入力すると、01:01:01 と表記されます。

- 大文字と小文字を混ぜて入力すると (Hh など)、0 埋め込みは行われません。たとえば Hh:Nn:Ss と入力すると、1:1:1 と表記されます。

Sybase Central では、データベースを作成するウィザードで時間フォーマットを設定できます。**[新しいデータベース作成パラメータ]** ページで **[時間形式]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light timestamp_format 作成パラメータ」 214 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」『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』
- 「作成パラメータの値へのアクセス」 35 ページ

例

3:30 PM にトランザクションが実行された場合、デフォルトの time_format 構文 *HH*:*NN*:*SS.sss* では、結果は次のようになります。

15:30:55.0

Ultra Light timestamp_format 作成パラメータ

データベースから取り出したタイムスタンプのフォーマットを設定します。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o timestamp_format=***value*;...

指定可能な値

文字列

デフォルト

YYYY-MM-DD HH:NN:SS.SSS

備考

Ultra Light では、date_format プロパティと time_format 作成パラメータで設定した日付と時間の単位からタイムスタンプが作成されます。日付と時間には合計 7 つの単位があります (年、月、日、時、分、秒、ミリ秒)。

タイムスタンプの値は、文字列で表すこともできます。タイムスタンプの値は、文字列変数に割り当ててから取り出します。

一般に、タイムスタンプ・カラムによって、統合データベースと同期させるときにデータの整合性が維持されます。タイムスタンプを使用して、各ユーザが最後に同期した時刻を追跡することで、複数のリモート・データベース間でデータの同時更新がいつ発生したかを特定できます。

ヒント
タイムスタンプとその増分は、統合データベースと Ultra Light データベースで同じ値になるようにしてください。これらの作成パラメータを統合データベースのものと合うように設定すると、タイムスタンプの不一致を回避できます。

フォーマットは次の記号を組み合わせた文字列です。

シンボル	説明
YY	2 桁の年度。
YYYY	4 桁の年度。
MM	2 桁の月、または 2 桁の分 (*hh*:*mm* のように、コロンの後ろに続く場合)。
MMM[*m*...]	月を示す略式文字で、文字数は "m" の数に一致。大文字の M の場合、出力も大文字になります。
D	1 桁の曜日 (0 = 日曜日、6 = 土曜日)。
DD	2 桁の指定月の日。先頭の 0 は必要ありません。

シンボル	説明
DDD[*d...*]	曜日を示す略式文字。大文字の D の場合、出力も大文字になります。
HH	2 桁の時間。先頭の 0 は必要ありません。
NN	2 桁の分。先頭の 0 は必要ありません。
SS[*.ss..*]	秒と秒の少数位。
AA	12 時間表記。正午前の時刻は AM で表します。
PP	12 時間表記。正午後の時刻は PM で表します。
JJJ	指定年の日 (1 ～ 366)。

既存のデータベースのタイムスタンプ・フォーマットは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

指定可能な値は、上の表に示す記号から構成されます。各記号は、フォーマットされる日付のデータで置き換えられます。

略式文字については、指定した文字数が数えられます。必要な場合は、A.M. または P.M. のインジケータ (ローカライズ可能) が、指定した文字数に対応するバイト数までトランケートされます。

文字データを表す記号 (*mmm* など) では、出力の文字を次のように制御できます。

- 記号をすべて大文字で入力すると、フォーマットはすべて大文字で表記されます。たとえば、MMM と入力すると、JAN と表記されます。
- 記号をすべて小文字で入力すると、フォーマットはすべて小文字で表記されます。たとえば、mmm と入力すると、jan と表記されます。
- 大文字と小文字を混ぜて入力すると、使用される言語に適切な文字が Ultra Light により選択されます。たとえば、Mmm と入力すると、英語では May、フランス語では mai と表記されます。

数値データを表す記号では、記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

- 記号をすべて大文字または小文字 (MM や mm など) で入力すると、0 埋め込みが行われます。たとえば、yyyy/mm/dd と入力すると、2002/01/01 と表記されます。
- 大文字と小文字を混ぜて入力すると (Mm など)、0 の埋め込みは行われません。たとえば、yyyy/Mm/Dd と入力すると、2002/1/1 と表記されます。

Sybase Central では、データベースを作成するウィザードでタイムスタンプ・フォーマットを設定できます。**[新しいデータベース作成パラメータ]** ページで **[タイムスタンプ形式]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light timestamp_increment 作成パラメータ」 217 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ
- 「タイムスタンプベースのダウンロード」 『Mobile Link - サーバ管理』
- 「Ultra Light での同時実行性」 12 ページ

例

2006 年 5 月 12 日 (金) の 3:30 PM にトランザクションが実行された場合、デフォルトの timestamp_format 構文 *YYYY-MM-DD HH:NN:SS.SSS* では、結果は次のようになります。

2006-05-12 15:30:55.0

Ultra Light timestamp_increment 作成パラメータ

タイムスタンプ値の精度を制限します。データベースにタイムスタンプが挿入されると、この増分に合わせてタイムスタンプはトランケートされます。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o timestamp_increment=***value*;...

指定可能な値

1 ～ 60000000 マイクロ秒

デフォルト

1 マイクロ秒

備考

1000000 マイクロ秒は 1 秒です。

既存のデータベースのタイムスタンプ・インクリメントは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

DEFAULT TIMESTAMP カラムがプライマリ・キーまたはロー識別子として使用されている場合、このインクリメントが役に立ちます。

Sybase Central では、データベースを作成するウィザードでタイムスタンプ・インクリメントを設定できます。**[新しいデータベース作成パラメータ]** ページで **[タイムスタンプ・インクリメント]** オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light timestamp_format 作成パラメータ」 214 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ
- 「タイムスタンプベースのダウンロード」 『Mobile Link - サーバ管理』
- 「Ultra Light での同時実行性」 12 ページ

例

'2000/12/05 10:50:53:700' などの値を格納するには、この作成パラメータを 100000 に設定します。この値に設定すると、秒の部分は小数点以下 1 桁の後でトランケートされます。

Ultra Light utf8_encoding 作成パラメータ

Unicode 用の 8 ビット・マルチバイト・エンコードである UTF-8 フォーマットでデータをエンコードします。

構文

{ **ulcreate** | **ulinit** | **ulload** } **-o utf8_encoding=***value*;...

値

ブール式

デフォルト

0 (データベースの UTF-8 エンコードなし)

備考

UTF-8 文字は 1 ～ 4 バイトで表されます。他のマルチバイト照合の場合、1 または 2 バイトが使用されます。すべてのマルチバイト照合で、2 バイト以上の文字はアルファベットであると見なされます。したがって、二重引用符を使用しないでこれらの文字を識別子で使用できます。

データベースを UTF-8 でエンコードすると、Ultra Light では UTF8BIN 照合を使用して文字がソートされます。UTF8BIN 文字セットは特定の言語に固有ではないので、この文字セットに特定のコード・ページは関連付けられていません。したがって、複数の言語のデータを同じ統合データベースに同期できます。UTF-8 でエンコードされている文字を、Unicode がサポートされていない統合テーブルに同期しようとすると、ユーザ・エラーがレポートされます。

Sybase Central では、データベースを作成するウィザードで UTF-8 エンコーディングを選択できます。**[新しいデータベースの照合と文字セット]**ページで**[はい。データベースの文字セットには UTF8 を使用します。]**オプションを選択します。

クライアント・アプリケーションから、データベース・マネージャ・クラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「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」 39 ページ
- 「Ultra Light 文字セット」 38 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- Ultra Light for Embedded SQL：「ULCreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C++：「CreateDatabase 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「CreateDatabase メソッド」 『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』
- 「作成パラメータの値へのアクセス」 35 ページ

Ultra Light データベース・プロパティ

目次

プロパティ	説明
case	大文字と小文字の区別のステータスを返します。データベースで大文字と小文字が区別される場合は、On を返します。それ以外の場合は、Off を返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light case 作成パラメータ」 189 ページを参照してください。
char_set	データベースの CHAR 文字セットを返します。データベースで使用される文字セットは、データベースの照合順と、データが UTF-8 コード化されているかどうかで決まります。 参照： ● 「Ultra Light utf8_encoding 作成パラメータ」 219 ページ ● 「Ultra Light collation 作成パラメータ」 193 ページ このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。
checksum_level	データベースのチェックサム検証のレベルを返します。値は、0 (チェックサムを追加しない)、1 (重要なページにチェックサムを追加する)、または 2 (すべてのページにチェックサムを追加する) のいずれかです。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light checksum_level 作成パラメータ」 191 ページを参照してください。

プロパティ	説明
collation	データベースの照合順の名前を返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light collation 作成パラメータ」 193 ページを参照してください。
commit_flush_count	コミット・カウント・スレッショルドを設定する commit_flush_count オプションの値を返します。「Ultra Light commit_flush_count オプション [テンポラリ]」 228 ページを参照してください。
commit_flush_timeout	時間間隔スレッショルドを設定する commit_flush_timeout オプションの値を返します。「Ultra Light commit_flush_timeout オプション [テンポラリ]」 230 ページを参照してください。
conn_count	データベースとの接続の数を返します。返される値は動的です。現在存在している接続の数によって変わります。Ultra Light では、最大 14 の同時データベース接続をサポートできます。
date_format	データベースで文字列変換に使用される日付フォーマットを返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light date_format 作成パラメータ」 194 ページを参照してください。
date_order	データベースで文字列変換に使用される日付順を返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light date_order 作成パラメータ」 197 ページを参照してください。

プロパティ	説明
encryption	データベース暗号化のタイプを返します。値は、NONE、SIMPLE、AES、AES_FIPS のいずれかです。 データベースで使用される暗号化は、強力な暗号化 (AES か AES_FIPS) と DBKEY 作成パラメータ、または難読化 (単純暗号化) のどちらが設定されているかによって判断されます。 このプロパティを変更できるのは、最初の値が None (fips も難読化も使用されていない) の場合だけです。その場合は、使用している API の適切な関数またはメソッドを呼び出して、Connection オブジェクトに新しい暗号化キーを指定することで、暗号化キーを変更できます。この場合、値は AES に変更されます。これは、データベースの作成後に fips 作成パラメータを設定することができないためです。次の項を参照してください。 ● Ultra Light for C/C++：「ULChangeEncryptionKey 関数」『Ultra Light - C/C++ プログラミング』 ● Ultra Light.NET：「ChangeEncryptionKey メソッド」『Ultra Light - .NET プログラミング』 ● Ultra Light for M-Business Anywhere：「ChangeEncryptionKey メソッド」『Ultra Light - M-Business Anywhere プログラミング』 ● 「Ultra Light データベースの保護」 42 ページ ● 「Ultra Light fips 作成パラメータ」 199 ページ ● 「Ultra Light obfuscate 作成パラメータ」 205 ページ ● 「Ultra Light DBKEY 接続パラメータ」 244 ページ
file	現在の接続のデータベース・ルート・ファイル名をパスを含めて返します。この値は、DBF 接続パラメータ値で指定されている値です。次の項を参照してください。 ● Ultra Light for C/C++：「GetDatabaseProperty 関数」『Ultra Light - C/C++ プログラミング』 ● Ultra Light.NET：「GetDatabaseProperty メソッド」『Ultra Light - .NET プログラミング』 ● Ultra Light for M-Business Anywhere：「getDatabaseProperty メソッド」『Ultra Light - M-Business Anywhere プログラミング』 ● 「Ultra Light DBF 接続パラメータ」 242 ページ
global_database_id	グローバル・オートインクリメント・カラムに使用される global_database_id オプションの値を返します。「Ultra Light global_database_id オプション」 231 ページを参照してください。

プロパティ	説明
max_hash_size	インデックスのハッシュに使用する、デフォルトの最大バイト数を返します。このプロパティは、インデックス単位で指定できます。「Ultra Light max_hash_size 作成パラメータ」 201 ページを参照してください。
ml_remote_id	Mobile Link 同期のデータベースをユニークに識別する ml_remote_id オプションの値を返します。「Ultra Light ml_remote_id オプション」 232 ページを参照してください。
name	現在の接続におけるデータベースの名前 (またはエイリアス) を返します。返される名前は、DBN 接続パラメータ値と一致します。DBN 接続パラメータを使用しなかった場合、返される名前はパスと拡張子のないデータベース・ファイルになります。 参照： ● 「Ultra Light DBN 接続パラメータ」 245 ページ ● 「Ultra Light DBF 接続パラメータ」 242 ページ
nearest_century	データベースで文字列変換に使用される基準年を返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light nearest_century 作成パラメータ」 203 ページを参照してください。
page_size	データベースのページ・サイズ (バイト単位) を返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light page_size 作成パラメータ」 206 ページを参照してください。
precision	データベースで文字列変換に使用される浮動小数点の精度を返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light precision 作成パラメータ」 208 ページを参照してください。
scale	データベースによる文字列変換中に、計算結果が最大精度にトランケートされるときの小数点以下の最大桁数を返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light scale 作成パラメータ」 210 ページを参照してください。

プロパティ	説明
time_format	データベースで文字列変換に使用される時刻フォーマットを返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light time_format 作成パラメータ」 212 ページを参照してください。
timestamp_format	データベースで文字列変換に使用されるタイムスタンプのフォーマットを返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light timestamp_format 作成パラメータ」 214 ページを参照してください。
timestamp_increment	2 つのユニークなタイムスタンプ間のマイクロ秒単位の最小差を返します。このプロパティの値はデータベースの作成時に設定され、変更するには新しいデータベースを作成する必要があります。「Ultra Light timestamp_increment 作成パラメータ」 217 ページを参照してください。

Ultra Light データベース・プロパティへのアクセス

Ultra Light には、データベース用に取り出すことができる一連のプロパティがあります。

データベース作成パラメータに対応していないデータベース・プロパティの設定を変更できます。

♦ Ultra Light のデータベース・プロパティをブラウズするには、次の手順に従います (Sybase Central の場合)。

1. データベースに接続します。
2. データベースを右クリックし、**[プロパティ]** を選択します。

   データベース・プロパティは、**[データベース・プロパティ]** ウィンドウの **[一般]** タブおよび **[詳細情報]** タブに表示されます。**[詳細情報]** タブでは、データベース・プロパティはプロパティ名のアルファベット順で表示されます。データベース・プロパティを値の順にソートするには、**[値]** カラムをクリックします。
3. データベース・プロパティのブラウズ中にプロパティが変更された場合は、**[再表示]** をクリックします。

♦ データベース・プロパティの値を取得するには、次の手順に従います (C/C++)。

- C/C++ の場合は、GetDatabaseProperty 関数を呼び出す。

  たとえば、conn_count プロパティの値を取得するには、次のように呼び出します。

  ```
  GetDatabaseProperty( ul_database_property_id conn_count )
  ```

  char_set プロパティの値を取得するには、次のように呼び出します。

  ```
  GetDatabaseProperty( ul_database_property_id char_set )
  ```

参照

- Ultra Light for C/C++：「GetDatabaseProperty 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「GetDatabaseProperty メソッド」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「getDatabaseProperty メソッド」『Ultra Light - M-Business Anywhere プログラミング』

Ultra Light データベース・オプション

目次

オプションは、データベースの動作を設定するときに使用します。データベース・オプションは、いつでも設定または変更できます。Ultra Light では、オプションは永続的または一時的です。永続的なオプションはデータベースの sysuldata システム・テーブルに格納されます。一時的なオプション設定は、データベースの実行中のみ保持されます。

オプションの値は、SET OPTION 文を使用して設定します。たとえば、次の文を実行すると、global_database_id オプションが 100 に設定されます。

```
SET OPTION global_database_id=100;
```

特定のデータベース・オプションについて現在の設定を取得するには、対応するデータベース・プロパティの値を問い合わせるか、適切なデータベース・プロパティ取得メソッドを使用します。たとえば、commit_flush_timeout データベース・オプションの現在の設定を取得するには、次の SQL 文を実行します。

```
SELECT DB_PROPERTY ( 'commit_flush_timeout' );
```

Ultra Light commit_flush_count オプション [テンポラリ]

コミット・カウント・スレッショルドを設定します。このスレッショルドに達すると、コミット・フラッシュが実行されます。

指定可能な値

整数

デフォルト

10

備考

トランザクション・カウントを無効にする場合は、0 を使用します。トランザクション・カウントを無効にすると、フラッシュがトリガされた際のコミット数に制限がなくなります。

必要がある場合は、データベースを開始するたびにこのオプションを設定する必要があります。

commit_flush_count と commit_flush_timeout は、どちらも一時的なデータベース・オプションです。これらのオプションは、データベースを起動するたびに設定する必要があります。オプションは、データベースが実行され続けている間は保持されます。これらは、COMMIT_FLUSH=grouped が接続文字列の一部として設定される場合にのみ必要です。

このオプションを設定し、接続文字列にグループ化されるように COMMIT_FLUSH 接続パラメータを設定すると、どちらかのスレッショルドに達するとフラッシュがトリガされます。フラッシュが実行されると、Ultra Light によってカウンタとタイマは**どちらも** 0 に戻されます。その後、どちらか一方がスレッショルドに達するまで、両方ともモニタリングされます。

コミット・フラッシュ・オプションを設定するうえで重要な考慮事項は、コミットされたトランザクションがフラッシュされるまでの遅延によって、データのリカバリ性はどれ程のリスクを受けるかということです。トランザクションがコミットされた後であっても、トランザクションが失われる可能性がわずかながら存在します。コミット後に深刻なハードウェア障害が発生した場合、それがトランザクションが記憶領域にフラッシュされる前ならば、トランザクションはリカバリ時にロールバックされます。遅延を長くするほど、Ultra Light のパフォーマンスは向上します。適切なカウントのスレッショルドの選択は慎重に行ってください。

クライアント・アプリケーションから commit_flush_count オプションを設定するには、オプションを設定するときに、使用しているプログラミング・インタフェースのデータベース・オプション設定機能を使用します。

参照

- 「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」 125 ページ
- 「Ultra Light commit_flush_timeout オプション [テンポラリ]」 230 ページ
- 「Ultra Light COMMIT_FLUSH 接続パラメータ」 239 ページ
- 「Ultra Light SET OPTION 文」 509 ページ
- Ultra Light for C/C++：「SetDatabaseOption 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「ULSetDatabaseOptionString 関数」『Ultra Light - C/C++ プログラミング』

Ultra Light commit_flush_timeout オプション [テンポラリ]

時間間隔スレッショルドを設定します。このスレッショルドに達すると、グループ化されたコミット・フラッシュが実行されます。

指定可能な値

整数 (ミリ秒単位)

デフォルト

10000 ミリ秒

備考

時間スレッショルドを無効にする場合は、0 を使用します。

必要がある場合は、データベースを開始するたびにこのオプションを設定する必要があります。

commit_flush_count と commit_flush_timeout は、どちらも一時的なデータベース・オプションです。これらのオプションは、データベースを起動するたびに設定する必要があります。オプションは、データベースが実行され続けている間は保持されます。これらは、COMMIT_FLUSH=grouped が接続文字列の一部として設定される場合にのみ必要です。

このオプションを commit_flush_timeout オプションとともに設定し、COMMIT_FLUSH 接続パラメータを grouped に設定した場合は、どちらかのスレッショルドに達するとフラッシュがトリガされます。フラッシュが実行されると、Ultra Light によってカウンタとタイマは**どちらも** 0 に戻されます。その後、どちらか一方がスレッショルドに達するまで、両方ともモニタリングされます。

コミット・フラッシュ・オプションを設定するうえで重要な考慮事項は、コミットされたトランザクションがフラッシュされるまでの遅延によって、データのリカバリ性はどれ程のリスクを受けるかということです。トランザクションがコミットされた後であっても、トランザクションが失われる可能性がわずかながら存在します。コミット後に深刻なハードウェア障害が発生した場合、それがトランザクションが記憶領域にフラッシュされる前ならば、トランザクションはリカバリ時にロールバックされます。遅延を長くするほど、Ultra Light のパフォーマンスは向上します。適切なタイムアウトのスレッショルドの選択は慎重に行ってください。

クライアント・アプリケーションから commit_flush_timeout オプションを設定するには、オプションを設定するときに、使用しているプログラミング・インタフェースのデータベース・オプション設定機能を使用します。

参照

- 「Ultra Light commit_flush_count オプション [テンポラリ]」 228 ページ
- 「Ultra Light COMMIT_FLUSH 接続パラメータ」 239 ページ
- 「Ultra Light SET OPTION 文」 509 ページ
- Ultra Light for C/C++：「SetDatabaseOption 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「ULSetDatabaseOptionString 関数」 『Ultra Light - C/C++ プログラミング』

Ultra Light global_database_id オプション

データベースの ID 番号を設定します。

指定可能な値

負でない一意の整数

デフォルト

特定のグローバル・オートインクリメント・カラムのデフォルト値の範囲は、$pn + 1 \sim p(n + 1)$ です。ここで、p はカラムの分割サイズを、n はグローバル・データベース ID 番号を示します。

備考

グローバル ID は、Mobile Link サーバとの同期時にプライマリ・キーの一意性を維持するために GLOBAL AUTOINCREMENT カラムの開始値を設定します。グローバル ID を設定してからデフォルト値を割り当ててください。テーブルにローが追加され、まだ値が設定されていない場合は、Ultra Light によって、global_database_id の値と分割サイズを組み合わせてカラムの値が生成されます。「グローバル・オートインクリメントの使用」 『Mobile Link - サーバ管理』を参照してください。

アプリケーションを配備するときには、Mobile Link サーバとの同期のために、各データベースに異なる ID 番号を割り当てる必要があります。既存のデータベースのグローバル ID はいつでも変更できます。

ulinfo ユーティリティを使用して、このオプションを設定することもできます。

```
ulinfo -g ID ...
```

クライアント・アプリケーションから global_database_id オプションを設定するには、使用しているプログラミング・インタフェースのデータベース ID 設定機能を使用します。

参照

- 「Ultra Light 永続データベース・オプション設定の変更」 233 ページ
- 「Ultra Light 情報ユーティリティ (ulinfo)」 281 ページ
- 「Ultra Light SET OPTION 文」 509 ページ
- 「Ultra Light での GLOBAL AUTOINCREMENT の使用」 135 ページ
- Ultra Light for C/C++ :「SetDatabaseID 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「ULSetDatabaseID 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET :「DatabaseID プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「setDatabaseID メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light データベースのカラムを 3001 から 4000 までオートインクリメントするには、グローバル ID を 3 に設定します。

```
SET OPTION global_database_id="3";
```


Ultra Light ml_remote_id オプション

Mobile Link による同期に使用される Ultra Light のユニークな識別子です。

指定可能な値

Mobile Link による同期時にデータベースをユニークに識別する任意の値

デフォルト

NULL

備考

「リモート ID」は、Ultra Light リモートのユニークな識別子で、Mobile Link による同期に使用されます。Mobile Link リモート ID の初期設定値は NULL です。最初の同期中に、Mobile Link サーバは、リモート ID を GUID に設定します。ただし、リモート ID は、すべての Mobile Link クライアント間でユニークであれば、意味のある任意の文字列にできます。この一意性の要件は常に適用されます。

リモート ID によって、Mobile Link ユーザ名の同期の進行状況が格納されます。ユニークなリモート ID を含めると、ユーザ名はユニークである必要がなくなります。リモート ID は、複数の Mobile Link ユーザが同じ Ultra Light クライアント・データベースを同期する場合に特に便利です。この場合、同期スクリプトでは、ユーザ名だけでなく、リモート ID を参照してください。

ulinfo ユーティリティを使用して、このオプションを設定することもできます。

```
ulinfo -r ID ...
```

クライアント・アプリケーションから ml_remote_id オプションを設定するには、オプションを設定するときに、使用しているプログラミング・インタフェースのデータベース・オプション設定機能を使用します。

参照

- 「Ultra Light 永続データベース・オプション設定の変更」 233 ページ
- 「リモート ID」 『Mobile Link - クライアント管理』
- 「REMOTE パーミッションの付与」 『SQL Remote』
- 「Ultra Light 情報ユーティリティ (ulinfo)」 281 ページ
- 「Ultra Light SET OPTION 文」 509 ページ
- Ultra Light for C/C++：「SetDatabaseOption 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「ULSetDatabaseOptionString 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「SetDatabaseOption メソッド」 『Ultra Light - .NET プログラミング』
- 「Ultra Light クライアント」 131 ページ
- 「Ultra Light ユーザ認証」 『Mobile Link - クライアント管理』
- 「User Name 同期パラメータ」 180 ページ
- および「Password 同期パラメータ」 168 ページ

Ultra Light 永続データベース・オプション設定の変更

Sybase Central から、永続データベース・オプションの設定を表示および変更できます。一時的に、Sybase Central から、Ultra Light データベース・オプションを表示または設定**できない**場合があります。

♦ Ultra Light の永続データベース・オプションをブラウズまたは変更するには、次の手順に従います (Sybase Central の場合)。

1. データベースに接続します。
2. データベースを右クリックし、**[オプション]** を選択します。
3. オプションを設定またはリセットする場合は、**[値]** フィールドに新しい値を入力します。
4. **[すぐに設定]** または **[すぐにリセット]** をクリックして変更をコミットします。

参照

- 「Ultra Light SET OPTION 文」 509 ページ
- 「Ultra Light データベース・プロパティへのアクセス」 226 ページ
- 「DB_PROPERTY 関数 [システム]」 398 ページ
- Ultra Light for C/C++ :「GetDatabaseProperty 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET :「GetDatabaseProperty メソッド」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「getDatabaseProperty メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

Ultra Light 接続パラメータ

目次

Ultra Light CACHE_SIZE 接続パラメータ

データベース・キャッシュのサイズを定義します。

構文

CACHE_SIZE=*number*{ **k** | **m** | **g** }

デフォルト

デフォルトのキャッシュ・サイズは、システムで使用可能なメモリ容量とデータベースの容量によって決まります。

備考

キャッシュ・サイズを指定しなかった場合、またはサイズを 0 に設定した場合は、デフォルトのサイズが使用されます。テスト結果により、パフォーマンスの向上が必要と判断した場合は、キャッシュ・サイズを大きくしてください。

デフォルトのサイズはバイト単位です。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ k、m、g のいずれかを使用してください。

最大キャッシュ・サイズを超える値を指定すると、自動的にプラットフォームのキャッシュ・サイズ上限が設定されます。キャッシュ・サイズをデータベース自体のサイズよりも大きくすると、パフォーマンスは向上しません。また、キャッシュ・サイズが大きいと、その他の使用可能なアプリケーションの数が少なくなることがあります。

パラメータ値に含まれる前後のスペースはすべて無視されます。この接続パラメータの値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- 「Ultra Light の最適化方法」 127 ページ
- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light page_size 作成パラメータ」 206 ページ
- 「Ultra Light RESERVE_SIZE 接続パラメータ」 256 ページ
- 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 C/C++：「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere：「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET：「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の接続文字列フラグメントによって、キャッシュ・サイズが 2 MB に設定されます。

```
"CACHE_SIZE=2m"
```

Ultra Light CE_FILE 接続パラメータ

データベース・ファイルを作成するときに、新しいデータベース・ファイル名を指定します。既存のデータベースへの接続を確立するときに、この接続パラメータがデータベースを識別します。

構文

CE_FILE=*path¥ce-db*

デフォルト

- DBF 接続パラメータ。
- DBF パラメータの値とこのパラメータの値をどちらも指定しなかった場合、デフォルト値は *¥UltraLiteDB¥ulstore.udb* です。

備考

Microsoft Windows Mobile デバイスへの接続にも、他のプラットフォームへの接続にも同じ接続文字列を使用する Ultra Light クライアント・アプリケーションの場合は、CE_FILE 接続パラメータを使用してください。

CE_FILE 接続パラメータは、DBF 接続パラメータより優先されます。Ultra Light 管理ツールから接続する場合、または接続オブジェクトが Windows Mobile データベースに接続する場合は、DBF 接続パラメータを使用します。

CE_FILE の値は、Windows Mobile のファイル名の要件を満たしている必要があります。データベースへの絶対パスを含める場合は、ディレクトリをすべて作成してから、このファイルのパスを設定する必要があります。ディレクトリは自動的には作成されません。

パラメータ値に含まれる前後のスペースはすべて無視されます。この接続パラメータの値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light 接続パラメータでのファイル・パスの指定」 49 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」 52 ページ
- 「Ultra Light DBF 接続パラメータ」 242 ページ
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の例では、新しい接続を作成し、Windows デスクトップおよび Windows Mobile プラットフォームに異なるデータベース・ファイルを指定します。

```
Set Connection = DatabaseMgr.OpenConnection("DBF=d:¥Dbfile.udb;CE_FILE=¥myapp¥MyDB.udb")
```

Ultra Light COMMIT_FLUSH 接続パラメータ

コミットされたトランザクションが、コミット呼び出しの後にいつ記憶領域にフラッシュされるかを規定します。Ultra Light アプリケーションからコミットが呼び出されないと、フラッシュされません。

構文

COMMIT_FLUSH={ immediate | grouped | on_checkpoint }

デフォルト

immediate

備考

この接続パラメータは、ハードウェア障害またはクラッシュの後に、どのトランザクションをリカバリするかを定義します。論理オートコミット操作を単一のリカバリ・ポイントとしてグループ化できます。

これらの操作をグループ化することで、Ultra Light のパフォーマンスを向上させることができますが、データのリカバリ性が低下します。トランザクションがコミットされた後であっても、トランザクションが記憶領域にフラッシュされる前にハードウェア障害またはクラッシュが発生すると、トランザクションが失われる可能性がわずかながらあります。

次のパラメータがサポートされます。

- **immediate** コミットされたトランザクションは、コミット操作が完了する前でも、コミット呼び出しが実行されるとすぐに記憶領域にフラッシュされます。
- **grouped** コミットされたトランザクションは、設定したスレッショルドに達している場合にかぎり、コミット呼び出しがあった時点で記憶領域にフラッシュされます。トランザクション・カウント・スレッショルドを commit_flush_count データベース・オプションを使用して設定するか、時間ベースのスレッショルドを commit_flush_timeout データベース・オプションを使用して設定できます。

  設定された場合、commit_flush_count オプションと commit_flush_timeout オプションはどちらもコミット・フラッシュのトリガとして動作し、先にスレッショルドに達した方がフラッシュをトリガします。フラッシュが実行されると、Ultra Light によってカウンタとタイマはどちらも 0 に戻されます。その後、どちらか一方が再度スレッショルドに達するまで、両方ともモニタリングされます。
- **on_checkpoint** コミットされたトランザクションは、チェックポイント操作の時点で記憶領域にフラッシュされます。次のいずれかを使用してチェックポイントを実行できます。
  - CHECKPOINT 文。チェックポイント・メソッドのない API では、この SQL 文を使用する必要があります。
  - Ultra Light Embedded SQL 用の ULCheckpoint 関数。
  - C++ コンポーネントの接続オブジェクトの Checkpoint メソッド。

参照

- 「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」 125 ページ
- 「Ultra Light commit_flush_count オプション [テンポラリ]」 228 ページ
- 「Ultra Light commit_flush_timeout オプション [テンポラリ]」 230 ページ
- 「Ultra Light CHECKPOINT 文」 478 ページ
- Ultra Light for 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』

Ultra Light CON 接続パラメータ

接続に名前を付け、マルチ接続アプリケーションで簡単に切り替えができるようにします。

構文

CON=*name*

デフォルト

接続名なし

備考

CON 接続パラメータは、アプリケーション全体に対して設定されます。

複数の同時接続を確立し、その接続間で切り替えを行わない場合は、この接続パラメータは使用しないでください。

接続名はデータベース名とは異なります。

パラメータ値に含まれる前後のスペースはすべて無視されます。この接続パラメータの値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light DBN 接続パラメータ」 245 ページ
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の接続文字列フラグメントによって、最初の接続名が MyFirstCon に設定されます。

```
"CON=MyFirstCon"
```


Ultra Light DBF 接続パラメータ

データベース・ファイルを作成するときに、新しいデータベース・ファイル名を指定します。接続を確立するときに、この接続パラメータで、ロードして接続するデータベース・ファイルを指定します。

構文

DBF=*ul-db*

デフォルト

- **デスクトップ・プラットフォームの場合**　NT_FILE または DBF の値を指定しなかった場合は、Ultra Light によってファイル名が *¥UltraLiteDB¥ulstore.udb* に設定されます。
- **Windows Mobile の場合**　CE_FILE または DBF の値を指定しなかった場合は、Ultra Light によってファイル名が *¥UltraLiteDB¥ulstore.udb* に設定されます。
- **Palm OS の場合**　PALM_FILE または DBF の値を指定しなかった場合は、Ultra Light によってファイル名が *ulstore.udb* に設定されます。

備考

単一の接続文字列から、異なるデバイスにある複数のデータベースに接続する場合は、次のパラメータを使用して、プラットフォーム固有の代替パラメータを指定できます。

- CE_FILE
- PALM_FILE
- NT_FILE

プラットフォーム固有の接続パラメータを指定した場合は、そのパラメータが DBF より優先されます。

DBF の値は、プラットフォームのファイル名の要件を満たしている必要があります。

Palm OS　Palm Install Tool では、VFS ボリュームにデータベースを配備できません。代わりに、カード・リーダなどのツールを使用してメディアに直接データベースをコピーする必要があります。

DBF パラメータを使用してファイルを作成する場合は、データベース名に *.pdb* 拡張子を使用してください。デスクトップで Palm OS データベースを管理する場合、このデータベース名が使用されます。管理ツールとユーティリティは、DBF 接続パラメータを使用するデータベースと *.pdb* 拡張子のデータベースに接続できます。ただし、このファイルをデバイスに配備すると、*.pdb* 拡張子は削除されます。したがって、拡張子は使用せず、Palm_FILE パラメータを使用して、アプリケーションをデータベースに接続してください。

たとえば、ファイルに *CustDB.pdb* という名前を付ける場合、管理ツール・デスクトップでは **DBF=CustDB.pdb** を使用します。これにより、データベースがデバイスに配備されたときに、アプリケーションは **PALM_FILE=CustDB** を使用して同じデータベースに接続します。

Windows Mobile　Windows Mobile デバイスに配備する場合は、Ultra Light のユーティリティやウィザードを使用して、接続している Windows Mobile デバイス上の Ultra Light データベースを

管理できます。Windows Mobile デバイス上のファイルを指定するには、絶対パスを指定し、**wce:¥** プレフィクスを使用してください。

パラメータ値に含まれる前後のスペースはすべて無視されます。値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light 接続パラメータでのファイル・パスの指定」 49 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」 52 ページ
- 「Ultra Light DBN 接続パラメータ」 245 ページ
- 「Ultra Light CE_FILE 接続パラメータ」 237 ページ
- 「Ultra Light PALM_FILE 接続パラメータ」 252 ページ
- 「Ultra Light NT_FILE 接続パラメータ」 248 ページ
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

デスクトップのディレクトリ *c:¥mydb* にインストールされているデータベース *MyULdb.udb* に接続するには、次の接続文字列を使用します。

```
"DBF=c:¥mydb¥MyULdb.udb"
```

接続している Windows Mobile デバイスの *UltraLite* フォルダに配備されている同じデータベースに接続するには、次の接続文字列を使用します。

```
"DBF=wce:¥UltraLite¥MyULdb.udb"
```

Ultra Light DBKEY 接続パラメータ

新しいデータベースを作成するときに、データベースの暗号化キーを指定します。既存のデータベースへの接続を確立するときに、この接続パラメータで、データベースの暗号化キーを指定します。

構文

DBKEY=*string*

デフォルト

キーは指定されません。

備考

データベースの正しい暗号化キーを指定しないと、接続が失敗します。

暗号化キーを使用してデータベースを作成すると、データベース・ファイルは AES の 128 ビット・アルゴリズムまたは FIPS アルゴリズムを使用して強力に暗号化されます。強力な暗号化を使用すると、巧妙で容赦ないデータへのアクセス試行に対抗するセキュリティが提供されます。ただし、強力な暗号化を使用すると、パフォーマンスに大きな影響を与える可能性があります。

Palm OS では、ユーザが別のアプリケーションに切り替えると、アプリケーションがシステムによって自動的に終了します。ただし、ユーザが同じアプリケーションに戻るたびにキーを再入力する必要がないように Ultra Light クライアントをプログラミングできます。

パラメータ値に含まれる前後のスペースはすべて無視されます。値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- Ultra Light for C/C++：「Palm OS の暗号化キーの保存、取り出し、クリア」『Ultra Light - C/C++ プログラミング』
- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light データベースの保護」 42 ページ
- 「Ultra Light obfuscate 作成パラメータ」 205 ページ
- 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 C/C++：「OpenConnection 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「データベースへの接続」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「Ultra Light データベースへの接続」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere：「openConnection メソッド」『Ultra Light - M-Business Anywhere 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.NET：「Open メソッド」『Ultra Light - .NET プログラミング』

Ultra Light DBN 接続パラメータ

アプリケーションが複数のデータベースに接続した場合に、データベースを名前で識別します。

構文

DBN=*db-name*

デフォルト

- **Palm OS**　作成者 ID。
- **その他のプラットフォーム**　パスと拡張子のないファイル名 (その他のプラットフォームがある場合)。長さは 16 文字以内です。

備考

Ultra Light では、データベースを開いてからデータベース名が設定されます。データベースが開いたら、クライアント・アプリケーションでファイルではなく名前を使用してデータベースに接続できます。

パラメータ値に含まれる前後のスペースはすべて無視されます。この接続パラメータの値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light DBF 接続パラメータ」 242 ページ
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次のパラメータを使用して Kitchener という実行中の Ultra Light データベースに接続することもできます。

```
DBN=Kitchener;DBF=cities.udb
```

Ultra Light MIRROR_FILE 接続パラメータ

(メイン・データベース・ファイルへの書き込みが発行されると同時に) データベースへのすべての書き込みが発行される、データベース・ミラー・ファイルの名前を指定します。

構文

MIRROR_FILE=path¥mirrorfile-db

デフォルト

なし

備考

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。したがって書き込みのオーバヘッドは 2 倍になり、読み込みのオーバヘッドには影響ありません。データベース・ファイルから破損したページが読み込まれた場合、ミラー・ファイルから読み込むことでページをリカバリできます。

ミラーリングは、Palm VFS を含めて、ファイルベースのストアを使用するすべてのプラットフォームでサポートされますが、Palm のレコード・データベースではサポートされていません。

データベースの起動時に **mirror_file=** オプションが指定された場合、Ultra Light は指定されたファイルを開き、メイン・データベース・ファイルと一致することを確認してから、処理を続行します。ミラー・ファイルがない場合は、この時点でメイン・ファイルをコピーして作成されます。ミラーがデータベース・ファイルではないか破損していると、エラーが報告されます。この場合、ファイルが削除されるか、別のミラーが指定されるまで、データベースは起動しません。ミラーがデータベースと一致しない場合は、SQLE_MIRROR_FILE_MISMATCH が生成され、データベースは起動しません。破損したページがリカバリされると、警告 SQLE_CORRUPT_PAGE_READ_RETRY が生成されます。ミラーリングを使用していないか、ミラー・ファイルも破損している場合は、エラー SQLE_DEVICE_ERROR が生成され、データベースが停止します。

メディア障害から効果的に保護するために、ミラー・ファイルを使用しているときはページ・チェックサムを有効にしてください。ミラーリングを行っているかどうかに関係なく、ページ・チェックサムによって、ページのロード後すぐに Ultra Light でページの破損が検出可能になり、破損したデータの参照を回避できます。チェックサムを有効にするには checksum_level データベース作成オプションを指定します。ミラー・ファイルを使用しているときにチェックサムが有効になっていなかった場合、警告 SQLE_MIRROR_FILE_REQUIRES_CHECKSUMS が生成されます。「Ultra Light checksum_level 作成パラメータ」 191 ページを参照してください。

ミラーは、データベース・ファイルとまったく同じコピーなので、データベースとして直接起動できます。破損したページは ulvalid ユーティリティによって報告されます。「Ultra Light データベース検証ユーティリティ (ulvalid)」 303 ページを参照してください。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light 接続パラメータでのファイル・パスの指定」 49 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」 52 ページ
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の例では、新しい接続を作成し、ミラー・ファイルを作成します。

```
Connection = DatabaseMgr.OpenConnection("DBF=c:¥Dbfile.udb; UID=JDoe;PWD=ULdb;
MIRROR_FILE=c:¥test¥MyMirrorDB.udb")
```

Ultra Light NT_FILE 接続パラメータ

データベース・ファイルを作成するときに、新しいデータベース・ファイル名を指定します。既存のデータベースへの接続を確立するときに、このパラメータがデータベースを識別します。

構文

NT_FILE=path¥nt-db

デフォルト

- DBF 接続パラメータ。
- DBF パラメータの値とこのパラメータの値をどちらも指定しなかった場合、デフォルト値は *¥UltraLiteDB¥ulstore.udb* です。

備考

デスクトップ・データベースへの接続にも、他のプラットフォームにあるデータベースへの接続にも同じ接続文字列を使用する Ultra Light クライアント・アプリケーションの場合は、NT_FILE 接続パラメータを使用してください。

この接続パラメータは DBF パラメータより優先されます。Ultra Light 管理ツールから接続する場合、または接続オブジェクトがデスクトップ・データベースに接続する場合は、DBF 接続パラメータを使用します。

NT_FILE の値は、Windows デスクトップ・プラットフォームのファイル名の要件を満たしている必要があります。

パスは絶対パスまたは相対パスのどちらでもかまいません。ファイル名にディレクトリを含める場合は、ディレクトリをすべて作成してから、このファイルのパスを設定する必要があります。ディレクトリは自動的には作成されません。

パラメータ値に含まれる前後のスペースはすべて無視されます。この接続パラメータの値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light 接続パラメータでのファイル・パスの指定」 49 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」 52 ページ
- 「Ultra Light DBF 接続パラメータ」 242 ページ
- 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 C/C++：「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere：「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET：「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の例では、新しい接続を作成し、デスクトップ、Palm OS、Windows Mobile の各プラットフォームに異なるデータベース・ファイルを指定します。

```
Connection = DatabaseMgr.OpenConnection("UID=JDoe;PWD=ULdb;
NT_FILE=c:¥test¥MyTestDB.udb;CE_FILE=¥database
¥MyCEDB.udb;PALM_FILE=MyPalmDB_MyCreatorID")
```

Ultra Light ORDERED_TABLE_SCAN 接続パラメータ (旧式)

クエリによるローへのアクセスに、プライマリ・キー・インデックスを使用するか、またはデータベース・ページから直接アクセスするかを制御します。

構文

ORDERED_TABLE_SCAN={ yes | no }

デフォルト

no (データベース・ページは直接スキャンされます)

備考

返される結果でローがプライマリ・キー・インデックスの順序になるようにする場合は、ORDER BY 句が最初に含まれるようにクエリを書き直してください。そうしないと、この接続でダイレクト・ページ・スキャンを使用できることによるパフォーマンス向上が実現されなくなります。この接続パラメータを使用するのは、ORDER BY 句を使用するようクエリを書き直すことが現実的でない場合に限定してください。

10.0.0 より前のバージョンの Ultra Light では、Ultra Light のオプティマイザによって他のインデックスが選択されていない場合には、結果を返すのにプライマリ・キー・インデックスが使用されます。

参照

- 「ダイレクト・ページ・スキャンの使用」 123 ページ
- 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 C/C++：「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere：「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET：「Open メソッド」 『Ultra Light - .NET プログラミング』

Ultra Light PALM_ALLOW_BACKUP 接続パラメータ

HotSync を使用したバックアップの動作を制御します。デフォルトでは Ultra Light によって無効になります。

構文

PALM_ALLOW_BACKUP={ yes | no }

デフォルト

no

備考

この接続パラメータは、Palm OS デバイスの Ultra Light の HotSync コンジットを使用したデスクトップ・バックアップ用にサポートされています。

Palm では、HotSync を使用してデータベースをデスクトップにバックアップできます。ほとんどの Ultra Light クライアント・アプリケーションでは、データは同期によってバックアップされるため、デスクトップへの非公式のバックアップを使用する必要はありません。このため、Ultra Light のランタイムでは、Palm のバックアップ動作が無効になります。ただし、同期時に HotSync で Ultra Light データベースをデスクトップにバックアップする必要がある場合は、この接続パラメータを使用して Ultra Light のデフォルトを変更できます。

HotSync を使用したバックアップを有効にしたら、ULDBUtil でバックアップを設定する必要はありません。バックアップを無効にするまで、同期のたびにバックアップが行われます。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light の Palm OS 用データ管理ユーティリティ (ULDBUtil)」 273 ページ
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

Ultra Light PALM_FILE 接続パラメータ

データベース・ファイルを作成するときに、新しいデータベース・ファイル名を指定します。既存のデータベースへの接続を確立するときに、この接続パラメータがデータベースを識別します。

構文 1：レコードベース・ストアの場合

PALM_FILE=*name*

構文 2：ファイルベース・ストアの場合

PALM_FILE=vfs:[*volume-label*:| *volume-ordinal*:]*filename*

デフォルト

- DBF 接続パラメータ。
- このパラメータの値と DBF パラメータの値をどちらも指定しなかった場合、デフォルト値は *ulstore.udb* です。

備考

Palm デバイスへの接続にも、他のプラットフォームへの接続にも同じ接続文字列を使用する Ultra Light クライアント・アプリケーションの場合は、PALM_FILE 接続パラメータを使用してください。

このパラメータは DBF 接続パラメータより優先されます。Ultra Light 管理ツールから接続する場合、または接続オブジェクトが Palm OS データベースに接続する場合は、DBF を使用します。

PALM_FILE の値は、Palm OS プラットフォームのファイル名の要件を満たしている必要があります。

Palm Install Tool では、VFS ボリュームにデータベースを配備できません。代わりに、カード・リーダなどのツールを使用してメディアに直接データベースをコピーする必要があります。

ファイルを作成する場合は、DBF 接続パラメータを使用し、データベース名に *.pdb* 拡張子を使用してください。デスクトップで Palm OS データベースを管理する場合、このデータベース名が使用されます。管理ツールとユーティリティは、DBF 接続パラメータを使用するデータベースと *.pdb* 拡張子のデータベースに接続できます。このファイルをデバイスに配備すると、*.pdb* 拡張子は削除されます。したがって、拡張子は使用せず、Palm_FILE 接続パラメータを使用して、アプリケーションをデータベースに接続してください。

たとえば、ファイルに *CustDB.pdb* という名前を付ける場合、管理ツール・デスクトップでは **DBF=CustDB.pdb** を使用します。データベースがデバイスに配備されると、アプリケーションは **PALM_FILE=CustDB** を使用して同じデータベースに接続します。

VFS ストアでは、常に絶対ファイル・パスを指定します。フル・パス名でディレクトリを指定し、そのディレクトリが見つからない場合は、新しく作成されます。

パラメータ値に含まれる前後のスペースはすべて無視されます。この接続パラメータの値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light 接続パラメータでのファイル・パスの指定」 49 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」 52 ページ
- 「Palm 作成者 ID の登録」 『Ultra Light - C/C++ プログラミング』
- 「Ultra Light DBF 接続パラメータ」 242 ページ
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の例では、新しい接続を作成し、デスクトップ、Palm OS、Windows Mobile の各プラットフォームに異なるデータベース・ファイルを指定します。

```
Connection = DatabaseMgr.OpenConnection("UID=JDoe;PWD=ULdb;
NT_FILE=c:¥test¥MyTestDB.udb;CE_FILE=¥database
¥MyCEDB.udb;PALM_FILE=MyPalmDB_MyCreatorID")
```

Ultra Light PWD 接続パラメータ

認証に使用するユーザ ID のパスワードを定義します。

構文

PWD=*password*

デフォルト

UID と PWD の両方を設定しなかった場合、Ultra Light では **UID=DBA** と **PWD=sql** で接続が確立されます。

備考

パスワードは NULL または空の文字列に設定できますが、最大長 31 文字以内で指定する必要があります。

すべてのデータベース・ユーザにはパスワードがあります。Ultra Light では、最大で 4 つのユーザ ID とパスワードの組み合わせがサポートされています。

この接続パラメータは暗号化されません。ただし、Ultra Light では、パスワードがハッシュされてから保存されるため、パスワードは Sybase Central だけで変更できます。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light ユーザ認証」 53 ページ
- 「ユーザ ID とパスワードの組み合わせの解釈」 53 ページ
- 「Ultra Light UID 接続パラメータ」 259 ページ
- Ultra Light for C/C++ :「ユーザの認証」 『Ultra Light - C/C++ プログラミング』
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「ユーザの認証」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「ユーザの認証」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light.NET :「ユーザの認証」 『Ultra Light - .NET 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の部分的な接続文字列は、ユーザ ID DBA とパスワード sql を指定します。

```
"UID=DBA;PWD=sql"
```

次の部分的な接続文字列は、ユーザ ID DBA と空のパスワードを指定します。

```
"UID=DBA;PWD=""
```

Ultra Light RESERVE_SIZE 接続パラメータ

実際にデータを挿入することなく、Ultra Light データベースが必要とするファイル・システム領域を事前に割り付けます。ファイル・システム領域を予約すると、他のファイルはこの領域を使用できなくなります。

構文

RESERVE_SIZE= *number*{ **k** | **m** | **g** }

デフォルト

0 (予約サイズなし)

備考

0 からデータベースの最大サイズまでの任意の値を指定できます。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ k、m、g のいずれかを使用してください。単位を指定しない場合、デフォルトはバイトです。

データベースをテスト・データを使用して実行することによってデータベース・サイズを確認し、Ultra Light の配備に適した予約サイズを選択してください。

RESERVE_SIZE の値がデータベースのサイズよりも小さい場合、このパラメータは無視されます。

ファイル・システム領域を予約すると、次の理由によりパフォーマンスが多少向上します。

- 段階的なサイズの増加と比較して、断片化レベルが低減される。
- メモリ不足障害を防ぐ。

Ultra Light データベースはデータとメタデータで構成されているので、データベース・サイズが大きくなるのは必要がある場合 (アプリケーションがデータベースを更新する場合) のみです。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」 236 ページ
- 「Ultra Light page_size 作成パラメータ」 206 ページ
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の接続文字列フラグメントによって、予約サイズが 128 KB に設定され、起動時にシステムによってデータベース用にこのシステム領域が予約されるようになります。

```
"RESERVE_SIZE=128K"
```

Ultra Light START 接続パラメータ

Ultra Light エンジンの実行プログラムを起動します。

構文

START=path¥uleng11.exe

備考

現在実行中でないエンジンに接続する場合は、StartLine (START) 接続パラメータだけを指定します。

スペースが含まれているパスには、引用符が必要です。パスにスペースが含まれていない場合、クライアントは SQLE_UNABLE_TO_CONNECT_OR_START を返します。

参照

- 「Ultra Light エンジンユーティリティ (uleng11)」 275 ページ
- 「Ultra Light データ管理コンポーネントの選択」 23 ページ
- 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 C/C++：「OpenConnection 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「データベースへの接続」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「Ultra Light データベースへの接続」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere：「openConnection メソッド」『Ultra Light - M-Business Anywhere 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.NET：「Open メソッド」『Ultra Light - .NET プログラミング』

例

次のコマンドは、*Program Files* ディレクトリの Ultra Light エンジンを起動します。

```
Start=¥"Program Files¥uleng11.exe"
```

または、文字列全体を一重引用符で囲んでこのパスを定義することもできます。

```
Start='"¥Program Files¥uleng11.exe"'
```


Ultra Light UID 接続パラメータ

データベースへの接続に使用するユーザ ID を指定します。値には、データベースで認証されたユーザを指定します。

構文

UID=*user*

デフォルト

UID と PWD を設定しなかった場合、Ultra Light では **UID=DBA** と **PWD=sql** で接続が確立されます。

備考

すべてのデータベース・ユーザにはユーザ ID があります。Ultra Light は、ユーザ ID とパスワードの組み合わせを最大 4 つまでサポートします。

Ultra Light のユーザ ID は、Mobile Link のユーザ名やその他の SQL Anywhere のユーザ ID とは別です。ユーザ ID は一度作成すると変更できません。変更する必要がある場合は、ユーザ ID を削除し、新しい ID を追加してください。

UID は NULL または空の文字列に設定できません。ユーザ ID の最大長は 31 文字です。ユーザ ID の大文字と小文字は区別されません。

パラメータ値に含まれる前後のスペースはすべて無視されます。この接続パラメータの値に、先頭の一重引用符、先頭の二重引用符、またはセミコロンを含めることはできません。

参照

- 「接続文字列を使用した Ultra Light 接続の確立」 51 ページ
- 「Ultra Light ユーザ認証」 53 ページ
- 「ユーザ ID とパスワードの組み合わせの解釈」 53 ページ
- Ultra Light for C/C++ :「ユーザの認証」 『Ultra Light - C/C++ プログラミング』
- 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 C/C++ :「OpenConnection 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「ユーザの認証」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL :「データベースへの接続」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere :「ユーザの認証」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「Ultra Light データベースへの接続」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for M-Business Anywhere :「openConnection メソッド」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light.NET :「ユーザの認証」 『Ultra Light - .NET 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.NET :「Open メソッド」 『Ultra Light - .NET プログラミング』

例

次の接続文字列フラグメントは、データベースのユーザ ID DBA とパスワード sql を指定します。

```
"UID=DBA;PWD=sql"
```


Ultra Light ユーティリティ

目次

Ultra Light には、コマンド・プロンプトからデータベースの基本的な管理作業を行うことができる一連のユーティリティがあります。これらのユーティリティの多くは、SQL Anywhere サーバのユーティリティと機能が似ています。ただし、オプションの使用方法が異なる場合があります。Ultra Light でのオプションの実装方法については、必ず Ultra Light のリファレンスを参照してください。

注意
この章に示すユーティリティのオプションでは、特に明記されていないかぎり、大文字と小文字が区別されます。オプションはここに示すとおり**正確に**入力してください。

サポートされている終了コード

ulcreate、ulload、ulsync、ulunload の各ユーティリティは、処理が正常に終了したかどうかを示す終了コードを返します。0 は処理が成功したことを示します。他の値は処理が失敗したことを示します。

終了コード	ステータス	説明
0	EXIT_OKAY	処理は成功した。
1	EXIT_FAIL	処理は失敗した。
3	EXIT_FILE_ERROR	データベースが見つからない。
4	EXIT_OUT_OF_MEMORY	デバイスの動的メモリが足りない。
6	EXIT_COMMUNICATIONS_FAIL	Ultra Light エンジンとの通信中に通信エラーが発生した。
9	EXIT_UNABLE_TO_CONNECT	指定された UID または PWD が無効であるため、データベースに接続できない。
12	EXIT_BAD_ENCRYPT_KEY	暗号化キーがないか、無効である。
13	EXIT_DB_VER_NEWER	データベースのバージョンに互換性がないことが検出された。データベースを新しいバージョンにアップグレードする必要がある。
255	EXIT_USAGE	コマンド・ライン・オプションが無効である。

Ultra Light 用 Interactive SQL ユーティリティ (dbisql)

SQL コマンドを実行するか、コマンド・ファイルを実行します。

構文

dbisql -c "*connection-string*" **-ul** [*options*] [*dbisql-command* | *command-file*]

オプション	説明
@data	指定された環境変数または設定ファイルからオプションを読み出します。指定された環境変数と設定ファイルが両方とも存在する場合は、環境変数が使用されます。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル難読化ユーティリティを使用して、設定ファイルの内容を難読化できます。「設定ファイルの使用」『SQL Anywhere サーバ - データベース管理』を参照してください。
-c "*connection-string*"	必須。*connection-string* の DBF 接続パラメータまたは file_name 接続パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-d *delimiter*	コマンド・デリミタを指定します。デリミタを囲む引用符は省略可能ですが、コマンド・シェル自体がデリミタを特別な方法で解釈するときは必ず指定します。 コマンド・デリミタは、データベースに格納されている設定に関係なく、その Interactive SQL セッション中のすべての接続に使用されます。
-d1	ユーザが明示的に実行するすべての文をコマンド・ウィンドウ (STDOUT) にエコーします。このエコーによって、SQL スクリプトのデバッグ、または Interactive SQL が長文の SQL スクリプトを処理しているときに有用なフィードバックが提供されます。
-f *filename*	*filename* というファイルを開きますが、実行しません。ファイル名にスペースが含まれる場合はファイル名を引用符で囲みます。 -f オプションが指定されている場合、-c オプションは無視されます。つまり、データベースへの接続は確立されません。 ファイルへのフル・パスを指定なかった場合は、現在のディレクトリへの相対パスと想定されます。そのファイルが存在しない場合、またはファイルではなくディレクトリである場合は、Interactive SQL がエラー・メッセージを出力して停止します。 このオプションは、Interactive SQL をウィンドウ・ベースのアプリケーションで実行している場合のみサポートされます。

オプション	説明
-nogui	コマンド・プロンプト・モードで実行します。*dbisql-command* または *command-file* のいずれかを指定する場合、-nogui が使用されます。
-onerror *behavior*	コマンド・ファイルからデータを読み出し中にエラーが起こった場合の事象を制御します。*behavior* には次のいずれかの値を定義できます。 ● **Stop** Interactive SQL が文の実行を停止します。 ● **Prompt** Interactive SQL は、ユーザに継続したいかどうかを確認するプロンプトを表示します。 ● **Continue** エラーは無視され、Interactive SQL は文の実行を継続します。 ● **Exit** Interactive SQL は終了します。 ● **Notify_Continue** エラーがレポートされますが、ユーザは継続させるために [Enter] を押すか、**[OK]** をクリックするよう要求されます。 ● **Notify_Stop** エラーがレポートされますが、ユーザは実行を停止するために [Enter] を押すか、**[OK]** をクリックするよう要求されます。 ● **Notify_Exit** エラーがレポートされますが、ユーザは Interactive SQL を終了するために [Enter] を押すか、**[OK]** をクリックするよう要求されます。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-ul	デフォルトで Ultra Light データベースに接続します。 Interactive SQL では、デフォルトで SQL Anywhere データベースに接続すると見なされます。-ul オプションを指定すると、デフォルトが Ultra Light への接続になります。デフォルトに設定されているデータベースのタイプに関係なく、SQL Anywhere または Ultra Light のデータベースに接続できます。**[接続]** ウィンドウのドロップダウン・リストからデータベースのタイプを選択して、このアクションを実行します。
-version	Interactive SQL のバージョン番号を表示します。Interactive SQL 内から **[ヘルプ] - [Interactive SQL について]** を選択して、バージョン番号を参照することもできます。
-x	コマンドをスキャンしますが、実行しません。このオプションは、長いコマンド・ファイルの構文エラーをチェックする場合に有用です。
SQL-command | *command-file*	SQL 文または指定された *command-file* を実行します。 *SQL-statement* または *command-file* を指定しなかった場合、Interactive SQL は対話型モードになり、コマンドをコマンド・ウィンドウに入力できます。

備考

Interactive SQL を使うと、SQL コマンドを実行したり、コマンド・ファイルを実行したりできます。また、次の項目に関するフィードバックも提供します。

- 影響を受けたローの数
- 各コマンドに必要な時間
- クエリの実行プラン
- エラー・メッセージ

Ultra Light データベースに接続する場合、SQL Anywhere に固有のメニュー項目がインタフェースに表示されません。たとえば、**[ツール] - [プロシージャ名のルックアップ]** や **[ツール] - [インデックス・コンサルタント]** は表示されません。

Ultra Light では、照合にコード・ページとソート順が含まれます。したがって、コード・ページの番号は、Ultra Light の照合名の一部として表示される番号に対応します。コマンド・プロンプトで ulcreate -l を実行すると、サポートされている照合と対応するコード・ページのリストが表示されます。

INPUT、OUTPUT、または READ 文の ENCODING 句を使用して、ファイルの読み込みまたは書き込みをするときに使用するコード・ページを指定できます。たとえば、英語版の 32 ビット Windows デスクトップ・コンピュータでは、ウィンドウを使用するプログラムは 1252 (ANSI) コード・ページを使用します。297 (IBM France) コード・ページを使用して作成された *status.txt* というファイルを Interactive SQL で読み込むには、次の文を使用します。

```
READ
ENCODING 297
status.txt;
```

終了コードは、0 (成功) または 0 以外の値 (失敗) です。0 以外の終了コードが設定されるのは、(SQL 文またはスクリプト・ファイル名を指定したコマンド・ラインによって) Interactive SQL をバッチ・モードで実行した場合だけです。

コマンド・ライン・モードのとき、Interactive SQL は、処理の成功または失敗をプログラム終了コードを設定することで示します。Windows オペレーティング・システムでは、プログラム終了コードに対して環境変数 ERRORLEVEL が設定されます。

参照

- 「設定ファイルの使用」 『SQL Anywhere サーバ - データベース管理』
- 「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』
- 「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』
- 「Ultra Light 接続パラメータ」 235 ページ
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「サポートされている終了コード」 262 ページ

例

次のコマンドは、Ultra Light の *CustDB.udb* データベースに対してコマンド・ファイル *mycom.sql* を実行します。ユーザ ID とパスワードが定義されていないため、デフォルトのユー

ザ ID **DBA** とパスワード **sql** が使用されます。コマンド・ファイルにエラーがあった場合は、処理は終了します。

```
dbisql -ul -c DBF=CustDB.udb -onerror exit mycom.sql
```


Ultra Light SQL プリプロセッサ・ユーティリティ (sqlpp)

Embedded SQL (ESQL) を含む C/C++ のプログラムをプリプロセッサで処理し、コンパイラを実行する前に、このプログラムに必要なコードを生成できるようにします。次の表では、万全を期すためにすべてのオプションについて説明していますが、Ultra Light に関連するオプションは **-eu** と **-wu** のみです。

構文

sqlpp -u [*options*] *esql-filename* [*output-filename*]

オプション	説明
-d	データ領域サイズを減らし、コード・サイズを増加させるコードを生成します。データ構造体を再利用し、実行時に初期化してから使用します。
-e *flag*	このオプションは、指定した規格に含まれない静的 Embedded SQL をエラーとして通知します。*level* 値は、使用する規格を表します。たとえば、sqlpp -e c03 ... は、コア SQL/2003 規格に含まれない構文を通知します。 *level* として使用される値は次のとおりです。 ● **c03** コア SQL/2003 構文でない構文を通知します。 ● **p03** 上級レベル SQL/2003 構文でない構文を通知します。 ● **c99** コア SQL/1999 構文でない構文を通知します。 ● **p99** 上級レベル SQL/1999 構文でない構文を通知します。 ● **e92** 初級レベル SQL/1992 構文でない構文を通知します。 ● **i92** 中級レベル SQL/1992 構文でない構文を通知します。 ● **f92** 上級レベル SQL/1992 構文でない構文を通知します。 ● **t** 標準ではないホスト変数型を通知します。 ● **u** Ultra Light がサポートしていない構文を通知します。 以前のバージョンの SQL Anywhere と互換性を保つために、e、I、f を指定することもできます。これらはそれぞれ e92、i92、f92 に対応します。
-h *width*	sqlpp によって出力される *.c* ファイル内の分割された行の最大長を *width* に制限します。分割された行が C コンパイラで 1 行として解析されるように、分割された行の末尾には円記号が追加されます。デフォルト値はありません (出力の行はデフォルトで分割されません)。
-k	コンパイルされるプログラムが SQLCODE のユーザ宣言をインクルードすることをプリプロセッサに通知します。

オプション	説明
-n	コード内の適切な場所で #line ディレクティブを使用して、C ファイル内に行番号情報を生成します。 このオプションを使用して、ソースのエラーをレポートし、また *output-filename* ファイルではなく、*esql-filename* ファイル内の行番号にあるソースをデバッグできます。
-o *O/S spec*	Ultra Light には不適用。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-r	Ultra Light には不適用。
-s *string-length*	プリプロセッサが C ファイルに出力する文字列の最大サイズを設定します。この値より長い文字列は、文字のリスト (*'a'*、*'b'*、*'c'* など) を使用して初期化されます。ほとんどの C コンパイラには、処理できる文字列リテラルのサイズに制限があります。このオプションを使用して上限を設定します。デフォルト値は 500 です。
-u	Ultra Light で必須です。Ultra Light データベースに必要な出力を生成します。
-w *level*	SQL 構文に準拠しないものを警告として通知します。*level* 値は、使用する規格を表します。たとえば sqlpp -w c03 ... は、コア SQL/2003 構文に含まれない SQL 構文を通知します。 *level* として使用される値は次のとおりです。 ● **c03** コア SQL/2003 構文でない構文を通知します。 ● **p03** 上級レベル SQL/2003 構文でない構文を通知します。 ● **c99** コア SQL/1999 構文でない構文を通知します。 ● **p99** 上級レベル SQL/1999 構文でない構文を通知します。 ● **e92** 初級レベル SQL/1992 構文でない構文を通知します。 ● **i92** 中級レベル SQL/1992 構文でない構文を通知します。 ● **f92** 上級レベル SQL/1992 構文でない構文を通知します。 ● **t** 標準ではないホスト変数型を通知します。 ● **u** Ultra Light がサポートしていない構文を通知します。 以前のバージョンの SQL Anywhere と互換性を保つために、e、I、f を指定することもできます。これらはそれぞれ e92、i92、f92 に対応します。

オプション	説明
-x	マルチバイト文字列をエスケープ・シーケンスに変更し、コンパイラをパススルーできるようにします。
-z *collation-sequence*	照合順を指定します。

備考

このプリプロセッサは、入力ファイル内の SQL 文を C/C++ に変換します。結果は *output-filename* に書き込みます。Embedded SQL を含むソース・プログラムの拡張子は通常は *sqc* です。デフォルトの *output-filename* は拡張子 *c* が付いた *esql-filename* ベースの名前です。ただし、*esql-filename* に拡張子 *.c* がすでに付いている場合、デフォルトの出力ファイル拡張子は *.cc* になります。

照合順は、プリプロセッサにプログラムのソース・コードで使用されている文字を理解させるために使用します。たとえば、識別子に使用できるアルファベット文字の識別などに使用されます。Ultra Light では、照合にコード・ページとソート順が含まれます。-z を指定しなかった場合は、プリプロセッサが、オペレーティング・システムに基づいて、使用する合理的な照合順を決定しようとします。

コマンド・プロンプトで ulcreate -l を実行すると、サポートされている照合と対応するコード・ページのリストが表示されます。

ヒント
SQL プリプロセッサ (sqlpp) には、Embedded SQL アプリケーション内の静的 SQL 文をコンパイル時に通知する機能があります。この機能は、Ultra Light アプリケーションを開発するときに特に便利です。この機能で、SQL 文に Ultra Light との互換性があることを確認できます。SQL Anywhere アプリケーションと Ultra Light アプリケーションの両方に対する SQL の互換性をテストするには、-e オプションか -w オプションまたはその両方を使用します。SQL FLAGGER の詳細については、「SQL FLAGGER を使用した SQL 準拠のテスト」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

参照

- 「SQL Anywhere Embedded SQL」 『SQL Anywhere サーバ - プログラミング』
- 「Ultra Light 文字セット」 38 ページ

例

次のコマンドは、Ultra Light アプリケーション用に Embedded SQL ファイル *srcfile.sqc* をクワイエット・モードで前処理します。

```
sqlpp -u -q MyEsqlFile.sqc
```

Ultra Light データベース作成ユーティリティ (ulcreate)

定義するプロパティで Ultra Light データベースを作成します。

構文

ulcreate [*options*][*new-database-file*]

オプション	説明
-c "*connection-string*"	*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースを作成します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。ユーザ ID とパスワードを指定した場合は、これらがデータベースへのアクセスに必要となります。 接続文字列のパラメータとしてファイル名を指定しなかった場合は、コマンドの末尾で *new-database-file* で指定したファイルが確認されます。
-g *global-ID*	初期データベース ID を、指定する INTEGER 値に設定します。この初期値と分割サイズが、グローバル・オートインクリメント・カラムがある新しいローに使用されます。アプリケーションを配備するときには、Mobile Link サーバとの同期のために、各データベースに対して異なる ID 番号の範囲を割り当てます。「Ultra Light global_database_id オプション」 231 ページを参照してください。
-l	使用可能な照合順をリストして終了します。
-o [*extended-options*]	Ultra Light データベースの作成パラメータのリストをセミコロンで区切って指定します。「Ultra Light で使用するデータベース作成パラメータの選択」 35 ページを参照してください。
-ol	利用可能なデータベースの作成パラメータをリストして終了します。「Ultra Light で使用するデータベース作成パラメータの選択」 35 ページを参照してください。
-p *creator-ID*	Palm OS で、データベースをレコード・ストアにインストールする場合に必要です。指定する Ultra Light クライアント・アプリケーションの 4 文字の *creator-ID* でデータベースを作成します。VFS ストアにデータベースを配備している場合は、このオプションを使用しないでください。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-t *file*	信頼できるパブリック・ルート証明書を含む指定ファイルをデータベースにロードします。このため、trusted_certificates 同期パラメータをダウンロードする必要がなくなります。この証明書は、サーバ認証に必要です。

オプション	説明
-v	冗長メッセージを表示します。
-y	データベース・ファイルが存在する場合、それを上書きします。
-z *collation-sequence*	使用する照合のラベルを指定します。
new-database-file	指定された名前でファイルを作成します。ユーザ ID (UID) やパスワード (PWD) のような初期データベース・パラメータを設定するための接続文字列を使用していない場合にかぎり、このスタンドアロン・ファイル名を使用してください。設定するスタンドアロン・ファイル名はプラットフォームに適切である必要があります。

備考

データベース・プロパティを設定しなかった場合は、大文字と小文字が区別されず、照合順は現在のロケールに依存するデータベースが作成されます。

データベースの大文字と小文字の区別の設定に関係なく、データベースのパスワードは常に大文字と小文字が区別されます。データベースの大文字と小文字の区別は、作成パラメータ case=respect を使用するかどうかで決まります。

データベース中のすべての文字列比較で使用される照合順です。Ultra Light では、照合にコード・ページとソート順が含まれます。-z を指定しなかった場合は、ulcreate が、デスクトップの現在のロケールに基づいて、使用する合理的な照合順を決定しようとします。

コマンド・プロンプトで ulcreate -l を実行すると、サポートされている照合と対応するコード・ページのリストが表示されます。

UTF-8 エンコーディングを使用すべきかどうかは、デバイスのオペレーティング・システムが判断します。

デスクトップに作成された Palm OS のデータベースは *.pdb* 拡張子で識別できる必要があります。ただし、データベースをデバイスに配備すると、拡張子は削除されます。ファイル名形式の詳細については、「Palm OS」 50 ページを参照してください。

ターゲットが VFS ボリュームの場合、Palm Install Tool を使用して Ultra Light データベースを配備することはできません。代わりに、カード・リーダなどのツールを使用してメディアに直接データベースをコピーする必要があります。

このユーティリティはエラー・コードを返します。0 以外の値は処理に失敗したことを示します。

参照

- 「Ultra Light データベースの作成と設定」 29 ページ
- 「Ultra Light 接続パラメータでのファイル・パスの指定」 49 ページ
- 「Ultra Light 接続パラメータ」 235 ページ
- 「サポートされている終了コード」 262 ページ
- 「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』
- 「Palm 作成者 ID の登録」 『Ultra Light - C/C++ プログラミング』
- 「Ultra Light case 作成パラメータ」 189 ページ
- 「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」 39 ページ
- 「Mobile Link のモデルの概要」 『Mobile Link - クイック・スタート』

例

大文字と小文字を区別せず、Unicode を使用せず、照合順が現在のロケールに依存する、*test.udb* という Ultra Light データベースを作成します。

```
ulcreate test.udb
```

大文字と小文字を区別し、日付フォーマットと日付順が ISO に準拠する *test.udb* というデータベースを作成します。

```
ulcreate -c DBF=test.udb -o case=respect;date_format=YYYY-MM-DD;date_order=YMD
```

暗号化され、暗号化キーが **afvc_1835** の *test.udb* というデータベースを作成します。

```
ulcreate -c "DBF=test.udb;DBKEY=afvc_1835"
```


Ultra Light の Palm OS 用データ管理ユーティリティ (ULDBUtil)

デバイスからデータベースを削除するか、次の同期時にデータベースをバックアップします。

構文

なし

備考

このユーティリティを使用して、次のタスクを実行できます。

- デバイスを複数のユーザで共有している場合にデータベースをデバイスから削除する。ファイルを削除すると、領域を節約し、プライバシーを守ることができます。このデータベースを再インストールするか、このデータベースから新しい空のデータベースを作成できます。
- 次の同期時にデータベースをバックアップする。この機能を使用すると、初期同期を実行してから、データベースをバックアップできます。このデータベースを他のデバイスに配備すると、各デバイスで初期同期を実行する必要がなくなります。

このユーティリティは次のファイルとしてインストールされます。

install-dir¥UltraLite¥Palm¥68k¥ULDBUtil.prc

インストールが完了すると、ULDBUtil はファイル名フィルタを使用します。このフィルタは、Ultra Light の拡張子要件を満たす Ultra Light データベース・ファイルのみを検出します。ファイル名フィルタのリストは、*.bak*、*.dat*、*.dba*、*.htm*、*.html*、*.in*、*.jpeg*、*.jpg*、*.prc*、*.pqa*、*.tda* の拡張子を持つすべてのファイルを除外します。Ultra Light データベースの名前にこのフィルタで使用される拡張子が付いていないことを確認してください。

♦ Palm OS デバイスから Ultra Light アプリケーションのデータを削除するには、次の手順に従います。

1. ULDBUtil に切り替えます。
2. デバイスに拡張カードがある場合は、アプリケーション・データベース・ファイルを削除するメディアを選択します (内部／外部ボリュームまたはレコードベース)。
3. Ultra Light バージョン 11 データベースのリストから、データベースを選択します。
4. Palm デバイスで [削除] をタップして、データを削除します。

次の手順は、HotSync コンジットを使用して配備されたデータベースにのみ適用されます。

♦ Palm OS デバイスからデスクトップにデータベースをバックアップするには、次の手順に従います。

1. ULDBUtil に切り替えます。

2. [バックアップ] オプションを選択して、次の同期時にデータベースをバックアップすることを HotSync に指定します。このオプションは、バックアップのたびに選択する必要があります。このオプションは、カードなどの VFS ボリュームに格納されたデータベースには使用できません。

ヒント
自動バックアップを有効にして、同期のたびにこのオプションを選択しないで済むようにするには、PALM_ALLOW_BACKUP パラメータを使用します。「Ultra Light PALM_ALLOW_BACKUP 接続パラメータ」 251 ページを参照してください。

参照

- 「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)」 278 ページ

例

なし

Ultra Light エンジンユーティリティ (uleng11)

32 ビット Windows デスクトップと Windows Mobile 上のアプリケーションからの同時 Ultra Light データベース接続を管理します。

構文

uleng11

備考

Ultra Light エンジンでは、起動時にメッセージ・ウィンドウが表示されません。

Ultra Light エンジンへの接続には、Ultra Light ランタイムによって使用されるライブラリとは別のライブラリのセットが必要です。TLS が有効化された同期または AES FIPS データベース暗号化も必要な場合は、追加のライブラリが必要となります。

次の項を参照してください。

- 「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』
- 「AES_FIP データベース暗号化を使用した Ultra Light の配備」 61 ページ
- 「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページ

参照

- 「Ultra Light データベースの操作」 73 ページ
- 「Ultra Light データ管理コンポーネントの選択」 23 ページ
- 「Ultra Light エンジン停止ユーティリティ (ulstop)」 276 ページ
- 「Ultra Light START 接続パラメータ」 258 ページ

Ultra Light エンジン停止ユーティリティ (ulstop)

32 ビット Windows デスクトップと Windows Mobile の Ultra Light エンジンを停止します。

構文

ulstop

備考

開発中にエンジンを手動でシャットダウンする場合は ulstop を使用してください。本配備で ulstop を使用する必要は一般的にはありません。

このユーティリティにはオプションはありません。

参照

- 「Ultra Light データ管理コンポーネントの選択」 23 ページ
- 「Ultra Light エンジンユーティリティ (uleng11)」 275 ページ

Ultra Light データベース消去 (ulerase)

Ultra Light データベースを消去します。

構文

ulerase -c "*connection-string*" [*options*] [*dbname*]

オプション	説明
-c "*connection-string*"*keyword=value*	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-ek *key*	暗号化されたデータベースで使用する暗号化キーを指定します。
-ep	暗号化キーの入力を求めるプロンプトを表示するよう指定します。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-v	冗長メッセージを表示します。

備考

データベースにアクセスできる必要があります。ユーザ ID とパスワードの組み合わせを使用して接続が許可される必要があります。そうでない場合、データベースは消去されません。

暗号化されたデータベースでは、接続文字列に指定されるキーが必要です。つまり、-ek <key> または -ep のどちらかを使用します。

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)

コンジットが各データベースの HotSync 同期操作を管理できるように、各 Palm OS データベースをインストールし、HotSync コンジットに登録してください。このユーティリティを使用して、コンジットをアンインストールすることもできます。

構文

ulcond11 -c "*connection-string*" [*options*] *creator-ID*

オプション	説明
creator-ID	必須。コンジットを使用するアプリケーションの作成者 ID を設定します。指定した作成者 ID のコンジットがすでに存在している場合は、新しいコンジットと置き換えられます。
-a	-c オプションで設定した接続文字列に、データベース接続文字列を追加します。このオプションを使用すると、コンジットに複数のデータベースを登録できます。
-c "*connection-string* **sync_profile**=*profile_name*{}	必須。*connection-string* の DBF パラメータで指定するデバイス上の Palm データベースに接続します。定義する接続文字列によって配備された Palm データベースがコンジットに登録されます。接続パラメータはコンジットの設定情報の一部として格納されます。 **sync_profile** には、使用する同期プロファイルとマージ・パラメータを指定します。デフォルトのプロファイルは、**ul_palm_conduit** です。 ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-d *filename*	プラグインの *.dll* ファイルの名前を設定します。 ユーティリティによって、*PluginDLL* という値でレジストリ・キー (*Software¥Sybase¥SQL Anywhere¥version¥Conduit¥creator-ID*) が作成されます。*PluginDLL* の値は、設定した *filename* です。 空の文字列 (**-d""**) を指定すると、レジストリから既存のファイル名がクリアされます。
-n *name*	HotSync マネージャが表示する名前を設定します。デフォルト値は **Conduit** です。 このオプションと -u オプションを同時に使用しないでください。

オプション	説明
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。 ulcond11 を Windows Vista で適切な権限を使用して実行していて、エラーがレポートされなかった場合、ウィンドウはすぐに停止されます。それ以外の場合は、5 秒の遅延が適用されます。
-QQ	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。 ulcond11 を Windows Vista で適切な権限を使用して実行していた場合、ウィンドウはすぐに停止されます。
-u	作成者 ID のコンジットをアンインストールします。-u を指定しなかった場合は、コンジットがインストールされます。-u を指定した場合は、-n を指定しないでください。

備考

ulcond11 はコマンド・ライン・ユーティリティです。Windows Vista では、ulcond11 は昇格されて実行されます。実行プログラムのウィンドウは 5 秒間だけ閉じるのが遅れるので、出力を見ることができます。

HotSync は、各同期がいつ実行され、インストールされている各コンジットが正しく動作したかどうかを記録します。HotSync ログ・ファイルは、Palm デスクトップ・インストール・ディレクトリのサブディレクトリ *User* にあります。

参照

- 「Ultra Light 接続パラメータ」 235 ページ
- 「Ultra Light の Palm OS 用データ管理ユーティリティ (ULDBUtil)」 273 ページ
- 「Ultra Light HotSync コンジットの配備」 65 ページ
- 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 182 ページ
- 「Palm OS の HotSync」 150 ページ
- 「Palm 作成者 ID の登録」 『Ultra Light - C/C++ プログラミング』

例

次のコマンドは、作成者 ID が **Syb2** であるアプリケーションの **CustDB** というコンジットをインストールします。次の例は CustDB サンプル・アプリケーションの設定です。

```
ulcond11 -c "DBF=custdb.udb;UID=DBA;PWD=sql" -n CustDB Syb2
```

次のコマンドは、同じ CustDB サンプル・アプリケーションのコンジットをアンインストールします。

```
ulcond11 -u Syb2
```

次のコマンドは、Palm のデフォルトの同期プロファイルを使用して、palmdb のコンジットをインストールします。

```
ulcond11 ... -c "dbf=palmdb;sync_profile=ul_palm_conduit{stream.host=override_host}" ...
```


Ultra Light 情報ユーティリティ (ulinfo)

Ultra Light データベースに関する情報を表示し、global_id または ml_remote_id データベース・オプションを変更またはクリアします。

構文

ulinfo -c "*connection-string*" [*options*]

オプション	説明
-c "*connection-string*"	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-g *ID*	初期グローバル・データベース ID を、指定する値に設定します。この値は、データベースでグローバル・オートインクリメント・カラムがあるすべての新しいローに使用されます。データベースでは、この基本値を使用して、追加する各ローかカラムまたはその両方に関連付けられる ID がオートインクリメントされます。 アプリケーションを配備するときには、Mobile Link サーバとの同期のために、各データベースに異なる ID 番号を割り当てる必要があります。
-oa	データベースが Ultra Light の旧バージョンで作成されたことをプロセスが識別した場合、そのプロセスをキャンセルします。
-or	読み込み専用モードでデータベースを開きます。Ultra Light によって、元のファイルのコピーが作成されます。このコピーを使用すると、データベースを変更しないでスクリプトをテストできます。コピーされたファイルの変更内容は終了時に破棄されます。 すでに Windows Mobile デバイスに配備されているデータベースにデスクトップから直接接続している場合は、このオプションはサポートされません。
-ou	Ultra Light の旧リリース・バージョンで作成されたデータベースをアップグレードします。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-r *ID*	初期 ml_remote_id を、指定する値に設定します。新しい Ultra Light データベースでは、デフォルトで Mobile Link のリモート ID が NULL に設定されます。このデフォルトを使用することもできます。Ultra Light と dbmlsync の両方で、同期の開始時に Mobile Link のリモート ID がユニークなユーザ ID (UUID) に設定されます。
-rc	Mobile Link のリモート ID を NULL に設定します。

オプション	説明
-v	冗長メッセージを表示します。指定するデータベースの内部構成に加えて現在のデータベース・プロパティを表示します。

備考

Ultra Light データベースを開くときに生成される警告メッセージは、-q オプションを使用しないかぎり、常に表示されます。

クワイエット **(-q)** フラグが使用されていないかぎり、ulinfo ユーティリティを使用すると、既存の同期プロファイルに関する情報と保留中の SQL パススルー・スクリプトの数が表示されます。次に例を示します。

```
Sync profile cv_prosync: "TableOrder=cv_prosync;MobiLinkUid=cv_prosync;ScriptVersion=test;"
Number of available SQL pass-through scripts: 0
```

参照

- 「Ultra Light 接続パラメータ」 235 ページ
- 「Ultra Light global_database_id オプション」 231 ページ
- 「Ultra Light ml_remote_id オプション」 232 ページ

例

すでに同期されている *sample.udb* というファイルの基本的なデータベース内部構成を表示します。

```
ulinfo -c DBF=cv_dbattr.udb
```

```
ulinfo -c DBF=cv_dbattr.udb
Utility Version 11.*suppressed*
Collation: 1252LATIN1
Number of Users: 1
  1. User: 'DBA'
Page size: 4096
Default index maximum hash size: 4
Checksum level: 0
MobiLink Remote ID: not set
Global database ID: 1000
Global autoincrement usage: 0%
Number of tables: 3
Number of columns: 7
Number of publications: 3
Number of tables that will always be uploaded: 0
Number of tables that are never synchronized: 0
Number of primary Keys: 3
Number of foreign Keys: 0
Number of indexes: 0
This database has not yet been synchronized.
Synchronization by publication number:
  1. Publication cv_sync
     Number of rows in next upload: 0
     Last download: 1900-01-01 00:00:00.000
  2. Publication cv_syncPub2
     Number of rows in next upload: 0
     Last download: 1900-01-01 00:00:00.000
  3. Publication cv_syncPub3
     Number of rows in next upload: 0
```



```
    Last download: 1900-01-01 00:00:00.000
Number of available SQL pass-through scripts: 0
ulsync ...
Utility Version 11.*suppressed*
Results of this synchronization:
Succeeded
Download timestamp: 20XX-XX-XX XX:XX:XX.XXXXXX
 Upload OK
 No ignored rows
 No part download remaining
 Authentication value: 1000 (0x3e8)
ulinfo -c dbf=cv_dbattr.udb
Utility Version 11.*suppressed*
Collation: 1252LATIN1
Number of Users: 1
  1. User: 'DBA'
Page size: 4096
Default index maximum hash size: 4
Checksum level: 0
MobiLink Remote ID: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
Global database ID: 1000
Global autoincrement usage: 0%
Number of tables: 3
Number of columns: 7
Number of publications: 3
Number of tables that will always be uploaded: 0
Number of tables that are never synchronized: 0
Number of primary Keys: 3
Number of foreign Keys: 0
Number of indexes: 0
Last synchronization completed successfully
Download occurred: *suppressed*
Upload OK
Upload rows not ignored
No partial downloads
Actual MobiLink Authentication value: 1000
Authentication valid
Synchronization by publication number:
  1. Publication cv_sync
     Number of rows in next upload: 0
     Last download: 20XX-XX-XX XX:XX:XX.XXXXXX
  2. Publication cv_syncPub2
     Number of rows in next upload: 0
     Last download: 1900-01-01 00:00:00.000
  3. Publication cv_syncPub3
     Number of rows in next upload: 0
     Last download: 1900-01-01 00:00:00.000
Number of available SQL pass-through scripts: 0
```

CustDB.udb というファイルのデータベース内部構成を表示し、冗長メッセージを有効にしてデータベース・プロパティを表示します。

```
ulinfo -c DBF=CustDB.udb -v

Utility Version 11.*suppressed*
Database information
Database name: custdb
Disk file 'C:¥Documents and Settings¥All Users¥Shared Documents¥SQL Anywhere 11¥Samples
¥UltraLite¥CustDB¥custdb.udb'
Collation: 1252LATIN1
Number of Users: 1
  1. User: 'DBA'
Page size: 4096
```

```
Default index maximum hash size: 8
Checksum level: 0
MobiLink Remote ID: not set
Global database ID: not set
Encryption: None
Character encoding set: 1252LATIN1
Case sensitive: OFF
Date format: YYYY-MM-DD
Date order: YMD
Nearest century: 50
Numeric precision: 30
Numeric scale: 6
Time format: HH:NN:SS.SSS
Timestamp format: YYYY-MM-DD HH:NN:SS.SSS
Timestamp increment: 1
Number of tables: 6
Number of columns: 16
Number of publications: 1
Number of tables that will always be uploaded: 0
Number of tables that are never synchronized: 1
Number of primary Keys: 6
Number of foreign Keys: 2
Number of indexes: 3
This database has not yet been synchronized.
Synchronization by publication number:
  1. Publication test
     Number of rows in next upload: 0
     Last download: 1900-01-01 00:00:00.000
Done.
```

sample.udb というファイルの ml_remote_id を NULL に設定します。

```
ulinfo -c DBF=sample.udb -rc
```


Ultra Light データベース初期化ユーティリティ (ulinit)

既存の SQL Anywhere データベースから Ultra Light データベースを作成します。

構文

ulinit -a "*SAconnection-string*" **-c** "*ULconnection-string*" **-n** *pubname* [*options*]

オプション	説明
-a *"SAconnection-string"*	必須。*SAconnection-string* で指定する SQL Anywhere リファレンス・データベースに接続します。
-c *"ULconnection-string"*	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-d	新しい Ultra Light データベースの各テーブルに、SQL Anywhere データベースの対応するテーブルからデータをコピーします。デフォルトでは、これ以降の同期時にこのデータがアップロードされません。次の同期時にデータをアップロードするには、**-I** と同時に **-d** を使用します。
-e *table, ...*	指定した *table* を除外します。指定したテーブルは、Ultra Light データベースに作成されません。複数のテーブルをカンマで区切って指定できます。次に例を示します。 -e mydbtable1,mydbtable5
-I	-d と同時に使用します。挿入されたローを次の同期時にアップロードします。デフォルトでは、このユーティリティによって挿入されたローは、同期時にアップロードされません。
-l *logfile*	実行時に DDL データベース・スキーマ作成の SQL 文のログを *logfile* に取ります。
-n *pubname*	必須。テーブルを Ultra Light データベース・スキーマに追加します。 *pubname* は、リファレンス・データベース内のパブリケーションを指定します。パブリケーション内のテーブルは、Ultra Light データベースに追加されます。 複数のパブリケーションからのテーブルを Ultra Light データベースに追加するには、オプションを複数回指定します。リファレンス・データベース内のすべてのテーブルを Ultra Light データベースに追加するには、**-n*** を指定します。
-o [*extended-options*]	Ultra Light データベースの作成パラメータのリストをセミコロンで区切って指定します。「Ultra Light で使用するデータベース作成パラメータの選択」 35 ページを参照してください。

オプション	説明
-p *creator-ID*	Palm OS で、データベースをレコード・ストアにインストールする場合に必要です。指定する Ultra Light クライアント・アプリケーションの 4 文字の *creator-ID* でデータベースを作成します。VFS ストアにデータベースを配備している場合は、このオプションを使用しないでください。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-s *pubname*	リファレンス・データベースの *pubname* と同じ定義を持つパブリケーションを Ultra Light データベースに作成します。パブリケーションは、同期の設定に使用されます。複数の同期パブリケーションを指定するには、複数の -s オプションを指定します。 このパブリケーション内のテーブルが -n オプションでリストされたパブリケーションに含まれている必要があります。 -s を指定しないと、Ultra Light リモートにはパブリケーションが指定されません。 Mobile Link との同期用のパブリケーションを作成する方法については、「Ultra Light のパブリケーション」 141 ページを参照してください。
-t *file*	信頼できるルート証明書を含むファイルを指定します。この証明書は、サーバ認証に必要です。
-w	警告を表示しません。

備考

SQL Anywhere リファレンス・データベースは、次のソースとして機能します。

- データベース設定 (照合順な ど)
- テーブル定義
- 同期パブリケーション

これらを組み合わせて、Ultra Light スキーマ (新しい Ultra Light データベースの構造を定義する情報) を作成できます。ただし、作成した新しいデータベースは、最初は空です。

SQL Anywhere リファレンス・データベースを使用しないで Ultra Light データベースを作成するには、次のいずれかの方法を使用します。

- SQL Anywhere 以外の RDBMS から Ultra Light データベースを初期化するには、Sybase Central の**同期モデル作成ウィザード**を使用します。このウィザードを実行すると、スキーマ情報を取得するために統合データベースに接続するように求められます。

- リファレンス・データベースから独立して設定できる空の Ultra Light データベースを作成するには、ulcreate ユーティリティまたは Ultra Light の**データベースの作成**ウィザードを使用します。

Ultra Light では、リファレンス・データベースで定義されていた照合順の名前が使用されます。ただし、*extended-options* リストで utf8_encoding プロパティを設定することで、データベースを UTF-8 を使用してエンコードできます。

コマンド・プロンプトで ulcreate -l を実行すると、サポートされている照合と対応するコード・ページのリストが表示されます。選択した照合順が Ultra Light でサポートされていない場合は、サポートされている照合順に変更してください。たとえば、リファレンス・データベースの照合が UCA 照合の場合は、次の手順に従います。

1. リファレンス・データベースをアンロードし、別の照合を使用して再ロードします。
2. この新しいバージョンのデータベースで ulinit を実行します。

デスクトップに作成された Palm のデータベースは *.pdb* 拡張子で識別できる必要があります。ただし、データベースをデバイスに配備すると、拡張子は削除されます。ファイル名形式の詳細については、「Palm OS」 50 ページを参照してください。

ターゲットが VFS ボリュームの場合、Palm Install Tool を使用して Ultra Light データベースを配備することはできません。代わりに、カード・リーダなどのツールを使用してメディアに直接データベースをコピーする必要があります。

参照

- 「SQL Anywhere リファレンス・データベースからの Ultra Light データベースの作成」 31 ページ
- 「Mobile Link のモデルの概要」 『Mobile Link - クイック・スタート』
- 「Ultra Light データベース作成ユーティリティ (ulcreate)」 270 ページ
- 「Ultra Light 接続パラメータ」 235 ページ

例

TestPublication で定義されているテーブルを含む *customer.udb* というファイルを作成します。

```
ulinit -a "DSN=dbdsn;UID=JimmyB;PWD=secret" -c DBF=customer.udb -n TestPublication
```

2 つの異なるパブリケーションを含む *customer.udb* というファイルを作成します。Pub1 には、優先的に同期するデータのサブセットが含まれ、Pub2 には残りのデータが含まれます。

```
ulinit -a "DSN=dbdsn;UID=JimmyB;PWD=secret" -c DBF=customer.udb -n Pub1 -n Pub2 -s Pub1 -s Pub2
```

登録された作成者 ID を使用して *customer.udb* という Palm OS 用のファイルを作成します。

```
ulinit -a "DSN=dbdsn;UID=JimmyB;PWD=secret" -c DBF=customer.udb.pdb -n TutCustomersPub -p
creator-id
```


Ultra Light データベースへの XML のロード・ユーティリティ (ulload)

XML ファイルから新しいデータベースまたは既存のデータベースにデータをロードします。

構文

ulload -c "*connection-string*" [*options*] *xml-file*

オプション	説明
-a	データとスキーマの定義を既存のデータベースに追加します。 Palm OS 用の既存のレコードベースのデータベース (拡張子が *.pdb* のデータベース) にデータを追加する場合は、-p オプションを使用しないでください。
-c "*connection-string*"	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-d	データだけをロードし、XML ファイル入力内のスキーマのメタデータは無視します。
-f *directory*	ロードする追加データを含むファイルが格納されているディレクトリを設定します。ulunload の **-f** オプションの説明 (「Ultra Light データベースのアンロード・ユーティリティ (ulunload)」 298 ページ) を参照してください。
-g *ID*	初期データベース ID を、指定する INTEGER 値に設定します。この値は、データベースでグローバル・オートインクリメント・カラムがあるすべての新しいローに使用されます。データベースでは、この基本値を使用して、追加する各ローかカラムまたはその両方に関連付けられる ID がオートインクリメントされます。 アプリケーションを配備するときには、Mobile Link サーバとの同期のために、各データベースに異なる ID 番号を割り当てる必要があります。
-I	挿入されたローを次の同期時にアップロードします。デフォルトでは、このユーティリティによって挿入されたローは、同期時にアップロードされません。
-n	スキーマのメタデータだけをロードし、XML ファイル入力内のデータは無視します。
-o [*extended-options*]	Ultra Light データベースの作成パラメータのリストをセミコロンで区切って指定します。「Ultra Light で使用するデータベース作成パラメータの選択」 35 ページを参照してください。

オプション	説明
-oa	データベースが Ultra Light の旧バージョンで作成されたことをプロセスが識別した場合、そのプロセスをキャンセルします。
-ol	利用可能なデータベースの作成パラメータをリストして終了します。「Ultra Light で使用するデータベース作成パラメータの選択」 35 ページを参照してください。
-onerror *behavior*	XML ファイルからデータを読み出し中にエラーが起こった場合の事象を制御します。*behavior* には次のいずれかの値を指定できます。 ● **continue** エラーは無視され、XML のロードが継続されます。 ● **prompt** 継続するかどうかを確認するプロンプトが表示されます。 ● **quit** XML のロードが停止し、エラーで終了します。この動作はデフォルトの動作です。 ● **exit** ulload が終了します。
-or	読み込み専用モードでデータベースを開きます。Ultra Light によって、元のファイルのコピーが作成され、このコピーを使用して、データベースを変更しないでスクリプトがテストされます。コピーされたファイルの変更内容は終了時に破棄されます。 すでに Windows Mobile デバイスに配備されているデータベースにデスクトップから直接接続している場合は、このオプションはサポートされません。
-ou	Ultra Light の旧リリース・バージョンで作成されたデータベースをアップグレードします。
-p *creator-ID*	Palm OS で、データベースをレコード・ストアにインストールする場合に必要です。指定する Ultra Light クライアント・アプリケーションの 4 文字の *creator-ID* でデータベースを作成します。VFS ストアにデータベースを配備している場合は、このオプションを使用しないでください。 Palm OS 用の既存のレコードベースのデータベース (拡張子が *.pdb* のデータベース) にデータを追加する場合は、-a オプションと同時にこのオプションを使用しないでください。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-s *file*	データベースのロードに使用された SQL 文を、指定する *file* に記録します。
-t *file*	信頼できるルート証明書を含むファイルを指定します。この証明書は、サーバ認証に必要です。

オプション	説明
-v	冗長メッセージを表示します。
-y	確認メッセージを表示しないでデータベース・ファイルを上書きします。このオプションは、ulload を使用して新しいデータベースを作成する場合にのみ適用されます。
xml-file	データのロード元の XML ファイルの名前を指定します。

備考

ulload ユーティリティは、ulunload、ulunloadold、または ulxml (Ultra Light バージョン 8 と 9) で生成された入力 XML ファイルを取得します。ulunload とこのユーティリティを同時に使用すると、データベースを再構築できます。データベースを再構築するもう 1 つの手段としては、ulunload を使用して SQL 文を生成し、DBISQL を使用して新しいデータベースに読み込む方法があります。

XML ファイルには、スキーマのメタデータかデータベース・データのメタデータまたはその両方が含まれている可能性があります。**-d** を使用すると、スキーマのメタデータは無視され、データのみが *.udb* ファイルに追加されます。**-n** を使用すると、データとメタデータが無視され、スキーマのみが *.udb* ファイルに追加されます。

コマンド・ラインでオプションを設定するか、証明書を指定すると、ulload で処理される *xml-file* 内の設定より優先されます。

ulload ユーティリティは、XML の読み込み時にデータベースに同期プロファイルをリストアします。

このユーティリティはエラー・コードを返します。0 以外の値は処理に失敗したことを示します。

デスクトップに作成された Palm のデータベースは *.pdb* 拡張子で識別できる必要があります。ただし、データベースをデバイスに配備すると、拡張子は削除されます。ファイル名形式の詳細については、「Palm OS」 50 ページを参照してください。

ターゲットが VFS ボリュームの場合、Palm Install Tool を使用して Ultra Light データベースを配備することはできません。代わりに、カード・リーダなどのツールを使用してメディアに直接データベースをコピーする必要があります。

参照

- 「Palm 作成者 ID の登録」 『Ultra Light - C/C++ プログラミング』
- 「Ultra Light 接続パラメータ」 235 ページ
- 「Ultra Light データベースのアンロード・ユーティリティ (ulunload)」 298 ページ
- 「サポートされている終了コード」 262 ページ
- 「Ultra Light global_database_id オプション」 231 ページ

例

新しい Ultra Light データベース・ファイル *sample.udb* を作成し、*sample.xml* 内のデータをロードします。


```
ulload -c DBF=sample.udb sample.xml
```

sample.xml から既存のデータベース *sample.udb* にデータをロードし、エラーが発生した場合はプロンプトを表示します。

```
ulload -d -c DBF=sample.udb -onerror prompt sample.xml
```

test_data.xml というファイルから、Ultra Light によって作成される *sample.udb* というデータベースのコピーに XML をロードします。変更内容は終了時に破棄します。このファイルで、XML データにエラーがないかどうかを確認し、エラーがあった場合は修正できます。データが正常にロードされたら、-or オプションを指定しないでコマンドを実行し、XML の更新内容を保持できます。

```
ulload -or -c DBF=sample.udb -a test_data.xml
```

Ultra Light 同期ユーティリティ (ulsync)

Ultra Light データベースを Mobile Link サーバと同期させます。このツールを使用して、アプリケーション開発中に同期をテストできます。

構文

ulsync -c "*connection-string*" [*options*] [*synchronization parameters { REPLACE | MERGE } profile-string*]

オプション	説明
-c "*connection-string*"	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-oa	データベースが Ultra Light の旧バージョンで作成されたことをプロセスが識別した場合、そのプロセスをキャンセルします。
-or	読み込み専用モードでデータベースを同期させます。Ultra Light によって、元のファイルのコピーが作成され、このコピーを使用して、データベースを変更しないでスクリプトがテストされます。コピーされたファイルの変更内容は終了時に破棄されます。 すでに Windows Mobile デバイスに配備されているデータベースにデスクトップから直接接続している場合は、このパラメータはサポートされません。
-ou	Ultra Light の旧リリース・バージョンで作成されたデータベースをアップグレードします。
-p	指定した同期プロファイルを使用して同期します。次の構文と同義です。 synchronize profileName merge 'syncOptions' ここでは、同期オプションは後続の ulsync オプションから取得されます。次に例を示します。 ulsync -p profileName "MobiLinkUid=ml;ScriptVersion=Version001...syncOptions" 「同期プロファイル・オプション」 295 ページを参照してください。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-r	前回の同期の結果を表示し、終了します。
-s	同期後に使用可能な SQL パススルー・スクリプトを実行します。

オプション	説明
-v	同期の進行状況のメッセージを表示します。C++ API または SQL の同期プロファイル文を使用して同期の進捗状況を表示するかどうかも指定します。「Ultra Light CREATE SYNCHRONIZATION PROFILE 文」 484 ページを参照してください。

備考

バージョン 10 以前で有効であったオプション **-a authenticate-parameters**、**-e sync-parms**、**-k stream-type**、**-n** (同期なし)、**-x protocol options** は、サポートされていません。**-e <keyword>=<value>** は現在、同期パラメータの文字列に含まれており、**-k** と **-x** は、**Stream=<stream{<stream-parms>}** 同期パラメータの文字列に収納されています。

用途によって、ulsync は次の SQL 文のいずれかと同義と考えることができます。

```
ulsync -p <profile> "<parms>"
```

この文は次の文と同義です。

```
SYNCHRONIZE PROFILE <profile> MERGE
<parms>
```

および

```
ulsync "<parms>"
```

この文は次の文と同義です。

```
SYNCHRONIZE USING <parms>
```

安全な同期のために、Ultra Light アプリケーションはパブリック証明書にアクセスする必要があります。次の方法で証明書を参照できます。

- ulinit、ulload、ulcreate で **-t** *file* オプションを使用して Ultra Light データベースの作成時に証明書情報を組み込む。
- **trusted_certificates=***file* ストリーム・オプションを使用して同期時に外部の証明書ファイルを参照する。

このユーティリティはエラー・コードを返します。0 以外の値は処理に失敗したことを示します。

参照

- 「同期プロファイル・オプション」 295 ページ
- 「Ultra Light 接続パラメータ」 235 ページ
- 「Ultra Light クライアント」 131 ページ
- 「サポートされている終了コード」 262 ページ
- 「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』
- 「TLS が有効化された同期を使用した Ultra Light の配備」 62 ページ

例

次のコマンドは、**remoteA** という Mobile Link ユーザの *myuldb.udb* というデータベース・ファイルを同期させます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;Stream=http;ScriptVersion=2"
```

次のコマンドは、*myuldb.udb* というデータベース・ファイルを HTTPS 経由で *c:¥certs¥rsa.crt* 証明書を使用して同期させます。データベースの作成時に信頼できる証明書ファイルがデータベースに追加されなかったため、**trusted_certificates=***file* オプションを使用する必要があります。さらに、Mobile Link ユーザ名は **remoteB** です。

```
ulsync -c DBF=myuldb.udb "Stream=https{trusted_certificates=c:¥certs¥rsa.crt};
MobiLinkUid=remoteB;ScriptVersion=2;UploadOnly=ON"
```

次のコマンドは、*synced.udb* というデータベース・ファイルについて、前回の同期の結果を表示します。

```
ulsync -r -c dbf=synced.udb
```

前回の同期の結果は、次のようにリストされます。

```
SQL Anywhere UltraLite Database Synchronize Utility Version XX.X
   Results of last synchronization:
   Succeeded
      Download timestamp: 2006-07-25 16:39:36.708000
      Upload OK
      No ignored rows
      Partial download retained
      Authentication value: 1000 (0x3e8)
```

次の例は、CustDB データベースをユーザ名 50 で TCP/IP 接続によってポート 2439 で同期させる場合に使用するコマンド・ラインを示します。

```
ulsync -c "dbf=C:¥Documents and Settings¥All Users¥Documents¥SQL Anywhere 11¥Samples¥UltraLite
¥SyncEncrypt¥custdb.udb"
MobiLinkUid=50;ScriptVersion=custdb 11.0;Stream=tcpip{port=2439}
```

同期プロファイル・オプション

同期プロファイル・オプションは、ulsync ユーティリティのコマンド・ラインで、使用する他のすべてのコマンド・ライン・オプションを定義した後に指定します。キーワードは大文字と小文字が区別されません。

同期プロファイル・オプション	有効な値	説明
AllowDownloadDupRows	ブール式	このオプションを指定すると、プライマリ・キーの同じローが複数ダウンロードされたときにエラーが発生しません。このオプションによって、同期を失敗させることなく、不整合なデータを同期できます。デフォルト値は "no" です。「Additional Parameters 同期パラメータ」 159 ページを参照してください。
AuthParms	文字列 (カンマ区切り)	Mobile Link サーバに送信される認証パラメータのリストを指定します。認証パラメータを使用すると、Mobile Link スクリプトでカスタム認証を実行できます。「Authentication Parameters 同期パラメータ」 161 ページを参照してください。
CheckpointStore	ブール式	同期中にデータベースのチェックポイントを追加して、同期プロセス中のデータベースの拡張を制限します。「Additional Parameters 同期パラメータ」 159 ページを参照してください。
ContinueDownload	ブール式	以前に失敗したダウンロードを再起動します。ダウンロードを続行すると、失敗した同期でダウンロードするように選択されていた変更のみを受信します。デフォルトでは、Ultra Light はダウンロードを続行しません。「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。
DisableConcurrency	ブール式	この同期中は他のスレッドからデータベースへのアクセスを禁止します。「Additional Parameters 同期パラメータ」 159 ページを参照してください。
DownloadOnly	ブール式	ダウンロード専用同期を実行します。「Download Only 同期パラメータ」 163 ページを参照してください。
KeepPartialDownload	ブール式	接続エラーが発生した場合に、部分的なダウンロードを維持するかどうかを制御します。デフォルトでは、Ultra Light は、部分的にダウンロードされた変更をロールバックしません。「Keep Partial Download 同期パラメータ」 165 ページを参照してください。

同期プロファイル・オプション	有効な値	説明
MobiLinkPwd	文字列	ユーザ名に対する既存の Mobile Link パスワードを指定します。「MobiLinkPwd (mp) 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。
MobiLinkUid	文字列	Mobile Link ユーザ名を指定します。「-u オプション」 『Mobile Link - クライアント管理』を参照してください。「-mn オプション」 『Mobile Link - クライアント管理』を参照してください。
NewMobiLinkPwd	文字列	Mobile Link ユーザの新しいパスワードを指定します。このオプションは、既存のパスワードを変更する場合に使用します。「-mn オプション」 『Mobile Link - クライアント管理』を参照してください。
Ping	ブール式	サーバとの通信の確認のみを行います。同期は実行しません。「Ping 同期パラメータ」 169 ページを参照してください。
Publications	文字列 (カンマ区切り)	同期させるパブリケーションを指定します。パブリケーションによって、同期に関与するリモート上のテーブルが決定されます。このパラメータが空白 (デフォルト) の場合は、すべてのテーブルが同期されます。このパラメータがアスタリスク (*) の場合は、すべてのパブリケーションが同期されます。「Ultra Light のパブリケーション」 141 ページを参照してください。
ScriptVersion	文字列	Mobile Link スクリプトのバージョンを指定します。スクリプト・バージョンは、同期中に統合データベースの Mobile Link によって実行されるスクリプトを決定します。スクリプト・バージョンを指定しない場合、「デフォルト」が使用されます。「ScriptVersion (sv) 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。
SendColumnNames	文字列	同期時にカラム名がアップロード・ファイルの一部として Mobile Link サーバに送信されるように指定します。デフォルトでは、カラム名は送信されません。「Send Column Names 同期パラメータ」 172 ページを参照してください。
SendDownloadACK	ブール式	クライアントからサーバにダウンロード確認が送信されるように指定します。デフォルトでは、Mobile Link サーバはダウンロード確認を提供しません。「Send Download Acknowledgement 同期パラメータ」 173 ページを参照してください。

同期プロファイル・オプション	有効な値	説明
Stream	文字列 (サブリスト付き)	Mobile Link のネットワーク同期プロトコルを指定します。「Stream Type 同期パラメータ」 175 ページを参照してください。
TableOrder	文字列 (カンマ区切り)	アップロードでのテーブルの順序を指定します。デフォルトでは、Ultra Light は、外部キーの関係に基づいて順序を選択します。「Additional Parameters 同期パラメータ」 159 ページを参照してください。
UploadOnly	文字列	同期にはアップロードのみが含まれ、ダウンロードは発生されないよう指定します。「Upload Only 同期パラメータ」 178 ページを参照してください。

ブール値に指定できるのは、Yes/No、1/0、True/False、On/Off です。いずれのブールでも、デフォルトは "No" です。他のすべての値では、デフォルトは単純に未指定です。

参照

- 「Ultra Light ALTER SYNCHRONIZATION PROFILE 文」 472 ページ
- 「Ultra Light DROP SYNCHRONIZATION PROFILE 文」 497 ページ
- 「Ultra Light SYNCHRONIZE 文」 512 ページ
- 「Ultra Light 作成パラメータ」 187 ページ

Ultra Light データベースのアンロード・ユーティリティ (ulunload)

使用するオプションによって、次のいずれかをアンロードします。

- Ultra Light データベース全体を XML または SQL にアンロードする。
- Ultra Light データのすべてまたは一部だけを XML または SQL にアンロードする。

構文

ulunload -c "*connection-string*" [*options*] *output-file*

オプション	説明
-b *max-size*	XML ファイルに格納するカラム・データの最大サイズを設定します。デフォルトは 10 KB です。すべてのデータを XML ファイルに保存するには (最大サイズなし)、**-b -1** を使用します。
-c "*connection-string*"	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-d	データベースのデータのみを出力ファイルにアンロードします。スキーマ情報はアンロードしません。
-e *table,...*	データベースのアンロード時に指定した *table* を除外します。複数のテーブルをカンマで区切って指定できます。次に例を示します。 -e mydbtable1,mydbtable5
-f *directory*	-b で指定した最大サイズよりも大きいデータを格納するディレクトリを設定します。デフォルトは、出力ファイルと同じディレクトリです。
-n	スキーマだけをアンロードし、データベース内のデータは無視します。
-oa	データベースが Ultra Light の旧バージョンで作成されたことをプロセスが識別した場合、そのプロセスをキャンセルします。
-or	読み込み専用モードでデータベースを開きます。Ultra Light によって、元のファイルのコピーが作成され、これがアンロードに使用されます。これにより、以前のバージョンのソフトウェアで作成されたデータベースがアップグレードされることはありません。 すでに Windows Mobile デバイスに配備されているデータベースにデスクトップから直接接続している場合は、このパラメータはサポートされません。
-ou	Ultra Light の旧リリース・バージョンで作成されたデータベースをアップグレードします。

オプション	説明
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-s	SQL Anywhere 互換の SQL 文としてアンロードします。SQL ファイルの出力は、DBISQL. を使用して Ultra Light または SQL Anywhere で読み込むことができます。
-t *table,...*	指定した *table* 内のデータだけをアンロードします。複数のテーブルをカンマで区切って指定できます。次に例を示します。 -t mydbtable2,mydbtable6
-v	冗長メッセージを表示します。
-x *owner*	テーブルが特定のユーザ ID の所有になるように出力します。このオプションは、-s オプションと同時に使用できます。
-y	確認メッセージを表示しないで *output-file* を上書きします。
output-file	必須。データベースのアンロード先のファイルの名前を設定します。-s オプションを使用すると、データベースは SQL 文としてアンロードされます。-s オプションを使用しなかった場合は、データベースは XML としてアンロードされます。

備考

デフォルトでは、データベース内のスキーマとデータを表す XML が出力されます。出力は、アーカイブに使用するか、Ultra Light データベースをすべてのリリース間で移植するために使用できます。

同期プロファイルがあるデータベースを保存すると、以前のバージョンの Ultra Light ユーティリティと互換性がない XML になります。対処方法として、XML を編集し、次のようにマーク付けされたテキスト・セクションを削除します。

```
<syncprofiles>...</syncprofiles>
```

データベースのアンロード時に次のデータは保持されません。

- 同期ステータス、同期数、ロー削除。アンロードの前にデータベースを同期させる必要があります。
- Ultra Light ユーザのエントリ。

保持されたデータベース・オプションまたはプロパティを確認するには、ulload ユーティリティでデータベースを再ロードした後に ulinfo を実行します。

カラム・データが、-b で指定した最大サイズを超えていた場合、オーバフローは **.bin* ファイルに保存されます。このファイルは、次のいずれかのディレクトリにあります。

- XML ファイルと同じディレクトリ
- -f で指定したディレクトリ

ファイルは次の命名規則に従います。

```
tablename-columnname-rownumber.bin
```

-x オプションで Ultra Light テーブルに所有権を割り当てることができます。テーブルに所有者を割り当てるのは、出力の SQL 文を、SQL Anywhere データベースの作成または変更に使用する場合だけです。Ultra Light で読み込まれるとき、所有者名は通知されることなく無視されます。

このユーティリティはエラー・コードを返します。0 以外の値は処理に失敗したことを示します。

このユーティリティを使用して Windows Mobile デバイス上で直接データベースをアンロードすると、Ultra Light はアンロードまたは操作の実行前にデータベースをバックアップできません。したがって、これらの操作を手動で行ってから、これらのウィザードを実行してください。

参照

- 「Ultra Light 接続パラメータ」 235 ページ
- 「サポートされている終了コード」 262 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- 「Ultra Light 情報ユーティリティ (ulinfo)」 281 ページ

例

sample.udb データベースを *sample.xml* ファイルにアンロードします。

```
ulunload -c DBF=sample.udb sample.xml
```

sample.udb データベースのデータを *sample1.sql* という SQL ファイルにアンロードします。SQL ファイルが存在する場合は、上書きします。

```
ulunload -c DBF=sample.udb -d -y sample.sql
```


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 バージョン 8.0.2 から 9.0.x のデータベースやスキーマ・ファイル (*.*usm*) を XML ファイルへアンロードします。

構文

ulunloadold -c "*connection-string*" [*options*] *xml-file*

オプション	説明
-b *max-size*	XML ファイルに格納するカラム・データの最大サイズを設定します。デフォルトは 10 KB です。すべてのデータを XML ファイルに保存するには (最大サイズなし)、**-b -1** を使用します。
-c "*connection-string*"	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-f *directory*	-b で指定した最大サイズよりも大きいデータを格納するディレクトリを設定します。デフォルトは、XML ファイルと同じディレクトリです。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-v	冗長メッセージを表示します。
-y	確認メッセージを表示しないで *xml-file* を上書きします。
xml-file	データのアンロード先の XML ファイルの名前を設定します。

備考

Ultra Light バージョン 11 では、バージョン 8.x または 9.x のデータベースを直接アップグレードできません。このツールを使用して ulload で使用できる XML ファイルを生成してから、バージョン 11 のデータベースを作成します。このユーティリティで Ultra Light バージョン 11 のデータベースはアンロードしないでください。代わりに ulunload ユーティリティを使用してください。

データベースのアンロード時に次のデータは保持されません。

- 同期ステータス、同期数、ロー削除。アンロードの前にデータベースを同期させる必要があります。
- Ultra Light ユーザのエントリ。

保持されたデータベース・オプションまたはプロパティを確認するには、ulload ユーティリティでデータベースを再ロードした後に ulinfo を実行します。

カラム・データが、-b で指定した最大サイズを超えていた場合、オーバフローは **.bin* ファイルに保存されます。このファイルは、次のいずれかのディレクトリにあります。

- XML ファイルと同じディレクトリ
- -f で指定したディレクトリ

ファイル名は次の命名規則に従います。

```
tablename-columnname-rownumber.bin
```

0 以外のエラー・コード値は処理に失敗したことを示します。

このユーティリティを使用して、Windows Mobile デバイス上で直接データベースをアンロードすることはできません。まずデスクトップ・コンピュータにデータベースをコピーしてください。

参照

- 「Ultra Light 接続パラメータ」 235 ページ
- 「Ultra Light データベースへの XML のロード・ユーティリティ (ulload)」 288 ページ
- 「Ultra Light データベースのアンロード・ユーティリティ (ulunload)」 298 ページ
- 「Ultra Light 情報ユーティリティ (ulinfo)」 281 ページ

例

Ultra Light 8.0.x のスキーマ・ファイル *dbschema8.usm* を Ultra Light バージョン 11 のデータベース *db.udb* 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
```

Palm OS 用の Ultra Light バージョン 9.0.x のデータベース *palm9db.pdb* を Ultra Light バージョン 11 のデータベース *palm11db.pdb* 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 DBF=palm9db.pdb dbdata.xml
```

```
ulload -c DBF=palm11db.pdb -p Syb1 dbdata.xml
```


Ultra Light データベース検証ユーティリティ (ulvalid)

Ultra Light データベースのフル (通常の) 検証を実行します。これには、次のものが含まれます。

- データベース・ページ - チェックサムが有効な場合にこれを使用してすべてのデータベース・ページを検証します (checksum_level データベース作成パラメータを参照してください)。特定の重要なページには必ずチェックサムが含まれており、チェックサムのないページにも基本的な妥当性検査が実行されます。
- テーブル - テーブルのロー数が各インデックスの数と一致していることをチェックしてテーブルを検証します。
- インデックス - エントリが有効なローを参照することをチェックしてインデックスを検証します。ulvalid -e では、テーブルの検証のみを行うエクスプレス・チェックを実行します。

構文

ulvalid -c "*connection-string*" [*options*]

オプション	説明
-c "*connection-string*"	必須。*connection-string* で指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-e	エクスプレス検証。テーブルの検証のみを実行します。このオプションを使用すると、通常の検証よりも速く検証できます。
-oa	データベースが Ultra Light の旧バージョンで作成されたことをプロセスが識別した場合、そのプロセスをキャンセルします。
-or	読み込み専用モードでデータベースを開きます。Ultra Light によって、元のファイルのコピーが作成されます。このコピーを使用すると、データベースを変更しないでスクリプトをテストできます。コピーされたファイルの変更内容は終了時に破棄されます。 すでに Windows Mobile デバイスに配備されているデータベースにデスクトップから直接接続している場合は、このパラメータはサポートされません。
-ou	Ultra Light の旧リリース・バージョンで作成されたデータベースをアップグレードします。
-q	ユーティリティをクワイエット・モードで実行するように設定します。情報のバナー、バージョン番号、ステータス・メッセージが非表示になります。エラー・メッセージは引き続き表示されます。
-v	冗長メッセージを表示します。

備考

データベースの検証によってテーブル・メタデータの精度を検証し、ファイルが破損していないことを確認します。

参照

- 「Ultra Light データベースの検証」 14 ページ

例

データベース *sample.udb* のエクスプレス検証の例は、クワイエット・モードで動作します。

```
ulvalid -c DBF=sample.udb -e -q
```


Ultra Light のシステム・テーブル

目次

すべての Ultra Light データベースのスキーマは、複数のシステム・テーブルに記述されています。Ultra Light ではテーブルの所有権がサポートされていないので、すべての Ultra Light ユーザがシステム・テーブルにアクセスできます。

システム・テーブルの内容は、Ultra Light によってのみ変更できます。したがって、テーブルの内容の変更に UPDATE、DELETE、INSERT コマンドは使用できません。また、ALTER TABLE 文と DROP 文を使って、これらのテーブルの構造を変更することもできません。

Ultra Light システム・テーブルの表示または非表示

♦ **Sybase Central でシステム・オブジェクトを表示または非表示にするには、次の手順に従います。**

1. データベースに接続します。
2. データベースのオブジェクトをブラウズします。
3. コンテンツのウィンドウ枠を右クリックし、**[システム・オブジェクトを表示]** または **[システム・オブジェクトの非表示]** を選択します。
4. **[テーブル]** フォルダをクリックし、使用可能なテーブルをブラウズします。

systable システム・テーブル

systable システム・テーブルの各ローは、データベース内のテーブル 1 つを示します。

カラム名	カラム型	説明
column_count	UNSIGNED INT	テーブル内のカラム数。
index_count	UNSIGNED INT	テーブル内のインデックス数。
indexcol_count	UNSIGNED INT	テーブル内のすべてのインデックスのカラムの合計数。
map_handle	UNSIGNED INT	内部でのみ使用。
table_name	VARCHAR(128)	テーブルの名前。
object_id	UNSIGNED INT	テーブルのユニークな識別子。
sync_type	VARCHAR(128)	Mobile Link による同期に使用されます。**no_sync** (非同期)、**all_sync** (すべてのローの同期)、**normal_sync** (変更されたローのみ同期) のいずれか。
table_name	VARCHAR(128)	テーブルの名前。
table_type	TINYINT	**sys** (システム・テーブル) または **user** (通常のテーブル) のいずれか。
tpd_handle	UNSIGNED INT	内部でのみ使用。

制約

PRIMARY KEY (object_id)

syscolumn システム・テーブル

syscolumn システム・テーブルの各ローは、カラム 1 つを示します。

カラム名	カラム型	説明
column_name	VARCHAR(128)	カラムのユニークな識別子。
default	VARCHAR(128)	このカラムのデフォルト値。例：autoincrement。
domain	UNSIGNED INT	カラムのドメインを示す列挙値。
domain_info	SMALLINT	サイズが可変のドメインで使用されます。
nulls	CHAR(1)	カラムのデフォルトに NULL が許可されるかどうかを決定します。
object_id	UNSIGNED INT	カラムのユニークな識別子。
table_id	UNSIGNED INT	カラムが属するテーブルの識別子。

制約

PRIMARY KEY (table_id, object_id)

FOREIGN KEY (table_id) REFERENCES systable (object_id)

sysindex システム・テーブル

sysindex システム・テーブルの各ローは、データベース内のインデックス 1 つを示します。

カラム名	カラム型	説明
check_on_commit	UNSIGNED INT	すべての外部キーに一致するプライマリ・ローがあることを確認するために参照整合性をいつチェックするかを示します。type が **foreign** の場合にのみ必要です。
index_name	UNSIGNED INT	インデックスの名前。
ixcol_count	UNSIGNED INT	インデックス内のカラム数。
nullable	BIT	type が **foreign** の場合にのみ必要です。NULL が許可されるかどうかを示します。
object_id	UNSIGNED INT	インデックスのユニークな識別子。
primary_index_id	UNSIGNED INT	type が **foreign** の場合にのみ必要です。プライマリ・インデックスの識別子をリストします。
primary_table_id	UNSIGNED INT	type が **foreign** の場合にのみ必要です。プライマリ・テーブルの識別子をリストします。
root_handle	UNSIGNED INT	内部でのみ使用。
table_id	UNSIGNED INT	インデックスが適用されるテーブルのユニークな識別子。
type	SMALLINT (10)	インデックスの型。次のいずれかです。 ● **primary** ● **foreign** ● **key** ● **unique** ● **index**
hash_size	SHORT	インデックスのハッシュに使用されるハッシュ・サイズ。

制約

PRIMARY KEY (table_id, object_id)

FOREIGN KEY (table_id) REFERENCES systable (object_id)

参照

- 「sysixcol システム・テーブル」 311 ページ

sysixcol システム・テーブル

sysixcol システム・テーブルの各ローは、sysindex にリストされているインデックスのカラム 1 つを示します。

カラム名	カラム型	説明
column_id	UNSIGNED INT	インデックス対象カラムのユニークな識別子。
index_id	UNSIGNED INT	このインデックス・カラムが属するインデックスのユニークな識別子。
order	CHAR(1)	インデックス内のカラムが昇順 (A) か、降順 (D) かを示します。
sequence	SMALLINT	インデックス内のカラムの順序。
table_id	UNSIGNED INT	インデックスが適用されるテーブルのユニークな識別子。

制約

PRIMARY KEY (table_id, index_id, sequence)

FOREIGN KEY (table_id, index_id) REFERENCES sysindex (table_id, object_id)

FOREIGN KEY (table_id, column_id) REFERENCES syscolumn (table_id, object_id)

参照

- 「sysindex システム・テーブル」 309 ページ

syspublication システム・テーブル

syspublication システム・テーブルの各ローは、パブリケーションを示します。

カラム名	カラム型	説明
download_timestamp	UNSIGNED INT	最後のダウンロードの時刻。
last_sync	UNSIGNED BIGINT	アップロードの進行状況を追跡するために使用します。
publication_id	UNSIGNED INT	パブリケーションのユニークな識別子。
publication_name	CHAR(128)	パブリケーションの名前。

制約

PRIMARY KEY (publication_id)

参照

- 「sysarticle システム・テーブル」 313 ページ

sysarticle システム・テーブル

sysarticle システム・テーブルの各ローは、パブリケーションに属するテーブルを示します。

カラム名	カラム型	説明
publication _id	UNSIGNED INT	このアーティクルが属するパブリケーションの識別子。
table_id	UNSIGNED INT	パブリケーションに属するテーブルの識別子。
where_expr	TINY INT	ローをフィルタする述部 (オプション)。

制約

PRIMARY KEY (publication_id, table_id)

FOREIGN KEY (publication_id) REFERENCES syspublication (publication_id)

FOREIGN KEY (table_id) REFERENCES systable (object_id)

参照

- 「syspublication システム・テーブル」 312 ページ

sysuldata システム・テーブル

sysuldata システム・テーブルの各ローは、オプションとプロパティの名前と値の組み合わせを示します。

カラム名	カラム型	説明
long_setting	LONGBINARY	長い値用の BLOB。
name	VARCHAR(128)	プロパティ名。
setting	VARCHAR(128)	プロパティの値。
type	VARCHAR(128)	**sys** (内部)、**opt** (オプション)、または **prop** (プロパティ) のいずれか。

制約

PRIMARY KEY (name, type)

参照

- 「Ultra Light データベース・プロパティ」 221 ページ

Ultra Light SQL リファレンス

この項は、Ultra Light SQL のリファレンスです。Ultra Light SQL は、SQL Anywhere データベースでサポートされている SQL のユニークなサブセットです。

Ultra Light SQL 要素

目次

Ultra Light のキーワード

各 SQL 文には 1 つまたは複数のキーワードが含まれています。SQL 文のキーワードでは大文字と小文字を区別しませんが、このマニュアルではキーワードを大文字で表記します。キーワードの中には、二重引用符で囲まないと識別子として使用できないものがあります。このようなキーワードを、「**reserved words**」と呼びます。「予約語」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

注意
Ultra Light では、SQL Anywhere キーワードのサブセットだけがサポートされています。ただし、将来のリリースで問題が発生しないように、SQL Anywhere のすべての予約語が Ultra Light の予約語でもあると考えてください。

Ultra Light の識別子

「**Identifiers**」は、ユーザ ID、テーブル、カラムなど、データベースのオブジェクト名を表します。識別子の最大長は 128 バイトです。

次のいずれかの条件と一致した場合に、識別子を二重引用符で囲みます。

- 識別子にスペースが含まれる。
- 識別子の先頭文字がアルファベット文字ではないデータベースの照合順は、どの文字がアルファベットまたは数字として扱われるかを指定する。
- 識別子に予約語が含まれる。「予約語」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- 識別子にアルファベット文字や数字以外の文字が含まれる。

1 つの円記号は、エスケープ文字として使用される場合のみ識別子で使用できます。

Ultra Light の文字列

文字列を使用して、文字データをデータベースに格納します。Ultra Light は、SQL Anywhere と同じ文字列の規則をサポートしています。文字列の比較結果や文字列のソート順は、データベース、文字セット、照合順における大文字と小文字の区別によって決まります。これらのプロパティは、データベースの作成時に設定されます。

参照

- 「文字列」『SQL Anywhere サーバ - SQL リファレンス』
- 「Ultra Light 文字セット」 38 ページ

Ultra Light のコメント

コメントは、SQL 文または文ブロックに説明テキストを付加するために使用します。Ultra Light ランタイムは、コメントを実行しません。

Ultra Light では次のコメント・インジケータを使用できます。

- **-- (二重ハイフン)** データベース・サーバは、この行の残りの文字を無視します。このインジケータは、SQL/2003 のコメント・インジケータです。
- **// (二重スラッシュ)** 二重スラッシュは、二重ハイフンと同じ意味です。
- **/* ... */ (スラッシュ – アスタリスク)** 2 つのコメント・マーカの間にある文字は、すべて無視されます。2 つのコメント・マーカは、同じ行にあっても、別の行にあってもかまいません。このスタイルで示されたコメントは、ネストできます。このスタイルは、C スタイル・コメントとも呼ばれます。

> **注意**
> パーセント記号 (%) は、Ultra Light ではサポートされていません。

例

- 次に、二重ハイフンのコメントの使用例を示します。

```
CREATE TABLE borrowed_book (
  loaner_name CHAR(100)     PRIMARY KEY,
  date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,
  date_returned       DATE,
  book              CHAR(20)
  FOREIGN KEY book REFERENCES library_books (isbn),
);
--This statement creates a table for a library database to hold information on borrowed books.
--The default value for date_borrowed indicates that the book is borrowed on the day the entry is
made.
--The date_returned column is NULL until the book is returned.
```

- 次に、C スタイル・コメントの使用例を示します。

```
CREATE TABLE borrowed_book (
  loaner_name CHAR(100)     PRIMARY KEY,
  date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,
  date_returned       DATE,
  book              CHAR(20)
  FOREIGN KEY book REFERENCES library_books (isbn),
);
/* This statement creates a table for a library database to hold information on borrowed books.
The default value for date_borrowed indicates that the book is borrowed on the day the entry is
made.
The date_returned column is NULL until the book is returned. */
```

Ultra Light の数値

数値を使用して、数値データをデータベースに格納します。数値は次のようなデータです。

- 任意の数字の並び
- 小数部が続く
- オプションでマイナス記号 (-) またはプラス記号 (+) を含む
- E の後に指数値が続く

たとえば、次に示す数値はすべて Ultra Light でサポートされています。

42

-4.038

.001

3.4e10

1e-10

Ultra Light の NULL 値

SQL Anywhere と同様に、NULL はすべてのデータ型のすべての有効な値とは異なる特殊な値です。ただし、NULL 値はすべてのデータ型で使用できます。NULL は、情報が不明 (値なし) であるか、該当しないことを表すために使用します。「NULL 値」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Ultra Light の特別値

特別値は、式や、テーブル作成時のカラムのデフォルトに使用できます。

CURRENT DATE 特別値

現在の年、月、日を返します。

データ型

DATE

備考

返される日付は、SQL 文が Ultra Light ランタイムで実行されたときのシステム・クロックの読み取り値が基準になります。CURRENT DATE を次のいずれかの状態で使用すると、すべての値は、個別のクロック読み取り値が基準になります。

- 同一文で CURRENT DATE を複数回使用
- 同一文で CURRENT DATE を CURRENT TIME または CURRENT TIMESTAMP と組み合わせて使用
- 同一文で CURRENT DATE を NOW 関数または GETDATE 関数と組み合わせて使用

参照

- 「Ultra Light の式」 343 ページ
- 「GETDATE 関数 [日付と時刻]」 404 ページ
- 「NOW 関数 [日付と時刻]」 428 ページ

CURRENT TIME 特別値

現在の時、分、秒 (小数位あり) で構成される時刻を返します。

データ型

TIME

備考

秒は小数第 6 位まで格納されます。現在時刻の精度はシステム・クロックの精度によって制限されます。

返される日付は、SQL 文が Ultra Light ランタイムで実行されたときのシステム・クロックの読み取り値が基準になります。CURRENT TIME を次のいずれかと組み合わせて使用すると、すべての値は、個別のクロック読み取り値が基準になります。

- 同一文で CURRENT TIME を複数回使用

- 同一文で CURRENT TIME を CURRENT DATE または CURRENT TIMESTAMP と組み合わせて使用
- 同一文で CURRENT TIME を NOW 関数または GETDATE 関数と組み合わせて使用

参照

- 「Ultra Light の式」 343 ページ
- 「GETDATE 関数 [日付と時刻]」 404 ページ
- 「NOW 関数 [日付と時刻]」 428 ページ

CURRENT TIMESTAMP 特別値

CURRENT DATE と CURRENT TIME を結合して、TIMESTAMP 値を形成します。年、月、日、時、分、秒、秒の小数位で構成されます。

データ型

TIMESTAMP

備考

秒は小数第 3 位まで格納されます。精度はシステム・クロックの精度によって制限されます。

DEFAULT CURRENT TIMESTAMP で宣言されたカラムには、ユニークな値が入るとはかぎりません。

CURRENT TIMESTAMP が返す情報は、GETDATE 関数と NOW 関数が返す情報と同じです。

CURRENT_TIMESTAMP は、CURRENT TIMESTAMP と同じです。

返される日付は、SQL 文が Ultra Light ランタイムで実行されたときのシステム・クロックの読み取り値が基準になります。CURRENT TIMESTAMP を次のいずれかと組み合わせて使用すると、すべての値は、個別のクロック読み取り値が基準になります。

- 同一文で CURRENT TIMESTAMP を複数回使用
- 同一文で CURRENT TIMESTAMP を CURRENT DATE または CURRENT TIME と組み合わせて使用
- 同一文で CURRENT TIMESTAMP を NOW 関数または GETDATE 関数と組み合わせて使用

参照

- 「CURRENT TIME 特別値」 324 ページ
- 「Ultra Light の式」 343 ページ
- 「NOW 関数 [日付と時刻]」 428 ページ
- 「GETDATE 関数 [日付と時刻]」 404 ページ
- 「NOW 関数 [日付と時刻]」 428 ページ

SQLCODE 特別値

SQLCODE 特別値が評価された時点の値です。

データ型

String

備考

SQLCODE 値は各文の後に設定されます。SQLCODE をチェックして、文の実行が成功したかどうかを確認できます。

参照

- 「Ultra Light の式」 343 ページ
- エラー・メッセージ

例

SELECT 文を使用して、結果セットから新しいローをフェッチしようとするたびにエラー・コードを生成します。例：SELECT a, b, SQLCODE FROM MyTable

Ultra Light の日付と時刻

日付と時刻の関数の多くは、日付と時刻の単位で構成される日付を使用します。Ultra Light と SQL Anywhere では同じ日付の単位がサポートされています。「日付の単位」 367 ページを参照してください。

Ultra Light のデータ型

Ultra Light SQL では、次のデータ型を使用できます。

- 整数
- Decimal
- 浮動小数点
- 文字
- Binary
- 日付／時刻

注意
Ultra Light SQL では、ドメイン (ユーザ定義のデータ型) はサポートされていません。

注意
LONGVARCHAR データ型と LONGBINARY データ型は連結できません。「文字列演算子」 357 ページを参照してください。

ホスト変数は、サポートされる任意のデータ型について作成できます。Ultra Light は、SQL Anywhere で使用できるデータ型のサブセットをサポートします。次のデータ型は、Ultra Light データベースでサポートされている SQL データ型です。

データ型	説明
BIT	ブール値 (0 または 1)。「BIT 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。
{ **CHAR** | **CHARACTER** } (*max-length*)	*max-length* の文字データです。範囲は 1 ～ 32767 バイトです。「CHAR 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。 式を評価するときのテンポラリ文字値の最大長は 2048 バイトです。
VARCHAR(*max-length* **)**	VARCHAR は、最大長 *max-length* の可変長の文字データに使用します。「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。

データ型	説明
LONG VARCHAR	任意の長さの文字データです。SQL 文の条件 (WHERE 句の条件など) は、LONG VARCHAR カラムでは実行できません。LONG VARCHAR カラムで実行可能な演算は、これらの挿入、更新、削除、またはクエリの *select-list* へのこれらの指定のみです。「LONG 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。 LONGVARCHAR データに、または LONGVARCHAR データから文字列をキャストできます。
[**UNSIGNED**] **BIGINT**	8 バイトの記憶領域を必要とする整数です。「BIGINT 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。
{ **DECIMAL** | **DEC** | **NUMERIC** } **(** *precision* , *scale*] **)**]	*precision* (合計桁数) と *scale* (小数点以下の桁数) の 2 つの部分で小数を表します。「DECIMAL データ型」『SQL Anywhere サーバ - SQL リファレンス』、「NUMERIC データ型」『SQL Anywhere サーバ - SQL リファレンス』、「Ultra Light precision 作成パラメータ」 208 ページ、「Ultra Light scale 作成パラメータ」 210 ページを参照してください。
DOUBLE [**PRECISION**]	倍精度の浮動小数点数です。このデータ型では、PRECISION は DOUBLE データ型名のオプション部分です。「DOUBLE 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。
FLOAT [**(** *precision* **)**]	単精度または倍精度の浮動小数点数です。「FLOAT 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。
[**UNSIGNED**] { **INT** | **INTEGER** }	4 バイトの記憶領域を必要とする符号なし整数です。「INTEGER 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。
REAL	4 バイトで格納される単精度浮動小数点数。「REAL 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。
[**UNSIGNED**] **SMALLINT**	2 バイトの記憶領域を必要とする整数です。「SMALLINT 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。
[**UNSIGNED**] **TINYINT**	1 バイトの記憶領域を必要とする整数です。「TINYINT 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。

データ型	説明
DATE	年、月、日などの暦日です。「DATE 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。
TIME	時、分、秒、秒以下で構成される時間です。「TIME 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。
DATETIME	TIMESTAMP と同じです。「DATETIME 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。
TIMESTAMP	年、月、日、時、分、秒、秒以下で構成される時刻です。「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。
VARBINARY (*max-length* **)**	BINARY と同じです。「VARBINARY 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。
BINARY (*max-length* **)**	最大長が *max-length* バイトのバイナリ・データです。最大長は 2048 バイト以内である必要があります。「BINARY 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。
LONG BINARY	任意の長さのバイナリ・データです。SQL 文の条件 (WHERE 句の条件など) では、LONG BINARY カラムは使用できません。LONG BINARY カラムで実行可能な演算は、これらの挿入、更新、削除、またはクエリの *select-list* へのこれらの指定のみです。「LONG BINARY 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。 LONGBINARY データに、または LONGBINARY データから値をキャストできます。
UNIQUEIDENTIFIER	通常は、ローを一意に識別する UUID (ユニバーサル・ユニーク識別子) 値を保持するために、プライマリ・キーまたはその他のユニーク・カラムに使用されます。Ultra Light には、UUID 値を生成する機能が用意されています。これらの値は、あるコンピュータで生成された値が、別のコンピュータで生成された UUID と一致しないように生成されます。したがって、この方法で生成された UNIQUEIDENTIFIER 値は、同期環境でキーとして使用できます。「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。

ユーザ定義データ型とそれに相当する型

SQL Anywhere データベースとは異なり、Ultra Light ではユーザ定義データ型はサポートされていません。SQL Anywhere の組み込みエイリアスに相当する Ultra Light データ型を次の表に示します。

SQL Anywhere データ型	Ultra Light で相当する型
MONEY	NUMERIC(19,4)
SMALLMONEY	NUMERIC(10,4)
TEXT	LONG VARCHAR
XML	LONG VARCHAR

データ型の明示的な変換

Ultra Light では、CAST または CONVERT 関数を使用することで、データ型の変換を明示的に要求できます。

注意
ほとんどの場合、自己キャストは処理にまったく影響しません。ただし、CHAR/VARCHAR、BINARY/VARBINARY、および NUMERIC への自己キャストでは、何らかの処理が行われます。

CAST または CONVERT は、次の表に示すように、ほとんどのデータ型の組み合わせで使用できます。

ただし、変換できるかどうかは変換に使用される値に左右されます。次の表の「**Value-dependent**」欄は、特定の変換エラーが発生しないよう、値に新しいデータ型との互換性が必要であることを示します。次に例を示します。

- **varchar "1234"** を **long** にキャストした場合、この変換はサポートされます。ただし、**varchar "hello"** を **long** にキャストした場合、**hello** が数値ではないため **SQLE_CONVERSION_ERROR** エラーが発生します。
- **long 1234** を **short** にキャストした場合、この変換はサポートされます。ただし、**long 1000000** を **short** にキャストした場合、**1000000** は short で保持できる数値の範囲を超えているため **SQLE_OVERFLOW_ERROR** エラーが発生します。

変換元	可	不可	値依存
BINARY または VARBINARY	CHAR または VARCHAR BINARY LONG BINARY BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG	LONG VARCHAR REAL TIME TIMESTAMP DOUBLE DATE	NUMERIC UID[1]
LONG BINARY	BINARY LONG BINARY	BIT CHAR または VARCHAR LONG VARCHAR TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG REAL DOUBLE NUMERIC DATE TIME TIMESTAMP UID	なし

変換元	可	不可	値依存
BIT	CHAR または VARCHAR BINARY BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT REAL SIGNED BIG DOUBLE NUMERIC	LONG VARCHAR LONG BINARY DATE TIME TIMESTAMP UID	なし
CHAR または VARCHAR	BINARY または VARBINARY CHAR または VARCHAR LONG VARCHAR	LONG BINARY	BIT TINYINT SIGNED SHORT SHORT INT LONG INT SIGNED LONG BIGINT SIGNED BIG DOUBLE NUMERIC REAL DATE TIME TIMESTAMP UID

変換元	可	不可	値依存
LONG VARCHAR	CHAR または VARCHAR LONG VARCHAR	BINARY または VARBINARY LONG BINARY BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG REAL NUMERIC DATE TIME TIMESTAMP DOUBLE UID	

変換元	可	不可	値依存
TINYINT	BINARY または VARBINARY CHAR または VARCHAR TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG REAL DOUBLE NUMERIC	LONG VARCHAR LONG BINARY DATE TIME TIMESTAMP UID	
SHORT INT	BINARY または VARBINARY CHAR または VARCHAR SHORT INT LONG INT SIGNED LONG BIGINT SIGNED BIG REAL DOUBLE NUMERIC	LONG VARCHAR LONG BINARY DATE TIME TIMESTAMP UID	BIT TINYINT SIGNED SHORT

変換元	可	不可	値依存
SIGNED SHORT	BINARY または VARBINARY CHAR または VARCHAR SIGNED SHORT SIGNED LONG SIGNED BIG REAL DOUBLE NUMERIC	LONG VARCHAR LONG BINARY DATE TIME TIMESTAMP UID	SHORT INT LONG INT BIGINT BIT TINYINT
LONG INT	BINARY または VARBINARY CHAR または VARCHAR LONG INT BIGINT SIGNED BIG REAL DOUBLE NUMERIC	LONG VARCHAR LONG BINARY DATE TIME TIMESTAMP UID	BIT TINYINT SHORT INT SIGNED SHORT SIGNED LONG
SIGNED LONG	BINARY または VARBINARY CHAR または VARCHAR SIGNED LONG SIGNED BIG REAL DOUBLE NUMERIC DATE TIMESTAMP	LONG VARCHAR LONG BINARY TIME UID	BIT TINYINT SHORT INT SIGNED SHORT LONG INT BIGINT

変換元	可	不可	値依存
BIGINT	BINARY または VARBINARY CHAR または VARCHAR BIGINT REAL DOUBLE NUMERIC	LONG VARCHAR LONG BINARY DATE TIME TIMESTAMP UID	BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG SIGNED BIG
SIGNED BIG	BINARY または VARBINARY CHAR または VARCHAR SIGNED BIG REAL DOUBLE NUMERIC DATE TIMESTAMP	LONG VARCHAR LONG BINARY TIME UID	BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT
REAL	CHAR または VARCHAR REAL DOUBLE NUMERIC	LONG VARCHAR BINARY または VARBINARY LONG BINARY DATE TIME TIMESTAMP UID	BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG

変換元	可	不可	値依存
DOUBLE	CHAR または VARCHAR DOUBLE NUMERIC	LONG VARCHAR BINARY または VARBINARY LONG BINARY DATE TIME TIMESTAMP UID	BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG REAL
NUMERIC	CHAR または VARCHAR REAL NUMERIC DOUBLE	LONG VARCHAR LONG BINARY DATE TIME TIMESTAMP UID	BINARY または VARBINARY[2] BIT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG TINYINT

変換元	可	不可	値依存
DATE	CHAR または VARCHAR SIGNED LONG SIGNED BIG DATE TIMESTAMP	LONG VARCHAR LONG BINARY BIT TINYINT SHORT INT SIGNED SHORT LONG INT BIGINT REAL DOUBLE NUMERIC TIME BINARY または VARBINARY UID	

変換元	可	不可	値依存
TIME	CHAR または VARCHAR TIME TIMESTAMP	LONG VARCHAR LONG BINARY BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG REAL DOUBLE NUMERIC DATE BINARY または VARBINARY UID	

変換元	可	不可	値依存
TIMESTAMP	CHAR または VARCHAR SIGNED LONG SIGNED BIG DATE TIME TIMESTAMP	LONG VARCHAR LONG BINARY BIT TINYINT SHORT INT SIGNED SHORT LONG INT BIGINT REAL DOUBLE NUMERIC BINARY または VARBINARY UID	
UID	CHAR または VARCHAR UID	LONG VARCHAR LONG BINARY BIT TINYINT SHORT INT SIGNED SHORT LONG INT SIGNED LONG BIGINT SIGNED BIG REAL DOUBLE NUMERIC DATE TIME TIMESTAMP	BINARY または VARBINARY[1]

[1] UUID との互換性を保つには、BINARY 値の長さが 16 バイトである必要があります。

[2] ソースの NUMERIC 値が BIGINT にキャストできる場合のみ機能します。

Ultra Light の式

式は、多くの場合、カラム参照の形式でデータを関数や演算子と組み合わせることによって形成されます。

構文

expression:
case-expression
| *constant*
| [*correlation-name*.]*column-name*
| - *expression*
| *expression operator expression*
| (*expression*)
| *function-name* (*expression*, ...)
| *if-expression*
| *special value*
| *input-parameter*

パラメータ

case-expression:
CASE *expression*
WHEN *expression*
THEN *expression*,...
[**ELSE** *expression*]
END

alternative form of case-expression:
CASE
WHEN *search-condition*
THEN *expression*,...
[**ELSE** *expression*]
END

constant:
integer | *number* | *string* | *host-variable*

special-value:
CURRENT { **DATE** | **TIME** | **TIMESTAMP** }
| **NULL**
| **SQLCODE**
| **SQLSTATE**

if-expression:
IF *condition*
THEN *expression*
[**ELSE** *expression*]
ENDIF

input-parameter:
{ **?** | :*name* [: *indicator-name*] }

operator:
{ **+** | **-** | ***** | **/** | **||** | **%** }

参照

- 「式内の定数」 344 ページ
- 「Ultra Light の特別値」 324 ページ
- 「式内のカラム名」 344 ページ
- 「Ultra Light SQL 関数」 365 ページ
- 「式のサブクエリ」 347 ページ
- 「Ultra Light の探索条件」 349 ページ
- 「Ultra Light のデータ型」 328 ページ
- 「CASE 式」 345 ページ
- 「入力パラメータ」 348 ページ

式内の定数

Ultra Light における定数とは、数値または文字列リテラルです。

構文

' *constant* **'**

使用法

文字列定数は、一重引用符 (')で囲まれています。

文字列内にアポストロフィを表すには、一重引用符を 2 つ続けて使用します (')。

参照

- 「エスケープ・シーケンス」 『SQL Anywhere サーバ - SQL リファレンス』

例

所有を表すフレーズを使用するには、次のような文字列リテラルを入力します。

'John''s database'

式内のカラム名

式内の識別子です。

構文

correlation-name.column-name

備考

カラム名は、オプションの相関名 (通常はテーブルの名前) の後に続きます。

カラム名がキーワードであるか、カラム名に英字、数字、アンダースコア以外の文字が使用されている場合は、二重引用符 ("") で囲んでください。次の例は、有効なカラム名です。

```
Employees.Name
address
```



```
"date hired"
"salary"."date paid"
```

参照

- 「Ultra Light FROM 句」 499 ページ

IF 式

データの特定サブセットを返す探索条件を設定します。

構文 1

```
IF search-condition
THEN expression1
[ ELSE expression2 ]
ENDIF
```

備考

互換性を保つために、この式は ENDIF または END IF で終了できます。

この式は次のように返します。

- *search-condition* が TRUE の場合、IF 式は *expression1* を返します。
- *search-condition* が FALSE で、ELSE 句が指定されている場合、IF 式は *expression2* を返します。
- *search-condition* が FALSE で *expression2* がない場合、IF 式は NULL を返します。
- *search-condition* が UNKNOWN の場合、IF 式は NULL を返します。

参照

- 「NULL 値」 『SQL Anywhere サーバ - SQL リファレンス』
- 「探索条件」 『SQL Anywhere サーバ - SQL リファレンス』

CASE 式

SQL の条件式です。

構文 1

```
CASE  expression1
WHEN expression2 THEN expression3, ...
[ ELSE expression4 ]
END
```

```
SELECT id,
  ( CASE name
    WHEN 'Tee Shirt' THEN 'Shirt'
    WHEN 'Sweatshirt' THEN 'Shirt'
    WHEN 'Baseball Cap' THEN 'Hat'
```

```
    ELSE 'Unknown'
  END ) as Type
FROM Product;
```

構文 2

CASE
WHEN *search-condition*
THEN *expression1*, ...
[**ELSE** *expression2*]
END

備考

互換性を保つために、この式は ENDCASE または END CASE で終了できます。

CASE 式は、正規表現が使用できればどこでも使用できます。

構文 1　CASE キーワードに続く式が最初の WHEN キーワードに続く式と等しい場合、対応する THEN キーワードの後の式が返されます。それ以外の場合、ELSE キーワードがあればそれに続く式が返されます。

たとえば、下記のコードでは CASE 式が SELECT 文の 2 番目の句として使用されています。この式によって、name カラムの値が Sweatshirt であるローが Product テーブルから選択されます。

構文 2　最初の WHEN キーワードに続く search-condition が TRUE の場合、対応する THEN キーワードに続く式が返されます。それ以外の場合、ELSE 句があればそれに続く式が返されます。

省略形 CASE 式の NULLIF 関数　NULLIF 関数は、CASE 文を省略形で記述する方法の 1 つです。NULLIF の構文は、次のとおりです。

NULLIF (*expression-1*, *expression-2*)

NULLIF は 2 つの式の値を比較します。1 番目の式と 2 番目の式が等しい場合、NULLIF は NULL を返します。1 番目の式と 2 番目の式が異なる場合、NULLIF は 1 番目の式を返します。

例

次の文では、CASE 式が SELECT 文の 3 番目の句として使用され、探索条件と文字列を関連付けています。name カラムの値が **Tee Shirt** の場合は、このクエリによって **Sale** が返されます。name カラムの値が **Tee Shirt** ではなく、quantity が 50 より大きい場合は、**Big Sale** が返されます。ただし、それ以外の場合は、このクエリによって **Regular price** が返されます。

```
SELECT id, name,
  ( CASE
    WHEN name='Tee Shirt' THEN 'Sale'
    WHEN quantity >= 50  THEN 'Big Sale'
    ELSE 'Regular price'
  END ) as Type
FROM Product;
```

集合式

Ultra Light ランタイムでは提供されない集計の計算を実行します。

構文

SUM(*expression* **)**

備考

集合式は、一連のローから単一の値を計算します。

集合式は、1 つの集合関数が使用される式か、1 つ以上のオペランドがある式です。

SELECT 文に GROUP BY 句がない場合、*select-list* の式にすべての集合式が含まれているか、または集合式がまったく含まれていない必要があります。SELECT 文に GROUP BY 句がある場合、*select-list* の非集合式を GROUP BY リストに記載します。

例

たとえば、次のクエリは、employee テーブル内の従業員の給与合計を計算します。このクエリでは、SUM(salary) が集合式です。

```
SELECT SUM( salary )
FROM employee;
```

式のサブクエリ

別の SELECT 文の内部でネストされた SELECT 文です。

構文

サブクエリは、通常のクエリのように構成されます。

備考

Ultra Light のサブクエリの参照は、次の場合にのみ使用できます。

- FROM 句内のテーブル式として。この形式のテーブル式 (**derived tables**) には、SELECT リスト内の値をフェッチする派生テーブル名とカラム名が必要です。
- EXISTS、ANY、ALL、IN の各探索条件に値を指定するため。

サブクエリは、サブクエリの前 (左) に指定されている名前を参照して作成できます。これは、左側への外部参照とも呼ばれます。サブクエリ内の項目は参照できません。これは、内部参照とも呼ばれます。

参照

- 「Ultra Light SELECT 文」 507 ページ
- 「サブクエリの使用」 『SQL Anywhere サーバ - SQL の使用法』
- 「Ultra Light の探索条件」 349 ページ

例

次のサブクエリは、在庫数の少ない項目のすべての製品 ID をリストするのに使用します。

```
FROM SalesOrderItems
 ( SELECT ID
```

```
FROM Products
WHERE Quantity < 20 );
```

入力パラメータ

エンド・ユーザが準備文に値を指定できるようにするためのプレースホルダとして機能します。文は、ユーザが指定する値を使用して実行されます。

構文

{ **?** | :*name* [: *indicator-name*] }

備考

式でプレースホルダ文字 ? または名前形式を使用します。入力パラメータは、カラム名または定数を使用できる任意の場所で使用できます。

文に値が渡される具体的なメカニズムは、Ultra Light クライアントの作成に使用する API によって異なります。

名前形式の使用　入力パラメータの名前形式には特別な意味があります。一般に、常に *name* を使用して、実際の値を渡す複数のロケーションを指定します。

Embedded SQL アプリケーションの場合にのみ、*indicator-name* が、NULL インジケータが配置される変数を指定します。名前形式と他のコンポーネントを同時に使用した場合、*indicator-name* は無視されます。

データ型の抽出　入力パラメータのデータ型は、文が次のいずれかのパターンから準備されるときに抽出されます。

- **CAST (? AS** *type* **)**

  ここで、*type* は CHAR(32) などのデータベースの型指定です。
- バイナリ演算子の 1 つのオペランドが入力パラメータである。型が抽出され、オペランドの型になります。

Ultra Light で型を抽出できなかった場合は、エラーが発生します。次に例を示します。

- **-?**：オペランドが単項である。
- **? + ?**：両方共入力パラメータである。

参照

- 「ホスト変数の使用」 『Ultra Light - C/C++ プログラミング』
- 「文の準備」 『SQL Anywhere サーバ - プログラミング』
- Ultra Light C/C++：「データ操作：挿入、削除、更新」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「データ操作：INSERT、UPDATE、DELETE」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「データ操作：INSERT、UPDATE、DELETE」 『Ultra Light - M-Business Anywhere プログラミング』

例

次の Embedded SQL 文には 2 つの入力パラメータがあります。

```
INSERT INTO MyTable VALUES ( :v1, :v2, :v1);
```

v1 の最初のインスタンスが、文内の v2 と v1 の両方のロケーションにその値を渡します。

Ultra Light の探索条件

WHERE 句、HAVING 句、ジョインの ON フレーズ、または IF 式の探索条件を指定します。探索条件は、述部とも言います。

構文

search-condition:
expression compare expression
| *expression* **IS** [**NOT**] { **NULL** | **TRUE** | **FALSE** | **UNKNOWN** }
| *expression* [**NOT**] **BETWEEN** *expression* **AND** *expression*
| *expression* [**NOT**] **IN (** *expression*, ... **)**
| *expression* [**NOT**] **IN (** *subquery* **)**
| *expression* [**NOT**] { **ANY** | **ALL** } **(** *subquery* **)**
| *expression* [**NOT**] **EXISTS (** *subquery* **)**
| **NOT** *search-condition*
| *search-condition* **AND** *search-condition*
| *search-condition* **OR** *search-condition*
| (*search-condition*)

パラメータ

compare:
= | > | < | >= | <= | <> | != | !< | !>

備考

Ultra Light では、次の箇所で探索条件を指定します。

- WHERE 句
- HAVING 句
- ON フレーズ
- SQL クエリ

探索条件は、SELECT 文の FROM 句で使用してテーブル内のローのサブセットを選択するか、IF や CASE などの式で使用して特定の値を選択できます。Ultra Light では、すべての条件が TRUE、FALSE、または UNKNOWN のいずれかに評価されます。これらを組み合わせた場合、**three-valued logic** と呼ばれます。比較される値のいずれかが NULL の場合、比較結果は UNKNOWN になります。探索条件は、条件の結果が TRUE の場合にのみ満たされます

Ultra Light では、次のタイプの探索条件がサポートされています。

- ALL 条件
- ANY 条件
- BETWEEN 条件
- EXISTS 条件
- IN 条件

この後の各項で、これらの条件について説明します。

注意
サブクエリは、多数の探索条件で使用される式の重要なクラスを構成します。

参照

- 「比較演算子」 350 ページ
- 「3 値的論理」 『SQL Anywhere サーバ - SQL リファレンス』
- 「式のサブクエリ」 347 ページ

比較演算子

探索条件で、複数の式を比較できる演算子です。

構文

expression operator expression

パラメータ

演算子	意味
=	等価
[**NOT**] **LIKE**	テキスト比較 (場合よっては正規表現を使用)
>	より大きい
<	より小さい
>=	大きいかまたは等価
<=	小さいかまたは等価
!=	等価ではない
<>	等価ではない
!>	以下

演算子	意味
!<	以上

備考

日付の比較　日付を比較するときには、< はより古いことを意味し、> はより新しいことを意味します。

LONG VARCHAR 値や LONG BINARY 値の比較　Ultra Light は、LONG VARCHAR 値や LONG BINARY 値を使用した比較をサポートしていません。

大文字と小文字の区別　Ultra Light では、比較処理は、該当のデータベースの文字の区別に合わせて実行されます。デフォルトでは、Ultra Light データベースは大文字と小文字を区別しないで作成されます。

NOT 演算子　NOT 演算子は式を否定します。

参照

- 「論理演算子」 351 ページ
- 「Ultra Light の探索条件」 349 ページ

例

次の 2 つのクエリはどちらも、単価が $10 以下の T シャツ (Tee Shirt) と野球帽 (BaseBall Cap) をすべて検索します。ただし、否定の論理演算子 (NOT) と否定の比較演算子 (!>) では、位置が異なることに注意してください。

```
SELECT ID, Name, Quantity
FROM Products
WHERE (name = 'Tee Shirt' OR name = 'BaseBall Cap')
AND NOT UnitPrice > 10;

SELECT ID, Name, Quantity
FROM Products
WHERE (name = 'Tee Shirt' OR name = 'BaseBall Cap')
AND UnitPrice !> 10;
```

論理演算子

次のいずれかを行います。

- 条件の比較 (AND、OR、NOT)
- 式 (IS) の真理値または NULL 値のテスト

構文 1

condition1 logical-operator condition2

構文 2

NOT *condition*

構文 3

expression **IS** [**NOT**] { *truth-value* | **NULL** }

備考

探索条件は、SELECT 文の FROM 句で使用してテーブル内のローのサブセットを選択するか、IF や CASE などの式で使用して特定の値を選択できます。Ultra Light では、すべての条件が TRUE、FALSE、または UNKNOWN のいずれかに評価されます。これらを組み合わせた場合、**three-valued logic** と呼ばれます。比較される値のいずれかが NULL の場合、比較結果は UNKNOWN になります。探索条件は、条件の結果が TRUE の場合にのみ満たされます

AND　両方の条件が TRUE の場合、結合した条件は TRUE になります。条件のいずれかが FALSE の場合は FALSE、それ以外の場合は UNKNOWN になります。

condition1 **OR** *condition2*

OR　条件のいずれかが TRUE の場合、結合した条件は TRUE になります。両方の条件が FALSE の場合は FALSE、それ以外の場合は UNKNOWN になります。

NOT　*condition* が FALSE の場合、NOT 条件は TRUE です。*condition* が TRUE の場合は FALSE、*condition* が UNKNOWN の場合は UNKNOWN になります。

IS　*expression* が指定の *truth-value* (TRUE、FALSE、UNKNOWN のいずれか) と評価されれば条件は TRUE になります。それ以外の場合、値は FALSE です。

参照

- 「3 値的論理」 『SQL Anywhere サーバ - SQL リファレンス』
- 「比較演算子」 350 ページ
- 「Ultra Light の探索条件」 349 ページ

例

IS NULL 条件は、カラムに NULL 値が含まれる場合に満たされます。IS NOT NULL 演算子を使用すると、この条件は、カラムに NULL でない値が含まれる場合に満たされます。たとえば WHERE paid_date IS NULL は、IS NULL 条件です。

ALL 条件

ALL 条件は、比較演算子と組み合せて使用して、1 つの値をサブクエリが生成するデータ値と比較します。

構文

expression compare　[**NOT**] **ALL** (*subquery*)

パラメータ

compare:
= | > | < | >= | <= | <> | != | !< | !>

備考

Ultra Light は指定された比較演算子を使用して、テスト値を結果セットのデータ値と比較します。すべての比較の結果が TRUE になる場合、ALL テストは TRUE を返します。

参照

- 「サブクエリと ALL テスト」 『SQL Anywhere サーバ - SQL の使用法』
- 「比較演算子」 350 ページ

例

注文番号 2001 のすべての製品が出荷された後に受けた注文の注文 ID と顧客 ID を検索します。

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > ALL (
 SELECT ShipDate
 FROM SalesOrderItems
 WHERE ID=2001);
```

ANY 条件

ANY 条件は、比較演算子と組み合せて使用して、1 つの値をサブクエリが生成するデータ値のカラムと比較します。

構文 1

expression compare [**NOT**] **ANY (** *subquery* **)**

構文 2

expression = **ANY (** *subquery* **)**

パラメータ

compare:
= | > | < | >= | <= | <> | != | !< | !>

備考

Ultra Light は指定された比較演算子を使用して、テスト値をカラムのデータ値と比較します。いずれかの比較の結果が TRUE になる場合、ANY テストは TRUE を返します。

構文 1　*expression* がサブクエリ結果のいずれかの値と等しい場合は TRUE、expression が NULL ではなく、サブクエリから返されたいずれの値とも等しくない場合は FALSE です。*expression* が NULL 値の場合、サブクエリ結果にローがあれば ANY 条件は UNKNOWN です。サブクエリ結果にローがなければ、条件は必ず FALSE になります。

参照

- 「サブクエリと ANY テスト」 『SQL Anywhere サーバ - SQL の使用法』
- 「比較演算子」 350 ページ

例

注文番号 2005 の最初の製品が出荷された後に受けた注文の注文 ID と顧客 ID を検索します。

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > ANY (
 SELECT ShipDate
 FROM SalesOrderItems
 WHERE ID=2005);
```

BETWEEN 条件

包括的範囲を指定します。包括的範囲には、その範囲の間の値だけではなく、上限値と下限値も含まれます。

構文

expression [**NOT**] **BETWEEN** *start-expression* **AND** *end-expression*

備考

BETWEEN 条件は TRUE、FALSE、または UNKNOWN として評価できます。NOT キーワードがない場合、*expression* が *start-expression* と *end-expression* の間にあれば、条件は TRUE と評価されます。NOT キーワードを使用すると条件の意味が逆になりますが、UNKNOWN は変わりません。

BETWEEN 条件は、次の 2 つの不等式の組み合わせと等価です。

[**NOT**] (*expression* >= *start-expression*
AND *expression* <= *end-expression*)

例

$10 未満か、$15 を超える製品をすべてリストします。

```
SELECT Name, UnitPrice
FROM Products
WHERE UnitPrice NOT BETWEEN 10 AND 15;
```

EXISTS 条件

サブクエリがクエリ結果のローを生成するかどうかを調べます。

構文

[**NOT**] **EXISTS (** *subquery* **)**

備考

EXISTS 条件は、サブクエリ結果にローが少なくとも 1 つあれば TRUE で、ローがなければ FALSE です。EXISTS 条件は、UNKNOWN にはなりません。

EXISTS 条件の論理は、NOT EXISTS というフォームで否定できます。この場合、テストはサブクエリがローを返さない場合に TRUE を、ローを返す場合に FALSE を返します。

例

2001 年 7 月 13 日より後に注文した顧客をリストします。

```
SELECT GivenName, Surname
FROM Customers
WHERE EXISTS (
 SELECT *
 FROM SalesOrders
 WHERE (OrderDate > '2001-07-13') AND
      (Customers.ID = SalesOrders.CustomerID));
```

IN 条件

メイン・クエリの値からサブクエリの別の値を探索することでメンバシップをチェックします。

構文

expression [**NOT**] **IN**
{ **(** *subquery* **)** | **(** *value-expr*, ... **)** }

パラメータ

value-expr は、単一値をとる式です。これには、文字列、数字、日付、または他の任意の SQL データ型などがあります。

備考

NOT キーワードがない場合、IN 条件は次の規則に従って評価されます。

- *expression* が NULL でなく、少なくとも 1 つの値と等しい場合、TRUE です。
- *expression* が NULL で、値リストが空でない場合、または少なくとも 1 つの値が NULL で、*expression* が他の値のいずれとも等しくない場合、UNKNOWN です。
- *expression* が NULL で、*subquery* が値を返さない場合、または *expression* が NULL でなく、いずれの値も NULL でなく、*expression* がいずれの値とも等しくない場合、FALSE です。

IN 条件の論理は、NOT IN という形式で否定できます。

探索条件 *expression* **IN** (*values*) は、探索条件 *expression* = **ANY** (*values*) と同じです。探索条件 *expression* **NOT IN** (*values*) は、探索条件 *expression* <> **ALL** (*values*) と同じです。

例

カナダのオンタリオ州、マニトバ州、ケベック州に在住する顧客の会社名と状態を選択します。

```
SELECT CompanyName , Province
FROM Customers
WHERE State IN( 'ON', 'MB', 'PQ');
```

Ultra Light の演算子

演算子は値の計算に使用します。計算された値は、さらに上位の式でオペランドとして使用できます。

Ultra Light SQL では、次のタイプの演算子がサポートされています。

- 比較演算子は、1 つ (単項) または 2 つ (バイナリ) の比較オペランドを使用して評価し、結果を返します。比較の結果は、通常の 3 つの論理値、true、false、または unknown になります。
- 算術演算子は、浮動小数点、小数、整数の数値を評価し、結果セットを返します。
- 文字列演算子は、2 つの文字列を連結します。たとえば、"my" + "string" は文字列 "mystring" を返します。
- ビット処理演算子は、整数の内部表現内の特定のビットをオンまたはオフにします。
- 論理演算子は、探索条件を評価します。論理的評価の結果は、通常の 3 つの論理値、true、false、または unknown になります。

一般的な演算子の優先度が適用されます。

参照

- 「演算子の優先度」 358 ページ
- 「比較演算子」 350 ページ
- 「算術演算子」 356 ページ
- 「文字列演算子」 357 ページ
- 「ビット処理演算子」 357 ページ
- 「論理演算子」 351 ページ

算術演算子

算術演算子を使用すると、計算を実行できます。

expression + expression　加算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

expression - expression　減算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

-expression　反転。式が NULL 値の場合、結果は NULL 値になります。

expression * expression　乗算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

expression / expression　除算。いずれかの式が NULL の場合、または 2 番目の式が 0 の場合、結果は NULL になります。

expression % expression　剰余による、2 つの整数での除算の余り (整数) の算出。たとえば、21 を 11 で割ると商は 1、余りは 10 なので、21 % 11 = 10 になります。いずれかの式が NULL 値の場合、結果は NULL 値になります。

参照

- 「算術演算」 『SQL Anywhere サーバ - SQL の使用法』

文字列演算子

文字列演算子を使用すると、文字列を結合できます。ただし、LONGVARCHAR データ型と LONGBINARY データ型ではできません。

expression || expression　文字列連結 (2 つのパイプ記号)。いずれかの文字列が NULL 値の場合、連結には空文字列として扱われます。

expression + expression　代替の文字列連結。+ 連結演算子を使用する場合は、暗黙的データ変換を行わないで、必ずオペランドを文字データ型に明示設定してください。

たとえば、次のクエリは整数値 **579:** を返します。

```
SELECT 123 + 456;
```

これに対し、次のクエリは文字列 **123456:** を返します。

```
SELECT '123' + '456';
```

CAST または CONVERT 関数を使用すると、データ型を明示的に変換できます。

ビット処理演算子

ビット処理演算子は、2 つの式間でビット操作を実行します。Ultra Light では、次の演算子を整数データ型に対して使用できます。

演算子	説明
&	ビット処理 AND
|	ビット処理 OR
^	ビット処理排他的 OR
~	ビット処理 NOT

ビット処理演算子 &、|、~ は、論理演算子 AND、OR、NOT で代用することはできません。ビット処理演算子は、値のビット表現を使用して整数値に対して作用します。

例

次の文は指定するビットが設定されているローを選択します。

```
SELECT *
FROM tableA
WHERE (options & 0x0101) <> 0;
```

演算子の優先度

式における演算子の優先度は次のとおりです。カッコ内の式が最初に評価され、続いて乗算と除算、最後に加算と減算が評価されます。その後、文字列の連結が行われます。リストの最上部にある演算子から順に評価されます。

ヒント
演算子の優先度に依存しないで、演算の順序を明示的に指定してください。式で複数の演算子を使用する場合は、カッコを使用して演算の順序を明確にしてください。

1. 名前、関数、定数、IF 式、CASE 式
2. ()
3. 単項演算子 (1 つのオペランドを必要とする演算子) : +、-
4. ~
5. **&**, | , ^
6. *, /, **%**
7. +, -
8. ||
9. 比較 : >、<、<>、!=、<=、>=、[NOT] BETWEEN、[NOT] IN、[NOT] LIKE
10. 比較 : IS [NOT] TRUE、FALSE、UNKNOWN
11. NOT
12. AND
13. OR

Ultra Light の変数

Ultra Light アプリケーションでは、SQL 変数 (グローバル変数を含む) は使用できません。

Ultra Light の実行プラン

Ultra Light の実行プランは、クエリの実行時にテーブルとインデックスがアクセスされる方法を示します。Ultra Light には **query optimizer** があります。オプティマイザは、Ultra Light ランタイムの内部コンポーネントであり、クエリの効率的なプランを生成しようとします。オプティマイザは、テンポラリ・テーブルを使用して中間結果が格納されないようにしたり、クエリで 2 つのテーブルがジョインされるときに、テーブルの関連するサブセットだけがアクセスされるようにしたりします。

オプティマイザの上書き

オプティマイザでは、常に最も効率的なアクセス・プランが目標とされますが、特に多数の可能性がある複雑なクエリでは、この目標の達成は保証されません。極端な場合、OPTION (FORCE ORDER) 句をクエリに追加することによって、Ultra Light が選択するテーブル順序を上書きして、クエリに出現する順序でテーブルにアクセスすることもできます。**このオプションは一般的には使用しないことをおすすめします。**パフォーマンスが低い場合、通常は、適切なインデックスを作成して実行速度を上げるようにしてください。

パフォーマンスに関するヒント
クエリを使用してデータを更新しない場合は、クエリで FOR READ ONLY 句を指定してください。こうすると、パフォーマンスが向上することがあります。「Ultra Light SELECT 文」 507 ページを参照してください。

実行プランを確認する場合

次の情報が必要な場合に、Interactive SQL で実行プランを確認します。

- 結果を返すために使用されるインデックス。インデックス・スキャン・オブジェクトにテーブルの名前と、そのテーブルで使用されているインデックスが含まれます。
- 結果を返すためにテンポラリ・テーブルが使用されるかどうか。テンポラリ・テーブルは、Ultra Light のテンポラリ・ファイルに書き込まれます。「Ultra Light のテンポラリ・ファイル」 11 ページを参照してください。
- テーブルがジョインされる順序。この情報により、パフォーマンスへの影響を確認できます。
- クエリの実行が遅い理由、またはクエリの実行が遅くならないかどうか。

Ultra Light の実行プランの表示

開発時のサポートとして、Interactive SQL を使用して、準備文の実行方法をまとめた Ultra Light のプランを表示できます。テキスト・プランは、Interactive SQL のプラン・ビューワに表示されます。

Ultra Light では、実行プランはプランをテキスト形式で短くまとめたものです。他のプランのタイプはサポートされていません。ただし、プランは短く、情報が 1 行にまとめてあるので、簡単に比較できます。

♦ **プラン・ビューワで実行プランを表示するには、次の手順に従います。**

1. **[ツール] - [プラン・ビューワを開く]** を選択します。
2. **[SQL]** ウィンドウ枠で、クエリを入力します。
3. **[プランの取得]** をクリックして、指定した SQL 文のプランを生成します。

例

プラン・ビューワの下ウィンドウ枠にテキスト・プランが表示されます。

次の文を考えてみます。

```
SELECT I.inv_no, I.name, T.quantity, T.prod_no
FROM Invoice I, Transactions T
WHERE I.inv_no = T.inv_no;
```

この文により、下記のようなプランが生成されます。

```
join[scan(Invoice,primary),index-scan(Transactions,secondary)]
```

このプランは、primary というインデックスを使用して Invoice テーブルのすべてのローを読み込むことによって、ジョイン操作を実行することを示します。次に、Transaction テーブルの secondary というインデックスを使用して、inv_no カラムが一致するローのみを読み込みます。

参照

- 「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 の実行プランの解釈」 361 ページ

Ultra Light の実行プランの解釈

Ultra Light のプランは、クエリのアクセス方法をテキスト形式で短くまとめたものなので、テーブルのジョイン操作またはスキャン操作が実装される方法を理解している必要があります。

- **スキャン操作の場合**　1 つのオペランドで表されます。1 つのテーブルのみに適用され、インデックスが使用されます。テーブル名とインデックス名は操作名の後に丸カッコ ((と))で表されます。
- **その他の操作**　1 つまたは複数のオペランドで表されます。このオペランド自体もプランの場合があります。Ultra Light では、これらのオペランドは、角カッコ ([と]) で囲まれた、カンマ区切りのリストです。

操作リスト

次の表は、Ultra Light でサポートされている操作を示します。

操作	説明
count(*)	テーブルのローの数を数えます。
distinct[*plan*]	クエリの DISTINCT 部分を実装し、重複するローを比較、排除します。基本となるプランでローがソートされるときに使用され、重複、連続するローが排除されます。2 つの連続するローが一致する場合、最初のローだけが結果セットに追加されます。
dummy	処理は行われません。次の 2 つの場合にのみ使用されます。 ● FROM 句で DUMMY を指定した場合。 ● クエリに FROM 句がない場合。
filter[*plan*]	基本となるプランによって指定される各ローに探索条件を実行します。true と評価されたローだけが、結果セットの一部として転送されます。
group-by[*plan*]	GROUP BY の結果を集約したものを作成し、グループ化されたデータの複数の行をソートします。ローは、発生する順序で表示され、連続するローを比較してグループ化されます。
group-single[*plan*]	1 行だけが返されることがわかっている場合にのみ、GROUP BY の結果を集約したものを作成します。
keyset[*plan*]	テンポラリ・テーブル内のローの作成に使用されたローを記録し、Ultra Light で元の行を更新できるようにします。これらのローを更新しない場合は、クエリで FOR READ ONLY 句を使用してこの操作をなくします。
index-scan(*table-name*, *index-name*)	テーブルの一部だけを読み出します。開始ローはインデックスを使用して検索します。
join[*plan*, *plan*]	2 つのプラン間で内部ジョインを行います。
lojoin[*plan*, *plan*]	2 つのプラン間で左外部ジョインを行います。
like-scan(*table-name*, *index-name*)	テーブルの一部だけを読み出します。開始ローはインデックスを使用してパターン一致で検索します。
rowlimit[*plan*]	伝達されたローに対してローの制限処理を行います。ローの制限は、SELECT 文の TOP n または FIRST 句によって設定されます。

操作	説明
scan(*table-name*, *index-name*)	インデックスが示す順序でテーブル全体を読み出します。
sub-query[*plan*]	サブクエリの開始を示します。
temp[*plan*]	基本となるプラン内のローからテンポラリ・テーブルを作成します。Ultra Light では、基本となるローをソートする必要があり、そのために使用できるインデックスが見つからなかった場合にテンポラリ・テーブルが使用されます。 インデックスを追加して、テンポラリ・テーブルを不要にできます。ただし、インデックスを追加すると、インデックス対象のテーブル内のローの挿入または同期に必要な時間が長くなります。
union-all[*plan*, ..., *plan*]	基本となるプランで生成されるローに対して UNION ALL 操作を行います。

Ultra Light SQL 関数

目次

関数は、データベースの情報を返すために使用します。式を使用できるところでは必ず関数を使用できます。

マニュアルに特に指定がないかぎり、関数の引数のいずれかが NULL である場合は、NULL が返されます。

関数には、SQL 文と同じ構文の表記規則を使用します。構文の表記規則の全リストについては、「SQL 構文の表記規則」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

関数のタイプ

この項では、使用可能な関数をタイプ別に分類します。

Ultra Light では、SQL Anywhere 用に記載されている関数のサブセットをサポートしていますが、一部に違いがあります。

注意
特に明記しないかぎり、NULL をパラメータとして受け取る関数は、NULL を返します。

Ultra Light 集合関数

集合関数は、データベースから選択されたロー・グループのデータを要約します。グループは、SELECT 文の GROUP BY 句を使用して形成されます。集合関数は、select リストと、SELECT 文の HAVING 句と ORDER BY 句でのみ使用できます。

関数のリスト

次の集合関数を使用できます。

- 「AVG 関数 [集合]」 377 ページ
- 「COUNT 関数 [集合]」 388 ページ
- 「LIST 関数 [集合]」 414 ページ
- 「MAX 関数 [集合]」 419 ページ
- 「MIN 関数 [集合]」 420 ページ
- 「SUM 関数 [集合]」 451 ページ

Ultra Light データ型変換関数

データ型変換関数を使用して、引数のデータ型を他のデータ型に変換したり、変換が可能かどうかをテストしたりします。

関数のリスト

次のデータ型変換関数を使用できます。

- 「CAST 関数 [データ型変換]」 379 ページ
- 「CONVERT 関数 [データ型変換]」 384 ページ
- 「HEXTOINT 関数 [データ型変換]」 405 ページ
- 「INTTOHEX 関数 [データ型変換]」 409 ページ
- 「ISDATE 関数 [データ型変換]」 410 ページ

Ultra Light 日付と時刻関数

日付と時刻関数は、date データ型と time データ型の操作を行ったり、日付または時刻の情報を返したりします。

この章では、「日時」を日付、時刻、またはタイムスタンプを意味する用語として使用しています。特定のデータ型 DATETIME を示す場合は、DATETIME を使用します。

datetime データ型の詳細については、「Ultra Light のデータ型」 328 ページを参照してください。

関数のリスト

次の日付と時刻関数を使用できます。

日付の単位

日付関数の多くは、「日付の単位」で構成される日付を使用します。次の表は、使用できる日付の単位の値を示します。

日付の単位	省略形	値
Year	yy	1 ～ 9999
Quarter	qq	1 ～ 4
Month	mm	1 ～ 12
Week	wk	1 ～ 54。日曜日を週の最初の日とします。
Day	dd	1 ～ 31
Dayofyear	dy	1 ～ 36
Weekday	dw	1 ～ 7 (日曜日 = 1、...、土曜日 = 7)
Hour	hh	0 ～ 23
Minute	mi	0 ～ 59
Second	ss	0 ～ 59
Millisecond	ms	0 ～ 999
Calyearofweek	cyr	整数。その週が何年に開始したかを示します。週に年の最初の数日が含まれている場合は、その年の最初の曜日に応じて、週の最初の日が前年になる場合があります。年の最初の曜日が月曜日～木曜日の場合は、前年に属する日がその年に含まれることはありませんが、年の最初の曜日が金曜日～日曜日の場合、年の最初の週はその年の最初の月曜日から開始します。
Calweekofyear	cwk	1 ～ 54。指定した日付がその年の第何週であるかを示します。
Caldayofweek	cdw	1 ～ 7。(月曜日 = 1、...、日曜日 = 7)

Ultra Light その他の関数

その他の関数は、算術式、文字列式、日付／時刻式、他の関数の戻り値に対して操作を実行します。

関数のリスト

次の各種関数を使用できます。

- 「ARGN 関数 [その他]」 373 ページ
- 「COALESCE 関数 [その他]」 384 ページ
- 「EXPLANATION 関数 [その他]」 402 ページ
- 「GREATER 関数 [その他]」 404 ページ
- 「IFNULL 関数 [その他]」 408 ページ
- 「ISNULL 関数 [その他]」 411 ページ
- 「LESSER 関数 [その他]」 414 ページ
- 「NEWID 関数 [その他]」 427 ページ
- 「NULLIF 関数 [その他]」 428 ページ

Ultra Light 数値関数

数値関数は、数値データ型の算術演算を実行したり、数値情報を返したりします。

関数のリスト

次の数値関数を使用できます。

- 「ABS 関数 [数値]」 372 ページ
- 「ACOS 関数 [数値]」 373 ページ
- 「ASIN 関数 [数値]」 375 ページ
- 「ATAN 関数 [数値]」 375 ページ
- 「ATAN2 関数 [数値]」 376 ページ
- 「CEILING 関数 [数値]」 380 ページ
- 「COS 関数 [数値]」 387 ページ
- 「COT 関数 [数値]」 387 ページ
- 「DEGREES 関数 [数値]」 399 ページ
- 「EXP 関数 [数値]」 402 ページ
- 「FLOOR 関数 [数値]」 403 ページ
- 「LOG 関数 [数値]」 416 ページ
- 「LOG10 関数 [数値]」 417 ページ
- 「MOD 関数 [数値]」 424 ページ
- 「PI 関数 [数値]」 431 ページ
- 「POWER 関数 [数値]」 432 ページ
- 「RADIANS 関数 [数値]」 433 ページ
- 「REMAINDER 関数 [数値]」 434 ページ
- 「ROUND 関数 [数値]」 438 ページ
- 「SIGN 関数 [数値]」 442 ページ
- 「SIN 関数 [数値]」 443 ページ
- 「SQRT 関数 [数値]」 446 ページ
- 「TAN 関数 [数値]」 454 ページ
- 「TRUNCNUM 関数 [数値]」 456 ページ

Ultra Light 文字列関数

文字列関数は、文字列に対して変換、抽出、操作の演算を実行したり、文字列に関する情報を返したりします。

マルチバイト文字セットを操作する場合は、使用する関数が文字とバイトのどちらの情報を返すかを十分に確認してください。

関数のリスト

次の文字列関数を使用できます。

Ultra Light システム関数

システム関数は、システム情報を返します。

関数のリスト

Ultra Light では、次のシステム関数を使用できます。

- 「DB_PROPERTY 関数 [システム]」 398 ページ
- 「ML_GET_SERVER_NOTIFICATION [システム]」 423 ページ
- 「SYNC_PROFILE_OPTION_VALUE 関数 [システム]」 453 ページ

SQL 関数 (A ～ D)

関数を 1 つずつリストし、その右側に関数のタイプ (数値、文字など) を示します。

特定のタイプのすべての関数へのリンクについては、「関数のタイプ」 366 ページを参照してください。

参照

- 「SQL 関数 (E ～ O)」 『SQL Anywhere サーバ - SQL リファレンス』
- 「SQL 関数 (P ～ Z)」 『SQL Anywhere サーバ - SQL リファレンス』

ABS 関数 [数値]

数値式の絶対値を返します。

構文

ABS(*numeric-expression* **)**

パラメータ

- **numeric-expression** 絶対値が返される数値。

戻り値

数値式の絶対値。

数値式データ型	戻り値
INT	INT
FLOAT	FLOAT
DOUBLE	DOUBLE
NUMERIC	NUMERIC

標準と互換性

- **SQL/2003** コア SQL に含まれない SQL 基本機能。

例

次の文は、値 66 を返します。

```
SELECT ABS( -66 );
```


ACOS 関数 [数値]

数値式のアークコサインをラジアン単位で返します。

構文

ACOS(*numeric-expression* **)**

パラメータ

- **numeric-expression**　角度のコサイン。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「ASIN 関数 [数値]」 375 ページ
- 「ATAN 関数 [数値]」 375 ページ
- 「ATAN2 関数 [数値]」 376 ページ
- 「COS 関数 [数値]」 387 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、0.52 のアークコサイン値を返します。

```
SELECT ACOS( 0.52 );
```

ARGN 関数 [その他]

引数リストから選択された引数を返します。

構文

ARGN(*integer-expression*, *expression* [, ...] **)**

パラメータ

- **integer-expression**　式リスト内での引数の位置。
- **expression**　関数に渡される任意のデータ型の式。すべて同じデータ型の式を指定してください。

戻り値

integer-expression の値に n を使用すると、残りの引数リストから n 番目の引数 (1 から開始) が返されます。

備考

式のデータ型は任意ですが、すべて同じデータ型にしてください。integer-expression は、1 からリスト内の式の数までの範囲内で指定してください。範囲外の値を指定すると、NULL が返されます。複数の式は、カンマで区切って指定します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 6 を返します。

```
SELECT ARGN( 6, 1,2,3,4,5,6 );
```

ASCII 関数 [文字列]

文字列式の最初のバイトの ASCII 値を整数で返します。

構文

ASCII(*string-expression* **)**

パラメータ

- **string-expression**　文字列。

戻り値

SMALLINT

備考

文字列が空の場合は、0 を返します。リテラル文字列は、引用符で囲んで指定します。データベース文字セットがマルチバイトであり、パラメータ文字列の最初の文字が複数バイトから構成される場合、結果は NULL です。

参照

- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 90 を返します。

```
SELECT ASCII( 'Z' );
```


ASIN 関数 [数値]

数値のアークサインをラジアン単位で返します。

構文

ASIN(*numeric-expression* **)**

パラメータ

- **numeric-expression**　角度のサイン。

戻り値

DOUBLE

備考

SIN 関数と ASIN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「ACOS 関数 [数値]」 373 ページ
- 「ATAN 関数 [数値]」 375 ページ
- 「ATAN2 関数 [数値]」 376 ページ
- 「SIN 関数 [数値]」 443 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、0.52 のアークサイン値を返します。

```
SELECT ASIN( 0.52 );
```

ATAN 関数 [数値]

数値のアークタンジェントをラジアン単位で返します。

構文

ATAN(*numeric-expression* **)**

備考

ATAN 関数と TAN 関数は逆変換の演算です。

パラメータ

- **numeric-expression**　角度のタンジェント。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「ACOS 関数 [数値]」 373 ページ
- 「ASIN 関数 [数値]」 375 ページ
- 「ATAN2 関数 [数値]」 376 ページ
- 「TAN 関数 [数値]」 454 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、0.52 のアークタンジェント値を返します。

```
SELECT ATAN( 0.52 );
```

ATAN2 関数 [数値]

2 つの数の比率のアークタンジェントをラジアン単位で返します。

構文

ATAN2 (*numeric-expression-1*, *numeric-expression-2* **)**

パラメータ

- **numeric-expression-1** アークタンジェントが計算される比率の分子。
- **numeric-expression-2** アークタンジェントが計算される比率の分母。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「ACOS 関数 [数値]」 373 ページ
- 「ASIN 関数 [数値]」 375 ページ
- 「ATAN 関数 [数値]」 375 ページ
- 「TAN 関数 [数値]」 454 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、比率 0.52 ～ 0.60 のアークタンジェント値を返します。

```
SELECT ATAN2( 0.52, 0.60 );
```

AVG 関数 [集合]

対象となるロー・セットの、数値式の平均値またはユニークな値からなるセットの平均値を計算します。

構文 1

AVG(*numeric-expression* | **DISTINCT** *numeric-expression* **)**

パラメータ

- **numeric-expression**　ロー・セットで平均値を計算する対象の式。
- **DISTINCT 句**　入力中で一意の数値の平均を計算します。

戻り値

グループにローが含まれていない場合は、NULL 値を返します。

引数が DOUBLE の場合は DOUBLE を、それ以外の場合は NUMERIC を返します。

備考

この平均値には、*numeric-expression* が NULL 値のローは含まれません。

参照

- 「SUM 関数 [集合]」 451 ページ
- 「COUNT 関数 [集合]」 388 ページ

標準と互換性

- **SQL/2003**　コア機能。

例

次の文は、値 49988.623200 を返します。

```
SELECT AVG( Salary ) FROM Employees;
```

次の文は、Products テーブルの製品価格の平均を返します。

```
SELECT AVG( DISTINCT UnitPrice ) FROM Products;
```

BYTE_LENGTH 関数 [文字列]

文字列のバイト数を返します。

構文

BYTE_LENGTH(*string-expression* **)**

パラメータ

- **string-expression**　長さが計算される文字列。

戻り値

INT

備考

string-expression の後続空白スペースは、返される長さに含まれます。

NULL 文字列の戻り値は NULL です。

マルチバイト文字セットの文字列の場合、BYTE_LENGTH の値は、CHAR_LENGTH で返される文字数と異なることがあります。

参照

- 「CHAR_LENGTH 関数 [文字列]」 383 ページ
- 「DATALENGTH 関数 [システム]」 389 ページ
- 「LENGTH 関数 [文字列]」 413 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 12 を返します。

```
SELECT BYTE_LENGTH( 'Test Message' );
```

BYTE_SUBSTR 関数 [文字列]

文字列の部分文字列を返します。部分文字列は、文字ではなくバイトを使用して計算されます。

構文

BYTE_SUBSTR(*string-expression*, *start* [, *length*] **)**

パラメータ

- **string-expression**　部分文字列が取得される文字列。

- **start**　部分文字列の開始位置を示す整数式。正の整数の場合、文字列の先頭から開始します。先頭文字の位置は 1 です。負の整数の場合、文字列の末尾から開始します。最後の文字の位置は -1 です。
- **length**　部分文字列の長さを示す整数式。正の *length* は、取得するバイト数を開始位置から**開始する**ことを指定します。負の *length* は、開始位置から最大 *length* バイト左側のバイトを返します。

戻り値

返される値は *string-expression* の型によって異なります。また、戻り値が LONG になるかどうかは、指定した引数によって決まります。たとえば、長さが 32K 未満の定数を指定した場合、LONG は返されません。

BINARY

VARCHAR

NVARCHAR

備考

length を指定すると、部分文字列は指定したバイト数に制限されます。*start* と *length* には、正または負の値を指定できます。負の数と正の数を適切に組み合わせて使用すると、文字列の先頭または末尾のどちらからでも部分文字列を取得できます。

start が 0 で length が負でない場合は、1 の *start* 値が使用されます。*start* が 0 で *length* が負の場合は、-1 の start 値が使用されます。

参照

- 「SUBSTRING 関数 [文字列]」 449 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 Test を返します。

```
SELECT BYTE_SUBSTR( 'Test Message', 1, 4 );
```

CAST 関数 [データ型変換]

指定されたデータ型に変換した式の値を返します。

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。これらの関数の使用の詳細については、「16 進値との変換」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

構文

CAST(*expression* **AS** *datatype* **)**

パラメータ

- **expression**　変換される式。
- **data type**　ターゲットのデータ型。

戻り値

指定されたデータ型。

備考

文字列型の長さを指定しない場合は、データベース・サーバによって適切な長さが選択されます。DECIMAL 変換で精度も位取りも指定しない場合は、データベース・サーバによって適切な値が選択されます。

参照

- 「CONVERT 関数 [データ型変換]」 384 ページ
- 「LEFT 関数 [文字列]」 412 ページ

標準と互換性

- **SQL/2003**　コア機能。

例

次の関数は、文字列を日付として使用することを保証します。

```
SELECT CAST( '2000-10-31' AS DATE );
```

式 1 + 2 の値を計算し、その結果を 1 文字の文字列にキャストします。

```
SELECT CAST( 1 + 2 AS CHAR );
```

次のように CAST 関数を使用して、文字列を短縮できます。

```
SELECT CAST ( 'Surname' AS CHAR(5) );
```

CEILING 関数 [数値]

指定した値以上になる最初の整数を返します。正の数値の場合、この処理は丸めとも呼ばれています。

構文

CEILING(*numeric-expression* **)**

パラメータ

- **numeric-expression**　切り上げ値を計算する対象の数値。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「FLOOR 関数 [数値]」 403 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 60 を返します。

```
SELECT CEILING( 59.84567 );
```

CHAR 関数 [文字列]

数値の ASCII 値を持つ文字を返します。

構文

CHAR(*integer-expression* **)**

パラメータ

- **integer-expression** ASCII 文字に変換される数値。0 ～ 255 の範囲内の数を指定します。

戻り値

VARCHAR

備考

返される文字は、バイナリ・ソート順に従って、現在のデータベース側文字セットの指定した数値式に対応しています。

整数式の値が 255 より大きいか、0 より小さい場合、CHAR は NULL を返します。

参照

- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 Y を返します。

```
SELECT CHAR( 89 );
```

CHARINDEX 関数 [文字列]

ある文字列内でのもう 1 つの文字列の位置を返します。

構文

CHARINDEX(*string-expression-1*, *string-expression-2* **)**

パラメータ

- **string-expression-1**　検索する文字列。
- **string-expression-2**　検索される文字列。

戻り値

INT

備考

string-expression-1 の先頭文字の位置を 1 とします。検索される文字列内に、検索する文字列のインスタンスが複数存在する場合、CHARINDEX 関数は最初のインスタンスの位置を返します。

検索される文字列内に、検索する文字列が存在しない場合、CHARINDEX 関数は 0 を返します。

参照

- 「SUBSTRING 関数 [文字列]」 449 ページ
- 「REPLACE 関数 [文字列]」 435 ページ
- 「LOCATE 関数 [文字列]」 415 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、姓に K という文字が含まれる場合にのみ、Employees テーブルの Surname カラムと GivenName カラムから姓名を返します。

```
SELECT Surname, GivenName
FROM Employees
WHERE CHARINDEX( 'K', Surname ) = 1;
```

返された結果：

Surname	GivenName
Klobucher	James
Kuo	Felicia

Surname	GivenName
Kelly	Moira

CHAR_LENGTH 関数 [文字列]

文字列の文字数を返します。

構文

CHAR_LENGTH (*string-expression* **)**

パラメータ

- **string-expression**　長さが計算される文字列。

戻り値

INT

備考

後続空白スペースは、返される長さに含まれます。

NULL 文字列の戻り値は NULL です。

マルチバイト文字セットの文字列の場合、CHAR_LENGTH 関数で返される値は、BYTE_LENGTH 関数で返されるバイト数と異なることがあります。

> **注意**
> データ型が CHAR、VARCHAR、LONG VARCHAR の場合、CHAR_LENGTH 関数と LENGTH 関数を使用すると同じ結果が得られます。ただし、BINARY データ型とビット配列データ型には LENGTH 関数を使用します。

参照

- 「BYTE_LENGTH 関数 [文字列]」 378 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　コア機能。

例

次の文は、値 8 を返します。

```
SELECT CHAR_LENGTH( 'Chemical' );
```

COALESCE 関数 [その他]

リストの中から NULL でない最初の式を返します。この関数は ISNULL 関数と同じです。

構文

COALESCE(*expression*, *expression* [, ...] **)**

パラメータ

- **expression**　任意の式。

  2 つ以上の式を関数に渡します。すべての式は比較可能であることが必要です。

戻り値

任意の値

備考

結果が NULL になるのはすべての引数が NULL の場合のみです。

このパラメータにはスカラ型を指定できますが、同じ型を指定する必要はありません。

この関数がデータベース・サーバで処理される方法の詳細については、「ISNULL 関数 [その他]」 411 ページを参照してください。

参照

- 「ISNULL 関数 [その他]」 411 ページ

標準と互換性

- **SQL/2003**　コア機能。

例

次の文は、値 34 を返します。

```
SELECT COALESCE( NULL, 34, 13, 0 );
```

CONVERT 関数 [データ型変換]

指定されたデータ型に変換した式を返します。

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。これらの関数の使用の詳細については、「16 進値との変換」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

構文

CONVERT(*datatype*, *expression* [, *format-style*] **)**

パラメータ

- **datatype**　変換後の式のデータ型。
- **expression**　変換される式。
- **format-style**　出力値に適用されるスタイル・コード。このパラメータを使用するのは、文字列を日付または時刻のデータ型に変換するとき、またはその反対方向に変換するときです。次の表は、サポートされるスタイル・コードと、そのスタイル・コードで作成される出力形式の表現です。スタイル・コードは世紀を出力形式に含めるかどうかによって 2 つのカラムに分けられます (たとえば、06 と 2006 など)。

100 以上の位なし (yy) のスタイル・コード	100 以上の位あり (yyyy) のスタイル・コード	出力形式
-	0 または 100	Mmm dd yyyy hh:nnAA
1	101	mm/dd/yy[yy]
2	102	[yy]yy.mm.dd
3	103	dd/mm/yy[yy]
4	104	dd.mm.yy[yy]
5	105	dd-mm-yy[yy]
6	106	dd Mmm yy[yy]
7	107	Mmm dd, yy[yy]
8	108	hh:nn:ss
-	9 または 109	Mmm dd yyyy hh:nn:ss:sssAA
10	110	mm-dd-yy[yy]
11	111	[yy]yy/mm/dd
12	112	[yy]yymmdd
-	13 または 113	dd Mmm yyyy hh:nn:ss:sss (24 時間表記、ヨーロッパのデフォルト、ミリ秒、4 桁の年)
-	14 または 114	hh:nn:ss:sss (24 時間表記)
-	20 または 120	yyyy-mm-dd hh:nn:ss (24 時間表記、ODBC 標準、4 桁の年)

100 以上の位なし (yy) のスタイル・コード	100 以上の位あり (yyyy) のスタイル・コード	出力形式
-	21 または 121	yyyy-mm-dd hh:nn:ss.sss (24 時間表記、ODBC 標準、ミリ秒、4 桁の年)

戻り値

指定されたデータ型。

備考

format-style 引数を指定しない場合は、スタイル・コード 0 が使用されます。

各出力記号 (Mmm など) が出力するスタイルの詳細については、「Ultra Light date_format 作成パラメータ」 194 ページを参照してください。

参照

- 「CAST 関数 [データ型変換]」 379 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、フォーマット・スタイルの使用方法を示します。

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 104 ) FROM SalesOrders;
```

OrderDate
16.03.2000
20.03.2000
23.03.2000
25.03.2000
...

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 7 ) FROM SalesOrders;
```

OrderDate
Mar 16, 00
Mar 20, 00

OrderDate
Mar 23, 00
Mar 25, 00
...

次の文は、整数への変換を示し、値 5 を返します。

```
SELECT CONVERT( integer, 5.2 );
```

COS 関数 [数値]

引数で与えられた角度のコサインをラジアン単位で返します。

構文

COS(*numeric-expression* **)**

パラメータ

- **numeric-expression**　角度 (ラジアン)。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

参照

- 「ACOS 関数 [数値]」 373 ページ
- 「COT 関数 [数値]」 387 ページ
- 「SIN 関数 [数値]」 443 ページ
- 「TAN 関数 [数値]」 454 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、角度 0.52 ラジアンのコサイン値を返します。

```
SELECT COS( 0.52 );
```

COT 関数 [数値]

引数で与えられた角度のコタンジェントをラジアン単位で返します。

構文

COT(*numeric-expression* **)**

パラメータ

- **numeric-expression**　角度 (ラジアン)。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

参照

- 「COS 関数 [数値]」 387 ページ
- 「SIN 関数 [数値]」 443 ページ
- 「TAN 関数 [数値]」 454 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、0.52 のコタンジェント値を返します。

```
SELECT COT( 0.52 );
```

COUNT 関数 [集合]

指定されたパラメータに従って、グループのロー数をカウントします。

構文 1

COUNT(
*
| *expression*
| **DISTINCT** *expression*
)

パラメータ

- *****　各グループの中のロー数を返します。
- **expression**　ローの数を返す式。
- **DISTINCT 式**　排他ローの数を返す式。

戻り値

INT

備考

値が NULL 値のローは、カウントに含まれません。

参照

- 「AVG 関数 [集合]」 377 ページ
- 「SUM 関数 [集合]」 451 ページ

標準と互換性

- **SQL/2003**　コア機能。構文 2 は機能 T611 です。

例

次の文は、ユニークな各都市と、その都市の値を持つロー数を返します。

```
SELECT City, COUNT( * ) FROM Employees GROUP BY City;
```

DATALENGTH 関数 [システム]

式の結果に必要な基本となる記憶領域の長さ (バイト) を返します。

構文

DATALENGTH(*expression* **)**

パラメータ

- **expression**　通常は、カラム名です。*expression* が文字列定数の場合は、引用符で囲みます。

戻り値

UNSIGNED INT

備考

DATALENGTH 関数の戻り値は次のとおりです。

データ型	DATALENGTH
SMALLINT	2
INTEGER	4
DOUBLE	8
CHAR	データの長さ
BINARY	データの長さ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、CompanyName カラムで最も長い文字列の長さ 27 を返します。

```
SELECT MAX( DATALENGTH( CompanyName ) )
FROM Customers;
```

次の文は、文字列 '8sdofinsv8s7a7s7gehe4h' の長さを返します。

```
SELECT DATALENGTH( '8sdofinsv8s7a7s7gehe4h' );
```

DATE 関数 [日付と時刻]

式を日付に変換し、時間、分、秒を削除します。

日付フォーマットの解釈の制御については、「Ultra Light date_order 作成パラメータ」 197 ページを参照してください。

構文

DATE(*expression* **)**

戻り値

DATE

パラメータ

- **expression**　日付フォーマットに変換される値。通常、文字列。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 1999-01-02 を日付として返します。

```
SELECT DATE( '1999-01-02 21:20:53' );
```

次の文は、SYSOBJECT システム・ビューにリストされるすべてのオブジェクトについて、作成日を返します。

```
SELECT DATE( creation_time ) FROM SYSOBJECT;
```

DATEADD 関数 [日付と時刻]

日付にいくつかの日付の単位を加算した日付を返します。

構文

DATEADD(*date-part*, *numeric-expression*, *date-expression* **)**

date-part :
year
| **quarter**
| **month**
| **week**

| **day**
| **dayofyear**
| **hour**
| **minute**
| **second**
| **millisecond**

パラメータ

- **date-part**　日付に加算される日付の単位。日付の単位の詳細については、「日付の単位」 367 ページを参照してください。
- **numeric-expression**　日付に加算される日付の単位の数。*numeric-expression* には任意の数値型を指定できますが、値はトランケートされて整数になります。
- **date-expression**　変更される日付。

戻り値

TIMESTAMP

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 1995-11-02 00:00:000.000 を返します。

```
SELECT DATEADD( month, 102, '1987/05/02' );
```

DATEDIFF 関数 [日付と時刻]

2 つの日付間の期間を返します。

構文

DATEDIFF(*date-part*, *date-expression-1*, *date-expression-2* **)**

date-part :
year
| **quarter**
| **month**
| **week**
| **day**
| **dayofyear**
| **hour**
| **minute**
| **second**
| **millisecond**

パラメータ

- **date-part**　期間を測定する日付の単位を指定します。

上記の日付オブジェクトから 1 つを選択します。日付の単位の完全なリストについては、「日付の単位」 367 ページを参照してください。

- **date-expression-1**　期間の開始日。この値を *date-expression-2* から引いて、2 つの引数の間に存在する *date-part* の数を返します。
- **date-expression-2**　期間の終了日。この値から *date-expression-1* を引いて、2 つの引数の間に存在する *date-part* の数を返します。

戻り値

INT

備考

この関数は、指定した 2 つの日付間に存在する日付の単位の数を計算します。結果は、日付の単位の数を表す、(date2 - date1) と同値の符号付き整数です。

DATEDIFF 関数の結果が日付の単位の整数倍でない場合、結果は丸められずに切り捨てになります。

day を日付の単位として使用する場合、DATEDIFF 関数は指定された 2 つの時刻の間の午前 0 時の回数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

month を日付の単位として使用する場合、DATEDIFF 関数は 2 つの日付の間に存在する月の初日の数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

week を日付の単位として使用する場合、DATEDIFF 関数は 2 つの日付の間に存在する日曜日の数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

より短い時間単位のために、オーバフロー値が用意されています。

- **ミリ秒**　24 日
- **秒**　68 年
- **分**　4083 年
- **その他**　オーバフロー制限なし

これらの制限値を超えると、この関数はオーバフロー・エラーを返します。

標準と互換性

- **SQL/2003**　Transact-SQL 拡張。

例

次の文は、1 を返します。

```
SELECT DATEDIFF( hour, '4:00AM', '5:50AM' );
```

次の文は、102 を返します。

```
SELECT DATEDIFF( month, '1987/05/02', '1995/11/15' );
```

次の文は、0 を返します。

```
SELECT DATEDIFF( day, '00:00', '23:59' );
```

次の文は、4 を返します。

```
SELECT DATEDIFF( day,
  '1999/07/19 00:00',
  '1999/07/23 23:59' );
```

次の文は、0 を返します。

```
SELECT DATEDIFF( month, '1999/07/19', '1999/07/23' );
```

次の文は、1 を返します。

```
SELECT DATEDIFF( month, '1999/07/19', '1999/08/23' );
```

DATEFORMAT 関数 [日付と時刻]

日付式を表す文字列を、指定したフォーマットで返します。

構文

DATEFORMAT(*datetime-expression*, *string-expression* **)**

パラメータ

- **datetime-expression**　変換される日時。
- **string-expression**　変換後の日付のフォーマット。

  日付フォーマットの説明については、「Ultra Light date_format 作成パラメータ」 194 ページを参照してください。

戻り値

VARCHAR

備考

string-expression には、許容される任意の日付フォーマットを使用できます。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 Jan 01, 1989 を返します。

```
SELECT DATEFORMAT( '1989-01-01', 'Mmm dd, yyyy' );
```

DATENAME 関数 [日付と時刻]

日時の値の特定部分の名前 (月の名前 "June" など) を文字列で返します。

構文

DATENAME(*date-part*, *date-expression* **)**

パラメータ

- **date-part**　名前が返される日付の単位。

  使用可能な日付の単位の完全なリストについては、「日付の単位」 367 ページを参照してください。

- **date-expression**　日付の単位名が返される日付。要求する *date-part* が含まれた日付を指定してください。

戻り値

VARCHAR

備考

結果が数値 (23 日など) の場合でも、DATENAME 関数は文字列を返します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 May を返します。

```
SELECT DATENAME( month, '1987/05/02' );
```

DATEPART 関数 [日付と時刻]

日時の値の一部の値を返します。

構文

DATEPART(*date-part*, *date-expression* **)**

パラメータ

- **date-part**　名前が返される日付の単位。

  使用可能な日付の単位の完全なリストについては、「日付の単位」 367 ページを参照してください。

- **date-expression**　値が返される日付。

戻り値

INT

備考

date-part フィールドが含まれた日付を指定してください。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 5 を返します。

```
SELECT DATEPART( month , '1987/05/02' );
```

次の例では、TableStatistics テーブルを作成し、SalesOrders テーブルに格納されている年ごとの注文の総数を挿入します。

```
CREATE TABLE TableStatistics (
  ID INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  Year INT,
  NumberOrders INT );
INSERT INTO TableStatistics ( Year, NumberOrders )
  SELECT DATEPART( Year, OrderDate ), COUNT(*)
  FROM SalesOrders
  GROUP BY DATEPART( Year, OrderDate );
```

DATETIME 関数 [日付と時刻]

式をタイムスタンプに変換します。

構文

DATETIME(*expression* **)**

パラメータ

- **expression**　変換される式。通常は文字列です。

戻り値

TIMESTAMP

備考

数値を変換しようとするとエラーが返ります。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 1998-09-09 12:12:12.000 のタイムスタンプを返します。

```
SELECT DATETIME( '1998-09-09 12:12:12.000' );
```

DAY 関数 [日付と時刻]

1 ～ 31 の整数を返します。

構文

DAY(*date-expression* **)**

パラメータ

- **date-expression**　日付。

戻り値

SMALLINT

備考

日付の月日に対応する整数 1 ～ 31。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 12 を返します。

```
SELECT DAY( '2001-09-12' );
```

DAYNAME 関数 [日付と時刻]

日付から曜日の名前を返します。

構文

DAYNAME(*date-expression* **)**

パラメータ

- **date-expression**　日付。

戻り値

VARCHAR

備考

返される曜日名は日本語で、日曜日、月曜日、火曜日、水曜日、木曜日、金曜日、土曜日です。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値土曜日を返します。

```
SELECT DAYNAME ( '1987/05/02' );
```

DAYS 関数 [日付と時刻]

日付を評価する関数。詳細については、この関数の使用法を参照してください。

構文 1：整数

DAYS([*datetime-expression*,] *datetime-expression* **)**

構文 2：タイムスタンプ

DAYS(*datetime-expression*, *integer-expression* **)**

パラメータ

- **datetime-expression**　日付と時刻。
- **integer-expression**　*datetime-expression* に加算する日数。*integer-expression* が負の場合、タイムスタンプから適切な日数が引かれます。整数式を指定する場合は、*datetime-expression* を日付またはタイムスタンプとして明示的にキャストしてください。

  データ型のキャストの詳細については、「CAST 関数 [データ型変換]」 379 ページを参照してください。

戻り値

datetime-expression を 2 つ指定した場合は INT。

2 番目の引数が整数の場合は TIMESTAMP。

備考

この関数の動作は、指定内容によって変わります。

- 1 つの日付を指定すると、0000-02-29 からの日数を返します。

  > **注意**
  > 0000-02-29 は実際の日付を指すための値ではありません。日付アルゴリズムで使用される日付です。

- 2 つの日付を指定すると、2 つの日付の間の日数を整数で返します。代わりに、DATEDIFF 関数を使用します。
- この関数で 1 つの日付と整数を指定すると、指定した日付に、指定した整数の日数が加算されます。代わりに、DATEADD 関数を使用します。

この関数では、時間、分、秒が無視されます。

参照

- 「DATEDIFF 関数 [日付と時刻]」 391 ページ
- 「DATEADD 関数 [日付と時刻]」 390 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、整数 729889 を返します。

```
SELECT DAYS( '1998-07-13 06:07:12' );
```

次の文は、整数値 -366 を返します。このことは、2 番目の日付が最初の日付より 366 日前の日付であることを示します。2 つ目の例 (DATEDIFF) の使用をおすすめします。

```
SELECT DAYS( '1998-07-13 06:07:12',
        '1997-07-12 10:07:12' );

SELECT DATEDIFF( day,
  '1998-07-13 06:07:12',
  '1997-07-12 10:07:12' );
```

次の文は、タイムスタンプ 1999-07-14 00:00:00.000 を返します。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT DAYS( CAST('1998-07-13' AS DATE ), 366 );

SELECT DATEADD( day, 366, '1998-07-13' );
```

DB_PROPERTY 関数 [システム]

指定したプロパティの値を返します。

構文

DB_PROPERTY(*property-name* **)**

パラメータ

- **property-name** データベース・プロパティ名。

戻り値

VARCHAR

備考

文字列を返します。

Ultra Light でオプションを設定するには、SET OPTION 文、またはコンポーネントの API に固有のデータベース・オプション設定メソッドを使用します。

参照

- 「Ultra Light SET OPTION 文」 509 ページ
- Ultra Light C++：「SetDatabaseOption 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light.NET：「SetDatabaseOption メソッド」 『Ultra Light - .NET プログラミング』

例

次の文は、現在のデータベースのページ・サイズをバイト単位で返します。

```
SELECT DB_PROPERTY( 'page_size');
```

DEGREES 関数 [数値]

数値をラジアンから度数に変換します。

構文

DEGREES(*numeric-expression* **)**

パラメータ

- **numeric-expression**　角度 (ラジアン)。

戻り値

numeric-expression で指定される角度を返します。

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。パラメータが NULL 値の場合、結果は NULL 値になります。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 29.79380534680281 を返します。

```
SELECT DEGREES( 0.52 );
```

DIFFERENCE 関数 [文字列]

2 つの文字列式の SOUNDEX 値の差を返します。

構文

DIFFERENCE (*string-expression-1*, *string-expression-2* **)**

パラメータ

- **string-expression-1**　最初の SOUNDEX 引数。
- **string-expression-2**　2 番目の SOUNDEX 引数。

戻り値

SMALLINT

備考

DIFFERENCE 関数は 2 つの文字列の SOUNDEX 値を比較し、値の類似性を評価し、0 ～ 4 の値を返します。最も一致する場合は 4 です。

この関数は常に何らかの値を返します。結果が NULL になるのは引数の 1 つが NULL の場合のみです。

参照

- 「SOUNDEX 関数 [文字列]」 444 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 3 を返します。

```
SELECT DIFFERENCE( 'test', 'chest' );
```

DOW 関数 [日付と時刻]

指定した日付の曜日を表す 1 ～ 7 の数を返します (日曜日 = 1、月曜日 = 2、以下同様)。

構文

DOW(*date-expression* **)**

パラメータ

- **date-expression**　評価する日付。

戻り値

SMALLINT

備考

DOW 関数は、first_day_of_week データベース・オプションで指定された値の影響を受けません。たとえば、first_day_of_week が月曜日に設定されている場合でも、DOW 関数は月曜日に 2 を返します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 5 を返します。

```
SELECT DOW( '1998-07-09' );
```

次の文は、Employees テーブルに問い合わせ、週の曜日を表す数として StartDate を返します。

```
SELECT DOW( StartDate ) FROM Employees;
```

SQL 関数 (E ～ O)

関数を 1 つずつリストし、その右側に関数のタイプ (数値、文字など) を示します。

特定のタイプのすべての関数へのリンクについては、「関数のタイプ」 366 ページを参照してください。

参照

- 「SQL 関数 (A ～ D)」 『SQL Anywhere サーバ - SQL リファレンス』
- 「SQL 関数 (P ～ Z)」 『SQL Anywhere サーバ - SQL リファレンス』

EXP 関数 [数値]

指数関数 (e を底とする数のべき乗) を返します。

構文

EXP(*numeric-expression* **)**

パラメータ

- **numeric-expression** 指数。

戻り値

DOUBLE

備考

EXP 関数は、*numeric-expression* で指定された値の指数値を返します。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 3269017.3724721107 を返します。

```
SELECT EXP( 15 );
```

EXPLANATION 関数 [その他]

SQL 文のプランの最適化方法を返します。

構文

EXPLANATION(string-expression **)**

パラメータ

- **string-expression**　SQL 文。通常は SELECT 文ですが、UPDATE 文または DELETE 文も指定できます。

戻り値

LONG VARCHAR

備考

最適化結果は文字列として返されます。

この情報は、追加するインデックスの決定や、パフォーマンスを向上するためのデータベース構造の決定に役立ちます。

参照

- 「Ultra Light の実行プラン」 360 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、SELECT 文を文字列パラメータとして渡し、クエリを実行するためのプランを返します。

```
SELECT EXPLANATION( 'SELECT * FROM Departments WHERE DepartmentID > 100' );
```

FLOOR 関数 [数値]

指定された値以下で、最大の整数値を返します。

構文

FLOOR(*numeric-expression* **)**

パラメータ

- **numeric-expression**　値を指定します。通常は FLOAT です。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「CEILING 関数 [数値]」 380 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、123 の Floor 値を返します。

```
SELECT FLOOR (123);
```

次の文は、123 の Floor 値を返します。

```
SELECT FLOOR (123.45);
```

次の文は、-124 の Floor 値を返します。

```
SELECT FLOOR (-123.45);
```

GETDATE 関数 [日付と時刻]

現在の年、月、日、時、分、秒、秒以下を返します。

構文

GETDATE( )

戻り値

TIMESTAMP

備考

精度はシステム・クロックの精度によって制限されます。

GETDATE 関数が返す情報は、NOW 関数と CURRENT TIMESTAMP 特別値が返す情報と同じです。

参照

- 「NOW 関数 [日付と時刻]」 428 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、システムの日付と時刻を返します。

```
SELECT GETDATE( );
```

GREATER 関数 [その他]

2 つのパラメータ値のうち、より大きい値を返します。

構文

GREATER(*expression-1*, *expression-2* **)**

パラメータ

- **expression-1**　比較される最初のパラメータ値。
- **expression-2**　比較される 2 番目のパラメータ値。

戻り値

任意の値

備考

パラメータが等しい場合は、最初のパラメータを返します。

参照

- 「LESSER 関数 [その他]」 414 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 10 を返します。

```
SELECT GREATER( 10, 5 ) FROM dummy;
```

HEXTOINT 関数 [データ型変換]

16 進文字列と同等の 10 進整数を返します。

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。これらの関数の使用の詳細については、「16 進値との変換」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

構文

HEXTOINT(*hexadecimal-string* **)**

パラメータ

- **hexadecimal-string**　整数に変換される文字列。

戻り値

HEXTOINT 関数は、プラットフォームに依存しない SQL INTEGER 相当の 16 進文字列を返します。右から 8 桁目が数値 8 ～ 9 か、大文字または小文字の A ～ F のいずれかであり、その前の桁がすべて大文字または小文字の F の場合は、16 進値が負の整数値になります。次の HEXTOINT は無効な使用例です。これは、引数が符号付き 32 ビット整数で表現できない正の整数値を示しているためです。

```
SELECT HEXTOINT( '0x0080000001' );
```

INT

備考

HEXTOINT 関数は、数値、大文字または小文字の A ～ F のみで構成される文字列リテラルまたは変数を受け入れます (0x プレフィクスが付いている場合、付いていない場合の両方)。次に HEXTOINT の有効な使用例をすべて示します。

```
SELECT HEXTOINT( '0xFFFFFFFF' );
SELECT HEXTOINT( '0x00000100' );
SELECT HEXTOINT( '100' );
SELECT HEXTOINT( '0xffffffff80000001' );
```

HEXTOINT 関数は 0x プレフィクスがあれば削除します。データが 8 桁を超える場合、符号付き 32 ビット整数値として表現できる値を示す必要があります。

参照

- 「INTTOHEX 関数 [データ型変換]」 409 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 420 を返します。

```
SELECT HEXTOINT( '1A4' );
```

HOUR 関数 [日付と時刻]

日時の時間コンポーネントを返します。

構文

HOUR(*datetime-expression* **)**

パラメータ

- **datetime-expression** 日時。

戻り値

SMALLINT

備考

返される値は日時の時間に対応する 0 ～ 23 の数値です。

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 21 を返します。

```
SELECT HOUR( '1998-07-09 21:12:13' );
```

HOURS 関数 [日付と時刻]

時間を評価する関数。詳細については、この関数の使用法を参照してください。

構文 1：整数

HOURS ([*datetime-expression*,] *datetime-expression* **)**

構文 2：タイムスタンプ

HOURS (*datetime-expression*, *integer-expression* **)**

パラメータ

- **datetime-expression**　日付と時刻。
- **integer-expression**　*datetime-expression* に加算する時間数。*integer-expression* が負の場合、日時から適切な時間数が引かれます。整数式を指定する場合は、*datetime-expression* を DATETIME データ型として明示的にキャストしてください。

  データ型のキャストの詳細については、「CAST 関数 [データ型変換]」 379 ページを参照してください。

戻り値

INT

TIMESTAMP

備考

この関数の動作は、指定内容によって異なります。

- 1 つの日付を指定すると、0000-02-29 からの時間数を返します。

  > **注意**
  > 0000-02-29 は実際の日付を指すための値ではありません。日付アルゴリズムで使用される日付です。

- 2 つのタイムスタンプを指定すると、2 つの時刻の間の時間数を整数で返します。代わりに、DATEDIFF 関数を使用します。
- 1 つの日付と整数を指定すると、指定した日付に、指定した整数の時間数が加算されます。代わりに、DATEADD 関数を使用します。

参照

- 「DATEDIFF 関数 [日付と時刻]」 391 ページ
- 「DATEADD 関数 [日付と時刻]」 390 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 4 を返します。これは、2 番目のタイムスタンプが、最初のタイムスタンプの 4 時間後であることを示します。2 つ目の例 (DATEDIFF) の使用をおすすめします。

```
SELECT HOURS( '1999-07-13 06:07:12',
  '1999-07-13 10:07:12' );

SELECT DATEDIFF( hour,
  '1999-07-13 06:07:12',
  '1999-07-13 10:07:12' );
```

次の文は、値 17517342 を返します。

```
SELECT HOURS( '1998-07-13 06:07:12' );
```

次の文は、日時 1999-05-13 02:05:07.000 を返します。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT HOURS(
  CAST( '1999-05-12 21:05:07' AS DATETIME ), 5 );

SELECT DATEADD( hour, 5, '1999-05-12 21:05:07' );
```

IFNULL 関数 [その他]

最初の式が NULL 値の場合は、2 番目の式の値を返します。最初の式が NULL でない場合は、3 番目の式の値を返します。最初の式が NULL ではなく、3 番目の式が存在しない場合は、NULL を返します。

構文

IFNULL(*expression-1*, *expression-2* [, *expression-3*] **)**

パラメータ

- **expression-1** 評価される式。この値によって、*expression-2* または *expression-3* のどちらが返るかが決まります。
- **expression-2** *expression-1* が NULL の場合の戻り値。
- **expression-3** *expression-1* が NULL でない場合の戻り値。

戻り値

返されるデータ型は、*expression-2* と *expression-3* のデータ型によって異なります。

標準と互換性

- **SQL/2003**　Transact-SQL 拡張。

例

次の文は、値 -66 を返します。

```
SELECT IFNULL( NULL, -66 );
```

次の文は、最初の式が NULL ではなく、3 番目の式が存在しないため、NULL を返します。

```
SELECT IFNULL( -66, -66 );
```

INSERTSTR 関数 [文字列]

別の文字列の指定された位置に文字列を挿入します。

構文

INSERTSTR(*integer-expression*, *string-expression-1*, *string-expression-2* **)**

パラメータ

- **integer-expression**　文字列の挿入位置。文字列の先頭に挿入する場合は、0 を使用します。
- **string-expression-1**　別の文字列が挿入される文字列。
- **string-expression-2**　挿入する文字列。

戻り値

LONG VARCHAR

参照

- 「STUFF 関数 [文字列]」 449 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 backoffice を返します。

```
SELECT INSERTSTR( 0, 'office ', 'back' );
```

INTTOHEX 関数 [データ型変換]

整数に対応する 16 進値の文字列を返します。

構文

INTTOHEX(*integer-expression* **)**

パラメータ

- **integer-expression**　16 進に変換される整数。

戻り値

VARCHAR

備考

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。詳細については、「16 進値との変換」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

参照

- 「HEXTOINT 関数 [データ型変換]」 405 ページ

標準と互換性

- **SQL/2003**　Transact-SQL 拡張。

例

次の文は、値 0000009c を返します。

```
SELECT INTTOHEX( 156 );
```

ISDATE 関数 [データ型変換]

文字列引数を日付に変換できるかどうかをテストします。

構文

ISDATE(*string* **)**

パラメータ

- **string**　文字列が有効な日付を表すかどうかを判断するために分析される文字列。

戻り値

INT

備考

変換できる場合は 1 を返し、できない場合は 0 を返します。引数が NULL の場合は 0 を返します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

ISNULL 関数 [その他]

リストの中から NULL でない最初の式を返します。この関数は COALESCE 関数と同じです。

構文

ISNULL(*expression*, *expression* [, ...] **)**

パラメータ

- **expression**　NULL かどうかがテストされる式。

  2 つ以上の式を関数に渡します。すべての式は比較可能であることが必要です。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベース・サーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベース・サーバは式を比較し、比較に使用したデータ型で結果を返します。データベース・サーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

参照

- 「COALESCE 関数 [その他]」 384 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 -66 を返します。

```
SELECT ISNULL( NULL ,-66, 55, 45, NULL, 16 );
```

LCASE 関数 [文字列]

文字列中のすべての文字を小文字に変換します。この関数は LOWER 関数と同じです。

構文

LCASE(*string-expression* **)**

パラメータ

- **string-expression**　小文字に変換される文字列。

戻り値

CHAR

NCHAR

LONG VARCHAR

VARCHAR

NVARCHAR

備考

LCASE 関数は LOWER 関数に似ています。

参照

- 「LOWER 関数 [文字列]」 418 ページ
- 「UCASE 関数 [文字列]」 458 ページ
- 「UPPER 関数 [文字列]」 458 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 chocolate を返します。

```
SELECT LCASE( 'ChoCOlatE' );
```

LEFT 関数 [文字列]

文字列の先頭からいくつかの文字を返します。

構文

LEFT(*string-expression*, *integer-expression* **)**

パラメータ

- **string-expression**　文字列。
- **integer-expression**　返す文字数。

戻り値

LONG VARCHAR

LONG NVARCHAR

備考

文字列にマルチバイト文字があり、適切な照合が使用されている場合、返されるバイト数が指定した文字数よりも大きい場合があります。

引数 string-expression の値より大きな *integer-expression* を指定できます。この場合、全体の値が返されます。

入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値は文字長のセマンティックの観点から記述されます。

参照

- 「RIGHT 関数 [文字列]」 437 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、Customers テーブルに含まれる各 Surname 値の最初の 5 文字を返します。

```
SELECT LEFT( Surname, 5) FROM Customers;
```

LENGTH 関数 [文字列]

指定した文字列の文字数を返します。

構文

LENGTH(*string-expression* **)**

パラメータ

- **string-expression**　文字列。

戻り値

INT

備考

この関数を使用すると、文字列の長さがわかります。たとえば、*string-expression* にカラム名を指定すると、カラムの値の長さがわかります。

文字列にマルチバイト文字があり、適切な照合が使用されている場合、LENGTH はバイト数ではなく、文字数を返します。文字列が BINARY データ型の場合、LENGTH 関数は BYTE_LENGTH 関数のように動作します。

注意
データ型が CHAR、VARCHAR、LONG VARCHAR の場合、LENGTH 関数と CHAR_LENGTH 関数を使用すると同じ結果が得られます。ただし、BINARY データ型とビット配列データ型には LENGTH 関数を使用します。

参照

- 「BYTE_LENGTH 関数 [文字列]」 378 ページ
- 「国際言語と文字セット」 『SQL Anywhere サーバ - データベース管理』
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 9 を返します。

```
SELECT LENGTH( 'chocolate' );
```

LESSER 関数 [その他]

2 つのパラメータ値のうち、より小さい値を返します。

構文

LESSER(*expression-1*, *expression-2* **)**

パラメータ

- **expression-1** 比較される最初のパラメータ値。
- **expression-2** 比較される 2 番目のパラメータ値。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベース・サーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベース・サーバは式を比較し、比較に使用したデータ型で結果を返します。データベース・サーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

備考

パラメータが等しい場合は、最初の値を返します。

参照

- 「GREATER 関数 [その他]」 404 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 5 を返します。

```
SELECT LESSER( 10, 5 ) FROM dummy;
```

LIST 関数 [集合]

カンマで区切られた値のリストを返します。

構文

LIST(
[**DISTINCT**] *string-expression*
[, *delimiter-string*] **)**

パラメータ

- **string-expression**　文字列式。通常はカラム名です。カラムの各ローに対して、カンマで区切られたリストに値が追加されます。DISTINCT を指定すると、ユニークな値のみが追加されます。
- **delimiter-string**　リスト項目のデリミタ文字列。デフォルト設定はカンマです。NULL 値または空の文字列を指定した場合は、デリミタはありません。*delimiter-string* は定数です。

戻り値

LONG VARCHAR

LONG NVARCHAR

備考

NULL 値は、リストに追加されません。LIST (X) は、グループの各ローの、NULL 以外のすべての X の値を (デリミタ付きで) 連結させて返します。グループ内に明確な X 値を持ったローが 1 つ以上存在しない場合、LIST(X) は空の文字列を返します。

LIST 関数は Window 関数として使用できませんが、Window 関数の入力には使用できます。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、Employees テーブルのすべての住所を返します。

```
SELECT LIST( Street ) FROM Employees;
```

LOCATE 関数 [文字列]

異なる文字列内のある文字列の位置を返します。

構文

LOCATE(*string-expression-1*, *string-expression-2* [, *integer-expression*] **)**

パラメータ

- **string-expression-1**　検索される文字列。
- **string-expression-2**　検索する文字列。
- **integer-expression**　文字列内の、検索を開始する位置。最初の文字の位置は 1 です。開始オフセットが負の場合は、最初に一致した文字列ではなく、最後に一致した文字列のオフセッ

トを返します。負のオフセットは、文字列の末尾のどれだけの部分が検索から除外されるかを示します。除外されるバイト数は、(-1 * offset) -1 で計算されます。

戻り値

INT

備考

integer-expression を指定すると、文字列のオフセットから検索が開始します。

最初の文字列には長い文字列 (255 バイト以上) を指定できますが、2 番目の文字列は 255 バイトに制限されます。長い文字列を 2 番目の引数として指定すると、LOCATE 関数は NULL 値を返します。文字列が見つからない場合は、0 を返します。長さ 0 の文字列を検索すると、1 を返します。いずれかの引数が NULL の場合は、結果は NULL です。

マルチバイト文字が使用され、適切な照合が使用されている場合は、開始位置と返される値は**バイト**で計算した位置と異なる場合があります。

参照

- 「Ultra Light 文字列関数」 370 ページ
- 「CHARINDEX 関数 [文字列]」 382 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 8 を返します。

```
SELECT LOCATE(
  'office party this week - rsvp as soon as possible',
  'party',
  2 );
```

LOG 関数 [数値]

数の自然対数を返します。

構文

LOG(*numeric-expression* **)**

パラメータ

- **numeric-expression**　数値。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

備考

引数は、組み込みの数値データ型の値を返す式です。

参照

● 「LOG10 関数 [数値]」 417 ページ

標準と互換性

● **SQL/2003**　ベンダ拡張。

例

次の文は、50 の自然対数を返します。

```
SELECT LOG( 50 );
```

LOG10 関数 [数値]

数値の対数 (基数 10) を返します。

構文

LOG10(*numeric-expression* **)**

パラメータ

● **numeric-expression**　数値。

戻り値

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。パラメータが NULL 値の場合、結果は NULL 値になります。

備考

引数は、組み込みの数値データ型の値を返す式です。

参照

● 「LOG 関数 [数値]」 416 ページ

標準と互換性

● **SQL/2003**　ベンダ拡張。

例

次の文は、10 を底とする 50 の対数を返します。

```
SELECT LOG10( 50 );
```

LOWER 関数 [文字列]

文字列中のすべての文字を小文字に変換します。この関数は LCASE 関数と同じです。

構文

LOWER(*string-expression* **)**

パラメータ

- **string-expression**　変換される文字列。

戻り値

CHAR

NCHAR

LONG VARCHAR

VARCHAR

NVARCHAR

備考

LCASE 関数は LOWER 関数と同じです。

参照

- 「LCASE 関数 [文字列]」 411 ページ
- 「UCASE 関数 [文字列]」 458 ページ
- 「UPPER 関数 [文字列]」 458 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　コア機能。

例

次の文は、値 chocolate を返します。

```
SELECT LOWER( 'chOCOLate' );
```

LTRIM 関数 [文字列]

先行ブランクと後続ブランクを文字列から削除します。

構文

LTRIM(*string-expression* **)**

パラメータ

- **string-expression**　削除される文字列。

戻り値

VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

戻り値

結果の実際の長さは、式の長さから、削除される文字数を引いた値です。すべての文字を削除すると、結果は空の文字列になります。

パラメータに NULL を指定できる場合、結果は NULL になります。

パラメータが NULL の場合、結果は NULL 値になります。

参照

- 「RTRIM 関数 [文字列]」 438 ページ
- 「TRIM 関数 [文字列]」 455 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

  SQL/2003 規格で定義される TRIM 仕様 (LEADING と TRAILING) は、それぞれ SQL Anywhere の LTRIM 関数と RTRIM 関数から指定されます。

例

次の文は、すべての先行ブランクを削除して、値 Test Message を返します。

```
SELECT LTRIM( '    Test Message' );
```

MAX 関数 [集合]

各ロー・グループで見つかった *expression* の最大値を返します。

構文 1

MAX(*expression* | **DISTINCT** *expression* **)**

パラメータ

- **expression**　最大値が計算される式。通常はカラム名です。
- **DISTINCT 式**　MAX(*expression*) の場合と同じ値を返します。万全を期すために含まれています。

戻り値

引数と同じデータ型。

備考

expression が NULL のローは、無視されます。グループにローが含まれていない場合は、NULL を返します。

参照

- 「MIN 関数 [集合]」 420 ページ

標準と互換性

- **SQL/2003** コア機能。

例

次の文は、Employees テーブル内の給与の最大値 138948.000 を返します。

```
SELECT MAX( Salary )
FROM Employees;
```

MIN 関数 [集合]

各ロー・グループで見つかった expression の最小値を返します。

構文 1

MIN(*expression* | **DISTINCT** *expression* **)**

パラメータ

- **expression** 最小値が計算される式。通常はカラム名です。
- **DISTINCT 式** MIN(*expression*) の場合と同じ値を返します。万全を期すために含まれています。

戻り値

引数と同じデータ型。

備考

expression が NULL のローは、無視されます。グループにローが含まれていない場合は、NULL を返します。

参照

- 「MAX 関数 [集合]」 419 ページ

標準と互換性

- **SQL/2003** コア機能。

例

次の文は、Employees テーブル内の給与の最小値 24903.000 を返します。

```
SELECT MIN( Salary )
FROM Employees;
```

MINUTE 関数 [日付と時刻]

日時値の分コンポーネントを返します。

構文

MINUTE(*datetime-expression* **)**

パラメータ

- **datetime-expression**　日時の値。

戻り値

SMALLINT

備考

返される値は日時の分に対応する 0 ～ 59 の数値です。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 22 を返します。

```
SELECT MINUTE( '1998-07-13 12:22:34' );
```

MINUTES 関数 [日付と時刻]

この関数の動作は、指定内容によって異なります。

- 1 つの日付を指定すると、0000-02-29 からの分数を返します。

  注意
  0000-02-29 は実際の日付を指すための値ではありません。日付アルゴリズムで使用される日付です。

- 2 つのタイムスタンプを指定すると、2 つの時刻の間の分数を整数で返します。代わりに、DATEDIFF 関数を使用します。
- 1 つの日付と整数を指定すると、指定した日付に、指定した整数の分数が加算されます。代わりに、DATEADD 関数を使用します。

構文 1：整数

MINUTES([*datetime-expression*,] *datetime-expression* **)**

構文 2：タイムスタンプ

MINUTES(*datetime-expression*, *integer-expression* **)**

パラメータ

- **datetime-expression**　日付と時刻。
- **integer-expression**　*datetime-expression* に加算する分数。*integer-expression* が負の場合、日時の値から適切な分数が引かれます。整数式を指定する場合は、*datetime-expression* を DATETIME データ型として明示的にキャストしてください。

戻り値

INT

TIMESTAMP

備考

この関数は整数を返すため、構文 1 で 4083-03-23 02:08:00 以上のタイムスタンプを使用すると、オーバフローが発生する場合があります。

参照

- 「CAST 関数 [データ型変換]」 379 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 240 を返します。これは、2 番目のタイムスタンプが、最初のタイムスタンプの 240 分後であることを示します。2 つ目の例 (DATEDIFF) の使用をおすすめします。

```
SELECT MINUTES( '1999-07-13 06:07:12',
   '1999-07-13 10:07:12' );

SELECT DATEDIFF( minute,
    '1999-07-13 06:07:12',
    '1999-07-13 10:07:12' );
```

次の文は、値 1051040527 を返します。

```
SELECT MINUTES( '1998-07-13 06:07:12' );
```

次の文は、タイムスタンプ 1999-05-12 21:10:07.000 を返します。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT MINUTES( CAST( '1999-05-12 21:05:07'
AS DATETIME ), 5);

SELECT DATEADD( minute, 5, '1999-05-12 21:05:07' );
```


ML_GET_SERVER_NOTIFICATION [システム]

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 にクエリできます。

構文

ML_GET_SERVER_NOTIFICATION(*notifier*, *address*, *key***)**

パラメータ

- **notifier**　ポーリング対象の Mobile Link サーバ上の Notifier 名。
- **address**　ストリーム・パラメータ。たとえば、

  ```
  tcpip{host=pc1;port=1234}
  ```

  のように指定します。
- **key**　省略可。通知要求キーです。

戻り値

指定された要求キーに対する通知要求の件名と内容を返します。

備考

指定された要求キーに対する要求がない場合、または指定された Notifier 名が Mobile Link サーバ上に存在しない場合、結果は NULL になります。要求キーに NULL が指定された場合、そのユーザのリモート ID が要求キーとして使用されます。要求が存在する場合、結果メッセージが [subject]content (たとえば、[sync]profile1) という形式で返されます。

この関数は、Mobile Link サーバからの応答を取得する際にネットワーク経由の通信を行います。そのため、ネットワーク遅延により実行に時間がかかることがあります。実行中、この関数をバックグラウンドで実行できる期間がある場合があります。その場合、他の接続上のランタイム内で作業を行うことができます。ただし、このような期間の存在が保証されているわけではなく、SQL の複雑さに依存します。この関数で使用する Mobile Link アドレスを取得する場合は、sync_profile_option_value 関数で既存の同期プロファイルを使用して、**Stream** プロファイル・オプションの値を取得することをおすすめします。この関数が返す値は、そのまま Mobile Link アドレス・パラメータとして使用できます。

参照

- 「SYNC_PROFILE_OPTION_VALUE 関数 [システム]」 453 ページ

標準と互換性

例

```
Select ml_get_server_notification('Notifier1', 'tcpip{host=sybase;port=1234}', 'MyKey'
```

MOD 関数 [数値]

整数を整数で割ったときの余りを返します。

構文

MOD(*dividend*, *divisor* **)**

パラメータ

- **dividend**　被除数 (分数の分子)。
- **divisor**　除数 (分数の分母)。

戻り値

SMALLINT

INT

NUMERIC

備考

被除数が負の場合、除算の結果は負または 0 になります。除数の符号は計算結果に影響を与えません。

参照

- 「REMAINDER 関数 [数値]」 434 ページ

標準と互換性

- **SQL/2003**　コア SQL に含まれない SQL 基本機能。

例

次の文は、値 2 を返します。

```
SELECT MOD( 5, 3 );
```

MONTH 関数 [日付と時刻]

指定した日付の月を返します。

構文

MONTH(*date-expression* **)**

パラメータ

- **date-expression**　日時の値。

戻り値

SMALLINT

備考

返される値は日時の月に対応する 1 ～ 12 の数値です。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 7 を返します。

```
SELECT MONTH( '1998-07-13' );
```

MONTHNAME 関数 [日付と時刻]

日付から月の名前を返します。

構文

MONTHNAME(*date-expression* **)**

パラメータ

- **date-expression**　日時の値。

戻り値

VARCHAR

備考

結果が数値 (2 月を示す 2 など) の場合でも、MONTHNAME 関数は文字列を返します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 September を返します。

```
SELECT MONTHNAME( '1998-09-05' );
```

MONTHS 関数 [日付と時刻]

この関数の動作は、指定内容によって異なります。

- 1 つの日付を指定すると、0000-02 からの月数を返します。

注意
0000-02 は実際の日付を指すための値ではありません。日付アルゴリズムで使用される日付です。

- 2 つのタイムスタンプを指定すると、2 つの日付の間の月数を整数で返します。代わりに、DATEDIFF 関数を使用します。
- 1 つの日付と整数を指定すると、指定した日付に、指定した整数の分数が加算されます。代わりに、DATEADD 関数を使用します。

構文 1：整数

MONTHS([*datetime-expression*,] *datetime-expression* **)**

構文 2：タイムスタンプ

MONTHS(*datetime-expression*, *integer-expression* **)**

パラメータ

- **datetime-expression**　日付と時刻。
- **integer-expression**　*datetime-expression* に加算する月数。*integer-expression* が負の場合、日時の値から適切な月数が引かれます。*integer-expression* を指定する場合は、*datetime-expression* を datetime データ型として明示的にキャストしてください。

  データ型のキャストの詳細については、「CAST 関数 [データ型変換]」 379 ページを参照してください。

戻り値

INT

TIMESTAMP

備考

MONTHS の値を求めるには、2 つの日付の間に月の最初の日がいくつあるかを計算します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 2 を返します。これは、2 番目の日付が、最初の日付の 2 か月後であることを示します。2 つ目の例 (DATEDIFF) の使用をおすすめします。

```
SELECT MONTHS( '1999-07-13 06:07:12',
  '1999-09-13 10:07:12' );

SELECT DATEDIFF( month,
  '1999-07-13 06:07:12',
  '1999-09-13 10:07:12' );
```

次の文は、値 23981 を返します。

```
SELECT MONTHS( '1998-07-13 06:07:12' );
```


次の文は、タイムスタンプ 1999-10-12 21:05:07.000 を返します。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT MONTHS( CAST( '1999-05-12 21:05:07'
AS DATETIME ), 5);

SELECT DATEADD( month, 5, '1999-05-12 21:05:07' );
```

NEWID 関数 [その他]

UUID (ユニバーサル・ユニーク識別子) 値を生成します。UUID は、GUID (グローバル・ユニーク識別子) と同じです。

構文

NEWID( )

パラメータ

NEWID 関数に関連付けられているパラメータはありません。

戻り値

UNIQUEIDENTIFIER

備考

NEWID 関数は、カラムの DEFAULT 句で使用できます。

UUID を使用して、テーブルのローをユニークに識別できます。コンピュータが異なると生成される値も異なるので、値は同期環境やレプリケーション環境でキーとして使用できます。

参照

- 「NEWID デフォルト」 『SQL Anywhere サーバ - SQL の使用法』
- 「STRTOUUID 関数 [文字列]」 448 ページ
- 「UUIDTOSTR 関数 [文字列]」 460 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、2 つのカラムを持つテーブル mytab を作成します。カラム pk は uniqueidentifier データ型とし、NEWID 関数をデフォルト値として割り当てます。カラム c1 は integer データ型です。

```
CREATE TABLE mytab(
  pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  c1 INT );
```

次の文は、ユニーク識別子を文字列として返します。

```
SELECT NEWID();
```

たとえば、戻り値が 96603324-6FF6-49DE-BF7D-F44C1C7E6856 の場合もあります。

NOW 関数 [日付と時刻]

現在の年、月、日、時、分、秒、秒以下を返します。精度はシステム・クロックの精度によって制限されます。

構文

NOW(*)

戻り値

TIMESTAMP

備考

NOW 関数が返す情報は、GETDATE 関数と CURRENT TIMESTAMP 特別値が返す情報と同じです。

参照

- 「GETDATE 関数 [日付と時刻]」 404 ページ
- 「CURRENT TIMESTAMP 特別値」 325 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、現在の日付と時刻を返します。

```
SELECT NOW( * );
```

NULLIF 関数 [その他]

式を比較して、CASE 式の省略形を提供します。

構文

NULLIF(*expression-1*, *expression-2* **)**

パラメータ

- **expression-1** 比較される式。
- **expression-2** 比較される式。

戻り値

最初の引数のデータ型。

備考

NULLIF は 2 つの式の値を比較します。

1 番目の式と 2 番目の式が等しい場合、NULLIF は NULL を返します。

1 番目の式と 2 番目の式が異なる場合、または 2 番目の式が NULL の場合、NULLIF は 1 番目の式を返します。

NULLIF 関数は、いくつかの CASE 式の簡単な作成方法を提供します。

参照

- 「CASE 式」 345 ページ

標準と互換性

- **SQL/2003** コア機能。

例

次の文は、値 a を返します。

```
SELECT NULLIF( 'a', 'b' );
```

次の文は、NULL を返します。

```
SELECT NULLIF( 'a', 'a' );
```

SQL 関数 (P ～ Z)

関数を 1 つずつリストし、その右側に関数のタイプ (数値、文字など) を示します。

特定のタイプのすべての関数へのリンクについては、「関数のタイプ」 366 ページを参照してください。

参照

- 「SQL 関数 (A ～ D)」 『SQL Anywhere サーバ - SQL リファレンス』
- 「SQL 関数 (E ～ O)」 『SQL Anywhere サーバ - SQL リファレンス』

PATINDEX 関数 [文字列]

文字列のパターンが最初に出現した開始位置を表す整数を返します。

構文

PATINDEX('%*pattern***%'**, *string-expression* **)**

パラメータ

- **pattern**　検索するパターン。先頭の % ワイルドカードを省略すると、PATINDEX 関数は、パターンが文字列の先頭に出現する場合は 1 を返し、それ以外の場合は 0 を返します。

  Ultra Light のパターンは、次のワイルドカードを使用します。

ワイルドカード	一致するもの
_ (アンダースコア)	任意の 1 文字
% (パーセント記号)	0 個以上の文字からなる任意の文字列
[]	指定範囲内、または一連の指定文字の任意の 1 文字
[^]	指定範囲外、または一連の指定文字以外の任意の 1 文字

- **string-expression**　パターンを検索する文字列。

戻り値

INT

備考

PATINDEX 関数は、パターンが最初に出現した開始位置を返します。パターンが見つからない場合は、0 を返します。

参照

- 「LOCATE 関数 [文字列]」 415 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 2 を返します。

```
SELECT PATINDEX( '%hoco%', 'chocolate' );
```

次の文は、値 11 を返します。

```
SELECT PATINDEX( '%4_5_', '0a1A 2a3A 4a5A' );
```

次の文は、14 を返します。これは、文字列式の最初の英数字以外の文字です。データベースで大文字と小文字が区別されない場合は、'%[^a-zA-Z0-9]%' の代わりにパターン '%[^a-z0-9]%' を使用できます。

```
SELECT PATINDEX( '%[^a-zA-Z0-9]%', 'SQLAnywhere11 has many new features' );
```

文字列の最初の英数字を取得するには、次の手順に従います。

```
SELECT LEFT( @string, PATINDEX( '%[^a-zA-Z0-9]%', @string ) );
```

PI 関数 [数値]

PI の数値を返します。

構文

PI(*)

戻り値

DOUBLE

標準と互換性

- **SQL/2003** ベンダ拡張。

備考

この関数は DOUBLE 値を返します。

例

次の文は、値 3.141592653... を返します。

```
SELECT PI( * );
```

POWER 関数 [数値]

数のべき乗を表す数を計算します。

構文

POWER(*numeric-expression-1*, *numeric-expression-2* **)**

パラメータ

- **numeric-expression-1**　べき乗の底。
- **numeric-expression-2**　指数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。引数のいずれかが NULL の場合、結果は NULL 値になります。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 64 を返します。

```
SELECT POWER( 2, 6 );
```

QUARTER 関数 [日付と時刻]

指定した日付式から、四半期を示す数を返します。

構文

QUARTER(*date-expression* **)**

パラメータ

- **date-expression**　日付。

戻り値

INT

備考

四半期は次のとおりです。

Quarter	期間 (開始日と終了日を含む)
1	1 月 1 日～ 3 月 31 日
2	4 月 1 日～ 6 月 30 日
3	7 月 1 日～ 9 月 30 日
4	10 月 1 日～ 12 月 31 日

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 2 を返します。

```
SELECT QUARTER( '1987/05/02' );
```

RADIANS 関数 [数値]

度数をラジアンに変換します。

構文

RADIANS(*numeric-expression* **)**

パラメータ

- **numeric-expression**　度数。この角度をラジアンに変換します。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、約 0.5236 の値を返します。

```
SELECT RADIANS( 30 );
```

REMAINDER 関数 [数値]

整数を整数で割ったときの余りを返します。

構文

REMAINDER(*dividend*, *divisor* **)**

パラメータ

- **dividend**　被除数 (分数の分子)。
- **divisor**　除数 (分数の分母)。

戻り値

INTEGER

NUMERIC

備考

または、MOD 関数も使用できます。

参照

- 「MOD 関数 [数値]」 424 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 2 を返します。

```
SELECT REMAINDER( 5, 3 );
```

REPEAT 関数 [文字列]

文字列を指定された回数だけ連結します。

構文

REPEAT(*string-expression*, *integer-expression* **)**

パラメータ

- **string-expression**　繰り返される文字列。
- **integer-expression**　文字列を繰り返す回数。*integer-expression* が負の場合は、空の文字列を返します。

戻り値

LONG VARCHAR

LONG NVARCHAR

備考

実際の結果文字列の長さが戻り値の最大長を超えると、エラーが発生します。この場合、結果は文字列に使用できる最大サイズまでトランケートされます。

または、REPLICATE 関数も使用できます。

参照

- 「REPLICATE 関数 [文字列]」 436 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 repeatrepeatrepeat を返します。

```
SELECT REPEAT( 'repeat', 3 );
```

REPLACE 関数 [文字列]

文字列を別の文字列で置換し、新しい結果を返します。

構文

REPLACE(*original-string*, *search-string*, *replace-string* **)**

パラメータ

いずれかの引数が NULL の場合、NULL を返します。

- **original-string**　検索される文字列。任意の長さの文字列を指定できます。
- **search-string**　検索して *replace-string* に置換する文字列。この文字列は 255 バイトに制限されています。*search-string* が空の文字列の場合は、元の文字列を未変更のまま返します。
- **replace-string**　*search-string* と置き換える置換文字列。任意の長さの文字列を指定できます。*replacement-string* が空の文字列の場合は、すべての *search-string* が削除されます。

戻り値

LONG VARCHAR

LONG NVARCHAR

備考

この関数はすべてのオカレンスを置換します。

大文字と小文字を区別するデータベースでの比較では、大文字と小文字が区別されます。

参照

- 「SUBSTRING 関数 [文字列]」 449 ページ
- 「CHARINDEX 関数 [文字列]」 382 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 xx.def.xx.ghi を返します。

```
SELECT REPLACE( 'abc.def.abc.ghi', 'abc', 'xx' );
```

次の文は、ALTER PROCEDURE 文を含む結果セットを生成します。ALTER PROCEDURE を実行すると、名前が変更されたテーブルを参照する格納済みプロシージャが修復されます (テーブル名を一意にすることをおすすめします)。

```
SELECT REPLACE(
  REPLACE( proc_defn, 'OldTableName', 'NewTableName' ),
  'CREATE PROCEDURE',
  'ALTER PROCEDURE')
FROM SYS.SYSPROCEDURE
WHERE proc_defn LIKE '%OldTableName%';
```

REPLICATE 関数 [文字列]

文字列を指定された回数だけ連結します。

構文

REPLICATE(*string-expression*, *integer-expression* **)**

パラメータ

- **string-expression**　繰り返される文字列。
- **integer-expression**　文字列を繰り返す回数。

戻り値

LONG VARCHAR

LONG NVARCHAR

備考

実際の結果文字列の長さが戻り値の最大長を超えると、エラーが発生します。この場合、結果は文字列に使用できる最大サイズまでトランケートされます。

または、REPEAT 関数も使用できます。

参照

- 「REPEAT 関数 [文字列]」 434 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 repeatrepeatrepeat を返します。

```
SELECT REPLICATE( 'repeat', 3 );
```

RIGHT 関数 [文字列]

文字列の一番右側にある文字を返します。

構文

RIGHT(*string-expression*, *integer-expression* **)**

パラメータ

- **string-expression** 左側をトランケートされる文字列。
- **integer-expression** 返される文字列の末尾の文字数。

戻り値

LONG VARCHAR

LONG NVARCHAR

備考

文字列にマルチバイト文字があり、適切な照合が使用されている場合、返されるバイト数が指定した文字数よりも大きい場合があります。

カラムの値より大きな *integer-expression* を指定できます。この場合、全体の値が返されます。

入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値は文字長のセマンティックの観点から記述されます。

参照

- 「LEFT 関数 [文字列]」 412 ページ
- 「国際言語と文字セット」 『SQL Anywhere サーバ - データベース管理』
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、Customers テーブルに含まれる各 Surname 値の最後の 5 文字を返します。

```
SELECT RIGHT( Surname, 5) FROM Customers;
```

ROUND 関数 [数値]

integer-expression で指定した小数点以下の桁数に *numeric-expression* を丸めます。

構文

ROUND(*numeric-expression*, *integer-expression* **)**

パラメータ

- **numeric-expression**　関数に渡される、丸めの対象となる数。
- **integer-expression**　正の整数は、丸めを行う小数点の右側の有効桁数を指定します。負の式は、丸めを行う小数点の左側の有効桁数を指定します。

戻り値

NUMERIC

備考

この関数の結果は numeric または double です。数値の結果があり、整数 *integer-expression* が負の値の場合、精度は 1 ずつ増えます。

参照

- 「TRUNCNUM 関数 [数値]」 456 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 123.200 を返します。

```
SELECT ROUND( 123.234, 1 );
```

RTRIM 関数 [文字列]

先行ブランクと後続ブランクを文字列から削除します。

構文

RTRIM(*string-expression* **)**

パラメータ

- **string-expression**　削除される文字列。

戻り値

VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

備考

結果の実際の長さは、式の長さから、削除される文字数を引いた値です。すべての文字を削除すると、結果は空の文字列になります。

引数が NULL の場合、結果は NULL 値になります。

参照

- 「TRIM 関数 [文字列]」 455 ページ
- 「LTRIM 関数 [文字列]」 418 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

  SQL/2003 規格で定義される TRIM 仕様 (LEADING と TRAILING) は、それぞれ SQL Anywhere の LTRIM 関数と RTRIM 関数から指定されます。

例

次の文は、すべての後続ブランクが削除された文字列 Test Message を返します。

```
SELECT RTRIM( 'Test Message    ' );
```

SECOND 関数 [日付と時刻]

指定した日付の秒を返します。

構文

SECOND(*datetime-expression* **)**

パラメータ

- **datetime-expression**　日時の値。

戻り値

SMALLINT

備考

指定した日時の秒に相当する 0 ～ 59 の数を返します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 25 を返します。

```
SELECT SECOND( '1998-07-13 21:21:25' );
```

SECONDS 関数 [日付と時刻]

この関数の動作は、指定した内容によって変わります。

- 1 つの日付を指定すると、0000-02-29 からの秒数を返します。

  注意
  0000-02-29 は実際の日付を指すための値ではありません。日付アルゴリズムで使用される日付です。

- 2 つのタイムスタンプを指定すると、2 つのタイムスタンプの間の秒数を整数で返します。代わりに、DATEDIFF 関数を使用します。
- 1 つの日付と整数を指定すると、指定したタイムスタンプに、指定した整数の秒数が加算されます。代わりに、DATEADD 関数を使用します。

構文 1：整数

SECONDS([*datetime-expression*,] *datetime-expression* **)**

構文 2：タイムスタンプ

SECONDS(*datetime-expression*, *integer-expression* **)**

パラメータ

- **datetime-expression**　日付と時刻。
- **integer-expression**　*datetime-expression* に加算する秒数。*integer-expression* が負の場合、日時の値から適切な分数が引かれます。整数式を指定する場合は、*datetime-expression* を datetime データ型として明示的にキャストしてください。

戻り値

INTEGER

TIMESTAMP

参照

- 「CAST 関数 [データ型変換]」 379 ページ
- 「DATEADD 関数 [日付と時刻]」 390 ページ
- 「DATEDIFF 関数 [日付と時刻]」 391 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 14400 を返します。これは、2 番目のタイムスタンプが、最初のタイムスタンプの 14400 秒後であることを示します。

```
SELECT SECONDS( '1999-07-13 06:07:12',
  '1999-07-13 10:07:12' );

SELECT DATEDIFF( second,
  '1999-07-13 06:07:12',
  '1999-07-13 10:07:12' );
```

次の文は、値 63062431632 を返します。

```
SELECT SECONDS( '1998-07-13 06:07:12' );
```

次の文は、日時 1999-05-12 21:05:120.000 を返します。

```
SELECT SECONDS( CAST( '1999-05-12 21:05:07'
AS TIMESTAMP ), 5);

SELECT DATEADD( second, 5, '1999-05-12 21:05:07' );
```

SHORT_PLAN 関数 [その他]

SQL 文の Ultra Light プランの最適化方法を短い記述の文字列で返します。この記述は、EXPLANATION 関数が返す記述と同じです。

構文

SHORT_PLAN(*string-expression* **)**

備考

クエリによっては、SQL Anywhere に選択したプランと Ultra Light の実行プランが異なる場合があります。

パラメータ

- **string-expression**　SQL 文。通常は SELECT 文ですが、UPDATE 文または DELETE 文も指定できます。

戻り値

LONG VARCHAR

参照

- 「EXPLANATION 関数 [その他]」 402 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、SELECT 文を文字列パラメータとして渡し、クエリを実行するためのプランを返します。

```
SELECT SHORT_PLAN(
  'SELECT * FROM Departments WHERE DepartmentID > 100' );
```

この情報は、追加するインデックスの決定や、良いパフォーマンスを得るためのデータベース構造の決定に役立ちます。

SIGN 関数 [数値]

数の符号を返します。

構文

SIGN(*numeric-expression* **)**

パラメータ

- **numeric-expression** 符号が返される数。

戻り値

SMALL INT

備考

負の数を指定すると、SIGN 関数は -1 を返します。

0 を指定すると、SIGN 関数は 0 を返します。

正の数を指定すると、SIGN 関数は 1 を返します。

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 -1 を返します。

```
SELECT SIGN( -550 );
```


SIMILAR 関数 [文字列]

2 つの文字列の類似性を示す数を返します。

構文

SIMILAR(*string-expression-1*, *string-expression-2* **)**

パラメータ

- **string-expression-1**　比較される最初の文字列。
- **string-expression-2**　比較される 2 番目の文字列。

戻り値

SMALL INT

備考

SIMILAR 関数は、2 つの文字列の類似性を表す 0 ～ 100 の整数を返します。結果は、2 つの文字列の文字が一致している割合として解釈できます。値が 100 の場合は、2 つの文字列は同じです。

この関数を使用して、名前 (顧客名など) のリストを修正できます。顧客がわずかに異なる名前で重複して登録されている場合があります。テーブルをそのテーブル自体とジョインし、90% より大きく、100% 未満の類似性をすべてレポートします。

SIMILAR 関数で実行される計算は、単に文字数が一致する場合よりも複雑になります。

参照

- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は値 75 を返します。2 つの値が 75 % 前後であることを示します。

```
SELECT SIMILAR( 'toast', 'coast' );
```

SIN 関数 [数値]

数のサインを返します。

構文

SIN(*numeric-expression* **)**

パラメータ

- **numeric-expression**　角度 (ラジアン)。

戻り値

DOUBLE

備考

SIN 関数は、引数のサインを返します。この引数はラジアン単位で表現される角度です。SIN 関数と ASIN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

参照

- 「ASIN 関数 [数値]」 375 ページ
- 「COS 関数 [数値]」 387 ページ
- 「COT 関数 [数値]」 387 ページ
- 「TAN 関数 [数値]」 454 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、0.52 の SIN 値を返します。

```
SELECT SIN( 0.52 );
```

SOUNDEX 関数 [文字列]

文字列の発音を表す数を返します。

構文

SOUNDEX(*string-expression* **)**

パラメータ

- **string-expression** 評価される文字列。

戻り値

SMALLINT

備考

文字列の SOUNDEX 関数の値は、先頭の文字とそれに続く H、Y、W 以外の 3 つの子音がベースになっています。文字列の最初の文字である場合を除き、*string-expression* の母音は無視されます。同じ文字が 2 つ続く場合は 1 文字としてカウントされます。たとえば、apple という単語は、文字 A、P、L、E がベースになります。

マルチバイト文字は、SOUNDEX 関数では無視されます。

完全とは言えませんが、SOUNDEX 関数は通常、類似発音で同じ文字で始まる語句に対して同じ数を返します。

SOUNDEX 関数は、英単語の処理に最も優れています。他の言語の処理は英語の場合よりも劣ります。

参照

- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、それぞれの名前の発音を表す 2 つの同じ数 3827 を返します。

```
SELECT SOUNDEX( 'Smith' ), SOUNDEX( 'Smythe' );
```

SPACE 関数 [文字列]

指定した数のスペースを返します。

構文

SPACE(*integer-expression* **)**

パラメータ

- **integer-expression** 返すスペースの数。

戻り値

LONG VARCHAR

備考

integer-expression が負の場合は、NULL 文字列を返します。

参照

- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、10 のスペースから成る文字列を返します。

```
SELECT SPACE( 10 );
```

SQRT 関数 [数値]

数の平方根を返します。

構文

SQRT(*numeric-expression* **)**

パラメータ

- **numeric-expression**　平方根が計算される数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 3 を返します。

```
SELECT SQRT( 9 );
```

STR 関数 [文字列]

指定した数に相当する文字列を返します。

構文

STR(*numeric-expression* [, *length* [, *decimal*]] **)**

パラメータ

- **numeric-expression**　-1E126 と 1E127 との間の任意の概数 (浮動小数点、実数、または倍精度)。
- **length**　返される文字数 (小数点、小数点の左右のすべての桁、ブランクを含む)。デフォルトは 10 です。
- **decimal**　返される小数点以下の桁数。デフォルトは 0 です。

戻り値

VARCHAR

備考

数の整数部分が指定した長さに合わない場合は、指定した長さの文字列がすべてアスタリスクで埋められて返されます。たとえば、次の文は *** を返します。

```
SELECT STR( 1234.56, 3 );
```

注意
サポートされている最大長は 128 です。長さが 1 ～ 128 の範囲外である場合は、結果が NULL になります。

参照

- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、6 スペースと 1235 (合計 10 文字) の文字列を返します。

```
SELECT STR( 1234.56 );
```

次の文は、結果 1234.6 を返します。

```
SELECT STR( 1234.56, 6, 1 );
```

STRING 関数 [文字列]

1 つ以上の文字列を連結して 1 つの長い文字列にします。

構文

STRING(*string-expression* **[, ...])**

パラメータ

- **string-expression**　評価される文字列。

  引数を 1 つだけ指定する場合は、1 つの式に変換されます。複数の引数を指定する場合は、連結されて 1 つの文字列になります。

戻り値

LONG VARCHAR

LONG NVARCHAR

LONG BINARY

備考

数値または日付をパラメータとして指定した場合は、文字列に変換されてから連結されます。また、1 つの式を唯一のパラメータとして指定すると、STRING 関数を使用して、その式を文字列に変換できます。

すべてのパラメータが NULL の場合、STRING は NULL を返します。NULL でないパラメータが存在すると、NULL パラメータはすべて空の文字列として処理されます。

参照

- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 *testing123* を返します。

```
SELECT STRING( 'testing', NULL, 123 );
```

STRTOUUID 関数 [文字列]

文字列の値をユニークな識別子 (UUID または GUID) の値に変換します。

> **新しいデータベースでは不要**
> バージョン 9.0.2 より前に作成されたデータベースでは、UUID 値のバイナリ表現と文字列表現の間を変換するための STRTOUUID 関数と UUIDTOSTR 関数が必要です。
>
> バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型がネイティブ・データ型に変更されています。これらのバージョンでは STRTOUUID 関数と UUIDTOSTR 関数を使用する必要はありません。
>
> 詳細については、「Ultra Light のデータ型」 328 ページを参照してください。

構文

STRTOUUID(*string-expression* **)**

パラメータ

- **string-expression** フォーマットが *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の文字列。

戻り値

UNIQUEIDENTIFIER

備考

フォーマットが *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の文字列をユニークな識別子の値に変換します。ここで、*x* は 16 進数です。

この関数は、UUID 値をデータベースに挿入するときに役立ちます。

参照

- 「UUIDTOSTR 関数 [文字列]」 460 ページ
- 「NEWID 関数 [その他]」 427 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

STUFF 関数 [文字列]

1 つの文字列から複数の文字を削除して、別の文字列に置き換えます。

構文

STUFF(*string-expression-1*, *start*, *length*, *string-expression-2* **)**

パラメータ

- **string-expression-1** STUFF 関数によって変更される文字列。
- **start** 削除を開始する文字の位置。文字列の先頭文字の位置を 1 とします。
- **length** 削除する文字数。
- **string-expression-2** 挿入する文字列。STUFF 関数を使用して文字列の一部を削除するには、NULL の置換文字列を使用します。

戻り値

LONG NVARCHAR

参照

- 「INSERTSTR 関数 [文字列]」 409 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 chocolate pie を返します。

```
SELECT STUFF( 'chocolate cake', 11, 4, 'pie' );
```

SUBSTRING 関数 [文字列]

文字列の部分文字列を返します。

構文

{ **SUBSTRING** | **SUBSTR** } **(** *string-expression*, *start*
[, *length*] **)**

パラメータ

- **string-expression**　部分文字列が返される文字列。
- **start**　文字単位で指定した、返される部分文字列の開始位置。
- **length**　文字単位で指定した、返される部分文字列の長さ。*length* を指定すると、部分文字列は指定した長さに限定されます。

戻り値

LONG BINARY

LONG VARCHAR

LONG NVARCHAR

備考

Ultra Light では、データベースに ansi_substring オプションがありませんが、デフォルトでは、SUBSTR 関数は ansi_substring が ON に設定されているように動作します。関数の動作は ANSI/ISO SQL/2003 の動作に対応します。

- **Start 値**　文字列の最初の文字の位置を 1 とします。負または 0 の開始オフセットは、文字列の左側が文字以外で埋められているように扱われます。
- **Length 値**　正の *length* は、部分文字列が開始位置の右側から *length* 文字で終わることを示します。

  負の *length* はエラーを返します。

  0 の *length* は、空の文字列を返します。

string-expression が binary データ型の場合、SUBSTRING 関数は BYTE_SUBSTR のように動作します。

文字列の末尾の文字を取得するには、RIGHT 関数を使用します。

入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値は文字長のセマンティックの観点から記述されます。

参照

- 「BYTE_SUBSTR 関数 [文字列]」 378 ページ
- 「LEFT 関数 [文字列]」 412 ページ
- 「RIGHT 関数 [文字列]」 437 ページ
- 「CHARINDEX 関数 [文字列]」 382 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　コア機能。

例

次の表は、SUBSTRING 関数から返される値を示します。

例	結果
SUBSTRING('front yard', 1, 4)	fron
SUBSTRING('back yard', 6, 4)	yard
SUBSTR('abcdefgh', 0, -2)	エラーを返します
SUBSTR('abcdefgh', -2, 2)	空の文字列を返します

SUM 関数 [集合]

ロー・グループごとに、指定された式の合計を返します。

構文 1

SUM(*expression* | **DISTINCT** *expression* **)**

パラメータ

- **expression**　合計するオブジェクト。通常はカラム名です。
- **DISTINCT 式**　入力の *expression* で一意の値の合計を計算します。

戻り値

INTEGER

DOUBLE

NUMERIC

備考

指定された式が NULL のローは含まれません。

グループにローが含まれていない場合は、NULL を返します。

参照

- 「COUNT 関数 [集合]」 388 ページ
- 「AVG 関数 [集合]」 377 ページ

標準と互換性

- **SQL/2003**　コア機能。

例

次の文は、値 3749146.740 を返します。

```
SELECT SUM( Salary )
FROM Employees;
```

SUSER_ID 関数 [システム]

指定したユーザ名の数値のユーザ ID を返します。

構文

SUSER_ID([*user-name*] **)**

パラメータ

- **user-name**　検索しているユーザ ID のユーザ名。

戻り値

INT

備考

user-name を指定しない場合は、現在のユーザの ID が返されます。

参照

- 「SUSER_NAME 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
- 「USER_ID 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』

標準と互換性

- **SQL/2003**　Transact-SQL 拡張。

例

次の文は、GROUPO ユーザの ID を返します。

```
SELECT SUSER_ID( 'GROUPO' );
```

SUSER_NAME 関数 [システム]

指定したユーザ ID のユーザ名を返します。

構文

SUSER_NAME([*user-id*] **)**

パラメータ

- **user-id**　検索しているユーザのユーザ ID。

戻り値

LONG VARCHAR

備考

user-id を指定しない場合は、現在のユーザのユーザ名が返されます。

参照

- 「SUSER_ID 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「USER_NAME 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』

標準と互換性

- **SQL/2003** Transact-SQL 拡張。

例

次の文は、ID 101 のユーザのユーザ名を返します。

```
SELECT SUSER_NAME( 101 );
```

SYNC_PROFILE_OPTION_VALUE 関数 [システム]

指定されたオプション名に対応するオプションの値を返します。

構文

SYNC_PROFILE_OPTION_VALUE(*profile_name*, *option_name***)**

パラメータ

- **profile_name** 調べる同期プロファイルの名前。
- **option_name** 値の取得対象であるオプションの名前。

戻り値

指定されたオプション名に対応するオプションの値を返します。

備考

ピリオドを含むオプション名を指定すると、ピリオドの前の部分を基本オプション名、ピリオドの後ろの部分をサブリスト・オプション名として、サブリストから値を取得します。

参照

- 「ML_GET_SERVER_NOTIFICATION [システム]」 423 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次のプロファイルの場合、以下のようになります。

```
MobilinkUid=joe;Stream=tcpip{host=sybase;port=1234};Ping=1
```

- **MobilinkUid** joe
- **Stream** tcpip{host=sybase;port=1234}
- **Stream.host** sybase
- **Stream.port** 1234
- **Ping** 1

TAN 関数 [数値]

数のタンジェントを返します。

構文

TAN(*numeric-expression* **)**

パラメータ

- **numeric-expression** 角度 (ラジアン)。

戻り値

DOUBLE

備考

ATAN 関数と TAN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

参照

- 「COS 関数 [数値]」 387 ページ
- 「SIN 関数 [数値]」 443 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、0.52 の tan 値を返します。

```
SELECT TAN( 0.52 );
```

TODAY 関数 [日付と時刻]

現在の日付を返します。

構文

TODAY(*)

戻り値

DATE

備考

従来の CURRENT DATE 関数の位置にこの構文を使用します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、システム・クロックによる現在の日付を返します。

```
SELECT TODAY( * );
SELECT CURRENT DATE;
```

TRIM 関数 [文字列]

先行ブランクと後続ブランクを文字列から削除します。

構文

TRIM(*string-expression* **)**

パラメータ

- **string-expression**　削除される文字列。

戻り値

VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

参照

- 「LTRIM 関数 [文字列]」 418 ページ
- 「RTRIM 関数 [文字列]」 438 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　TRIM 関数は SQL/2003 のコア機能です。

SQL Anywhere では、SQL/2003 で定義されている追加のパラメータ *trim specification* と *trim character* をサポートしていません。SQL Anywhere の TRIM の実装は、BOTH の TRIM 仕様に対応しています。

SQL/2003 規格で定義される TRIM 仕様 (LEADING と TRAILING) は、それぞれ SQL Anywhere の LTRIM 関数と RTRIM 関数から指定されます。

例

次の文は、先行ブランクと後続ブランクなしの値 chocolate を返します。

```
SELECT TRIM( '   chocolate   ' );
```

TRUNCNUM 関数 [数値]

指定した桁数で小数点以下を切り捨てます。

構文

{ **TRUNCNUM** | **"TRUNCATE"** }(*numeric-expression*, *integer-expression*)

パラメータ

- **numeric-expression**　トランケートされる数。
- **integer-expression**　正の整数は、丸めを行う小数点の右側の有効桁数を指定します。負の式は、丸めを行う小数点の左側の有効桁数を指定します。

戻り値

NUMERIC

備考

いずれかのパラメータが NULL 値の場合、結果は NULL 値になります。

参照

- 「ROUND 関数 [数値]」 438 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 600 を返します。

```
SELECT TRUNCNUM( 655, -2 );
```

次の文は、値 655.340 を返します。

```
SELECT TRUNCNUM( 655.348, 2 );
```


TSEQUAL 関数 [システム] (旧式)

2 つのタイムスタンプの値を比較し、これらが同じかどうかを返します。

構文

TSEQUAL (*timestamp1*, *timestamp2* **)**

パラメータ

- **timestamp1**　タイムスタンプ式。
- **timestamp2**　タイムスタンプ式。

戻り値

BIT

備考

TSEQUAL 関数は、WHERE 句でのみ使用できます。これは、UPDATE 文の一部として最も一般的に使用されます。

timestamp1 と *timestamp2* が同じ場合は、フェッチされた後にローが変わっています。ローが変わった場合は、そのタイプスタンプが変更されており、TSEQUAL 関数は FALSE を返します。TSEQUAL 関数が FALSE を返すと、アプリケーションはどのローも更新されていないと判断し、別のユーザによってローが変更されたと想定します。更新されたローは再フェッチされます。

TSEQUAL 関数を使用すると、ローがフェッチされた後に変更されているかどうかを判断できます。

参照

- 「timestamp カラムのデータ型」『SQL Anywhere サーバ - SQL の使用法』
- 「TIMESTAMP 特別値」『SQL Anywhere サーバ - SQL リファレンス』
- 「Transact-SQL の特殊な timestamp カラムとデータ型」『SQL Anywhere サーバ - SQL の使用法』

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

TIMESTAMP カラム Products.LastUpdated を作成し、ローが最後に更新された時刻のタイムスタンプを格納するとします。次の UPDATE 文は、TSEQUAL 関数を使用してローが更新されたかどうかを判断します。LastUpdated の値が '2010/12/25 11:08:34.173226' の場合、ローは更新されています。

```
UPDATE Products
SET Color = 'Yellow'
WHERE ID = '300'
AND TSEQUAL( LastUpdated, '2010/12/25 11:08:34.173226' );
```

UCASE 関数 [文字列]

文字列中のすべての文字を大文字に変換します。この関数は UPPER 関数と同じです。

構文

UCASE(*string-expression* **)**

パラメータ

- **string-expression**　大文字に変換される文字列。

戻り値

VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

参照

- 「UPPER 関数 [文字列]」 458 ページ
- 「LCASE 関数 [文字列]」 411 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 CHOCOLATE を返します。

```
SELECT UCASE( 'ChocoLate' );
```

UPPER 関数 [文字列]

文字列中のすべての文字を大文字に変換します。この関数は UCASE 関数と同じです。

構文

UPPER(*string-expression* **)**

パラメータ

- **string-expression**　大文字に変換される文字列。

戻り値

VARCHAR

NVARCHAR

LONG VARCHAR

LONG NVARCHAR

備考

UCASE 関数は UPPER 関数に似ています。

参照

- 「UCASE 関数 [文字列]」 458 ページ
- 「LCASE 関数 [文字列]」 411 ページ
- 「LOWER 関数 [文字列]」 418 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 CHOCOLATE を返します。

```
SELECT UPPER( 'ChocoLate' );
```

USER_ID 関数 [システム]

指定したユーザ名の数値のユーザ ID を返します。

構文

USER_ID([*user-name*] **)**

パラメータ

- **user-name** 検索しているユーザ ID のユーザ名。

戻り値

INT

備考

user-name を指定しない場合は、現在のユーザの ID が返されます。

参照

- 「USER_NAME 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「SUSER_ID 関数 [システム]」 『SQL Anywhere サーバ - SQL リファレンス』

標準と互換性

- **SQL/2003** Transact-SQL 拡張。

例

次の文は、GROUPO ユーザの ID を返します。

```
SELECT USER_ID( 'GROUPO' );
```

USER_NAME 関数 [システム]

指定したユーザ ID のユーザ名を返します。

構文

USER_NAME([*user-id*] **)**

パラメータ

- **user-id**　検索しているユーザのユーザ ID。

戻り値

LONG VARCHAR

備考

user-id を指定しない場合は、現在のユーザのユーザ名が返されます。

参照

- 「USER_ID 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』
- 「SUSER_NAME 関数 [システム]」『SQL Anywhere サーバ - SQL リファレンス』

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、ユーザ ID 101 のユーザ名を返します。

```
SELECT USER_NAME( 101 );
```

UUIDTOSTR 関数 [文字列]

ユニークな識別子の値 (UUID または GUID) を文字列の値に変換します。

新しいデータベースでは不要
バージョン 9.0.2 より前に作成されたデータベースでは、UUID 値のバイナリ表現と文字列表現の間を変換するための STRTOUUID 関数と UUIDTOSTR 関数が必要です。

バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型がネイティブ・データ型に変更されています。これらのバージョンでは STRTOUUID 関数と UUIDTOSTR 関数を使用する必要はありません。

詳細については、「Ultra Light のデータ型」 328 ページを参照してください。

構文

UUIDTOSTR(*uuid-expression* **)**

パラメータ

- **uuid-expression**　ユニークな識別子の値。

戻り値

VARCHAR

備考

ユニークな識別子を、フォーマットが *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の文字列値に変換します。ここで、x は 16 進数です。バイナリ値が有効な uniqueidentifier でない場合は、NULL を返します。

この関数は、UUID 値を表示する場合に役立ちます。

参照

- 「NEWID 関数 [その他]」 427 ページ
- 「STRTOUUID 関数 [文字列]」 448 ページ
- 「Ultra Light 文字列関数」 370 ページ

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、2 つのカラムを持つテーブル mytab を作成します。カラム pk は uniqueidentifier データ型で、カラム c1 は integer データ型です。それぞれに値 1 と値 2 を持つ 2 つのローをカラム c1 に挿入します。

```
CREATE TABLE mytab(
   pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
   c1 INT );
INSERT INTO mytab( c1 ) values ( 1 );
INSERT INTO mytab( c1 ) values ( 2 );
```

次の SELECT 文を実行すると、新規に作成したテーブルのすべてのデータを返します。

```
SELECT * FROM mytab;
```


2 つのカラムと 2 つのローを持ったテーブルが表示されます。カラム pk に表示される値は、バイナリ値です。

ユニークな識別子の値を読みやすい形式に変換するには、次のコマンドを実行します。

```
SELECT UUIDTOSTR(pk), c1 FROM mytab;
```

UUIDTOSTR 関数は、バージョン 9.0.2 以降で作成されたデータベースでは不要です。

WEEKS 関数 [日付と時刻]

2 つの日付の間の週数を返します。

構文 1

WEEKS([*datetime-expression*,] *datetime-expression* **)**

構文 2

WEEKS(*datetime-expression*, *integer-expression* **)**

パラメータ

- **datetime-expression**　日付と時刻。
- **integer-expression**　*datetime-expression* に加算する週数。*integer-expression* が負の場合、日時の値から適切な週数が引かれます。*integer-expression* を指定する場合は、*datetime-expression* を DATETIME として明示的にキャストしてください。

戻り値

構文 1 は、INTEGER を返します。

構文 2 は、TIMESTAMP を返します。

備考

1 つの日付を指定すると (構文 1)、WEEKS 関数は、0000-02-29 からの週数を返します。

2 つの日付を指定すると (構文 1)、WEEKS 関数は、2 つの日付の間の週数を返します。WEEKS 関数は DATEDIFF 関数に似ていますが、2 つの日付間の週数を計算する場合に使用される方法はこれと同じではなく、返される結果が異なる場合があります。WEEKS の戻り値は、2 つの日付間の日数を 7 で割った値を丸めて求められますが、DATEDIFF では週の境目の数が使用されます。このため、戻り値が異なるケースがあり得ます。たとえば、最初の日付が金曜日で次の日付が翌月曜日の場合、WEEKS 関数は週数として 0 を返し、DATEDIFF 関数は 1 を返します。どちらかが優れているという問題ではないので、WEEKS と DATEDIFF との動作の違いをふまえて使用してください。

DATEDIFF 関数の詳細については、「DATEDIFF 関数 [日付と時刻]」 391 ページを参照してください。

1 つの日付と整数を指定すると (構文 2)、WEEKS 関数は、指定された日付に、指定された整数の週数を加算します。この動作は DATEADD 関数と似ています。

DATEADD 関数の詳細については、「DATEADD 関数 [日付と時刻]」 390 ページを参照してください。

参照

データ型のキャストの詳細については、「CAST 関数 [データ型変換]」 379 ページを参照してください。

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の文は、値 8 を返します。これは、2008-09-13 10:07:12 が、2008-07-13 06:07:12 の 8 週間後であることを示します。

```
SELECT WEEKS( '2008-07-13 06:07:12',
  '2008-09-13 10:07:12' );
```

次の文は、値 104792 を返します。これは、指定された日付が、0000-02-29 の 104792 週間後であることを示します。

```
SELECT WEEKS( '2008-07-13 06:07:12' );
```

次の文は、タイムスタンプ 2008-06-16 21:05:07.0 を返します。これは、返された日時が 2008-05-12 21:05:07 の 5 週間後であることを示します。

```
SELECT WEEKS( CAST( '2008-05-12 21:05:07'
AS TIMESTAMP ), 5);
```

YEAR 関数 [日付と時刻]

タイムスタンプ値をパラメータとして受け取り、そのタイムスタンプで指定された年を返します。

構文

YEAR(*datetime-expression* **)**

パラメータ

- **datetime-expression** 日付、時刻、またはタイムスタンプ。

戻り値

SMALLINT

備考

値は SMALL INT として返されます。

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次の例は、値 2001 を返します。

```
SELECT YEAR( '2001-09-12' );
```

YEARS 関数 [日付と時刻]

2 つの日付を指定すると、2 つの日付の間の年数を整数で返します。この計算には DATEDIFF 関数を使用することをおすすめします。「DATEDIFF 関数 [日付と時刻]」 391 ページを参照してください。

1 つの日付を指定すると、その年が返されます。この計算には DATEPART 関数を使用することをおすすめします。「DATEPART 関数 [日付と時刻]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

1 つの日付と整数を指定すると、指定した日付に、指定した整数の年数が加算されます。この計算には DATEADD 関数を使用することをおすすめします。「DATEADD 関数 [日付と時刻]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

構文 1

YEARS([*datetime-expression*,] *datetime-expression* **)**

構文 2

YEARS(*datetime-expression*, *integer-expression* **)**

パラメータ

- **datetime-expression**　日付と時刻。
- **integer-expression**　*datetime-expression* に加算する年数。*integer-expression* が負の場合、日時の値から適切な年数が引かれます。*integer-expression* を指定する場合は、*datetime-expression* を datetime データ型として明示的にキャストしてください。

  データ型のキャストの詳細については、「CAST 関数 [データ型変換]」 379 ページを参照してください。

戻り値

構文 1 は、INTEGER を返します。構文 2 は、TIMESTAMP を返します。

備考

YEARS の値を求めるには、2 つの日付の間に年の最初の日がいくつあるかを計算します。

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文はどちらも -4 を返します。


```
SELECT YEARS( '1998-07-13 06:07:12',
          '1994-03-13 08:07:13' );

SELECT DATEDIFF( year,
  '1998-07-13 06:07:12',
  '1994-03-13 08:07:13' );
```

次の文は、1998 を返します。

```
SELECT YEARS( '1998-07-13 06:07:12' )
SELECT DATEPART( year, '1998-07-13 06:07:12' );
```

次の文は、指定した日付の 300 年後を返します。

```
SELECT YEARS( CAST( '1998-07-13 06:07:12' AS TIMESTAMP ), 300 )

SELECT DATEADD( year, 300, '1998-07-13 06:07:12' );
```

YMD 関数 [日付と時刻]

指定した年、月、日に相当する日付値を返します。値は -32768 ～ 32767 の small integer です。

構文

YMD(
integer-expression1,
integer-expression2,
integer-expression3 **)**

パラメータ

- **integer-expression1**　年。
- **integer-expression2**　月の数。月が 1 ～ 12 の範囲外である場合は、年が相応に調整されます。
- **integer-expression3**　日の数。任意の整数を指定でき、日付が相応に調整されます。

戻り値

DATE

標準と互換性

- **SQL/2003**　ベンダ拡張。

例

次の文は、値 1998-06-12 を返します。

```
SELECT YMD( 1998, 06, 12 );
```

値が通常の範囲から外れている場合、日付は相応に調整されます。たとえば、次の文は値 2000-03-01 を返します。

```
SELECT YMD( 1999, 15, 1 );
```

Ultra Light SQL 文

目次

Ultra Light SQL でサポートされている SQL 文は、SQL Anywhere データベースでサポートされている文のサブセットです。

始める前に

- Ultra Light のテーブルでは、所有者の概念がサポートされていません。既存の SQL と、プログラムで生成される SQL のために、Ultra Light では *owner.table-name* の構文が許可されます。ただし、テーブル所有者は Ultra Light ではサポートされていないので、所有者はチェックされません。
- Ultra Light SQL 文のマニュアルは、SQL Anywhere の文で使用されている構文と同じ表記規則に従っています。この表記規則について、またこの表記規則を使用して SQL 構文が表されている方法を理解している必要があります。「SQL 構文の表記規則」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。
- Ultra Light SQL を使用するとトランザクションが作成されます。トランザクションは、最後の ROLLBACK または COMMIT 後のすべての変更内容 (INSERT、UPDATE、DELETE) から構成されます。「Ultra Light でのトランザクション処理」 16 ページを参照してください。

  変更内容は、COMMIT を実行することによって確定できます。ROLLBACK 文を実行すると、変更内容が削除されます。「Ultra Light COMMIT 文」 479 ページと「Ultra Light ROLLBACK 文」 506 ページを参照してください。
- Interactive SQL で使用される文を探している場合は、「SQL 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。Interactive SQL で使用される文には、文の名前に **[Interactive SQL]** が続きます (例：「CONFIGURE 文 [Interactive SQL]」『SQL Anywhere サーバ - SQL リファレンス』)。

Ultra Light の文のカテゴリ

SQL 文は、文の最初の語で分類されています。この語はほとんど常に動詞です。このアクション指向型の構文では、言語の性質が、データベースに対する一連の命令文 (コマンド) になります。Ultra Light でサポートされている SQL 文は次のように分類できます。

- **データ検索文**　クエリとも呼ばれます。これらの文は、テーブルからデータ式のローを選択できます。データ検索は SELECT 文を使用して行います。「Ultra Light SELECT 文」 507 ページを参照してください。
- **データ操作文**　データベースの内容を変更できます。データ操作は次の文を使用して行います。
  - 「Ultra Light DELETE 文」 494 ページ
  - 「Ultra Light INSERT 文」 501 ページ
  - 「Ultra Light UPDATE 文」 517 ページ
- **データ定義文**　データベースの構造またはスキーマを定義できます。スキーマの変更には次の文を使用します。
  - 「Ultra Light ALTER DATABASE SCHEMA FROM FILE 文」 470 ページ
  - 「Ultra Light CREATE INDEX 文」 480 ページ
  - 「Ultra Light CREATE TABLE 文」 489 ページ
  - 「Ultra Light DROP INDEX 文」 495 ページ
  - 「Ultra Light DROP TABLE 文」 498 ページ
  - 「Ultra Light ALTER TABLE 文」 474 ページ
  - 「Ultra Light TRUNCATE TABLE 文」 514 ページ
- **トランザクション制御文**　Ultra Light アプリケーション内でトランザクションを制御できます。トランザクション制御には次の文を使用します。
  - 「Ultra Light CHECKPOINT 文」 478 ページ
  - 「Ultra Light COMMIT 文」 479 ページ
  - 「Ultra Light ROLLBACK 文」 506 ページ
- **同期管理**　Mobile Link サーバとの同期を一時的に制御できます。同期管理には次の文を使用します。
  - 「Ultra Light START SYNCHRONIZATION DELETE 文」 510 ページ
  - 「Ultra Light STOP SYNCHRONIZATION DELETE 文」 511 ページ
  - 「Ultra Light CREATE PUBLICATION 文」 482 ページ
  - 「Ultra Light ALTER PUBLICATION 文」 471 ページ
  - 「Ultra Light DROP PUBLICATION 文」 496 ページ

参照

- 「Ultra Light の式」 343 ページ
- 「Ultra Light の演算子」 356 ページ

Ultra Light ALTER DATABASE SCHEMA FROM FILE 文

この文は、SQL スクリプトを使用して既存の Ultra Light データベースのスキーマ定義を修正するために使用します。

構文

ALTER DATABASE SCHEMA FROM FILE *filename*

パラメータ

filename　既存の Ultra Light データベースのスキーマをアップグレードするときに使用する SQL スクリプトの名前とパスを定義します。

備考

スクリプトに必要な DDL 文を抽出するには、ulinit または ulunload のいずれかを使用します。これらのユーティリティを使用して、DDL 文が構文的に正しいことを確認してください。ulinit (-l *ログ・ファイル* オプション) または ulunload (-n -s *出力ファイル* オプション) を使用します。「Ultra Light データベース初期化ユーティリティ (ulinit)」 285 ページと「Ultra Light データベースのアンロード・ユーティリティ (ulunload)」 298 ページを参照してください。

この文は、データベースのバックアップを作成した後に実行します。

SQL スクリプト・ファイルの文字セットは、アップグレード対象のデータベースの文字セットに一致している必要があります。

この文の実行中は、デバイスをリセットしないでください。スキーマのアップグレード中にデバイスをリセットすると、Ultra Light データベースは使用できなくなります。

スキーマに適合しないローはすべて削除されます (たとえば、一意性制約が追加され、複数のローに同じ値が入っている場合、1 つのローだけを残して他のローはすべて削除されます)。この場合、SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE 警告が生成されます。この警告によってエラーを検出し、データベースをバックアップからリストアできます。

参照

- 「Ultra Light スキーマのアップグレードの配備」 70 ページ

例

次の文は、SQL スクリプト *MySchema.sql* を使用して、データベースのスキーマを修正します。

```
ALTER DATABASE SCHEMA FROM FILE 'MySchema.sql';
```


Ultra Light ALTER PUBLICATION 文

この文を使用して、パブリケーションを変更します。パブリケーションは、同期対象のリモート・データベース内のデータを識別します。

構文

ALTER PUBLICATION *publication-name alterpub-clause*

alterpub-clause :
ADD TABLE *table-name* [**WHERE** *search-condition*]
| **ALTER TABLE** *table-name* [**WHERE** *search-condition*]
| { **DROP** | **DELETE** } **TABLE** *table-name*
| **RENAME** *publication-name*

関連する動作

オートコミット。

参照

- 「Ultra Light の探索条件」 349 ページ
- 「Ultra Light での同期の設計」 139 ページ
- 「Ultra Light CREATE PUBLICATION 文」 482 ページ
- 「Ultra Light DROP PUBLICATION 文」 496 ページ
- 「Ultra Light START SYNCHRONIZATION DELETE 文」 510 ページ
- 「Ultra Light STOP SYNCHRONIZATION DELETE 文」 511 ページ

例

次の ALTER PUBLICATION 文は、Customer テーブルを pub_contact パブリケーションに追加します。

```
ALTER PUBLICATION pub_contact
  ADD TABLE Customers;
```


Ultra Light ALTER SYNCHRONIZATION PROFILE 文

この文は、Ultra Light 同期プロファイルを変更するために使用します。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。

構文

ALTER SYNCHRONIZATION PROFILE *sync-profile-name*
{ **REPLACE** | **MERGE** } *sync-option* [**;** ...]

sync-option :
sync-option-name **=** *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

パラメータ

- **sync-profile-name**　同期プロファイルの名前。
- **REPLACE 句**　この句を使用して、プロファイルに対して現在定義されているオプションを削除し、指定したオプションを代わりに追加します。
- **MERGE 句**　この句を使用して、既存のオプションを変更したり、同期プロファイルに新しいオプションを追加したりします。
- **sync-option**　1 つ以上の同期オプションの値ペアがセミコロンで区切られた文字列です。例：'option1=value1;option2=value2'
- **sync-option-name**　同期プロファイル・オプションの名前。
- **sync-option-value**　同期プロファイル・オプションの値。

備考

Ultra Light でサポートされる同期プロファイル・オプションのリストについては、「Ultra Light CREATE SYNCHRONIZATION PROFILE 文」 484 ページを参照してください。

関連する動作

なし

参照

- 「Ultra Light DROP SYNCHRONIZATION PROFILE 文」 497 ページ
- 「Ultra Light SYNCHRONIZE 文」 512 ページ

例

次の例は、ALTER SYNCHRONIZATION PROFILE...REPLACE 文を示します。

```
CREATE SYNCHRONIZATION PROFILE myProfile1;
ALTER SYNCHRONIZATION PROFILE myProfile1
  REPLACE 'publication=p1;uploadonly=on';
```

次の例は、ALTER SYNCHRONIZATION PROFILE...MERGE 文を示します。


```
CREATE SYNCHRONIZATION PROFILE myProfile2 'publication=p1;verbosity=high';
ALTER SYNCHRONIZATION PROFILE myProfile2
   MERGE'publication=p2;uploadonly=on';
```

Ultra Light ALTER TABLE 文

この文は、テーブル定義を修正するために使用します。

構文

ALTER TABLE *table-name* {
add-clause
| *modify-clause*
| *drop-clause*
| *rename-clause*
}

add-clause :
ADD { *column-definition* | *table-constraint* }

modify-clause :
ALTER *column-definition*

drop-clause :
DROP { *column-name* | **CONSTRAINT** *constraint-name* }

rename-clause :
RENAME {
new-table-name
| [*old-column-name* **TO**] *new-column-name*
| **CONSTRAINT** *old-constraint-name* **TO** *new-constraint-name* }

column-definition :
column-name data-type
[[**NOT**] **NULL**]
[**DEFAULT** *column-default*]
[**UNIQUE**]

column-default :
GLOBAL AUTOINCREMENT [**(** *number* **)**]
| **AUTOINCREMENT**
| **CURRENT DATE**
| **CURRENT TIME**
| **CURRENT TIMESTAMP**
| **NULL**
| **NEWID( )**
| *constant-value*

table-constraint :
[**CONSTRAINT** *constraint-name*]
{ *fkey-constraint* | *unique-key-constraint* }
[**WITH MAX HASH SIZE** *integer*]

fkey-constraint :
[**NOT NULL**] **FOREIGN KEY** [*role-name*] **(** *ordered-column-list* **)**
REFERENCES *table-name* **(** *column-name*, ... **)**
[**CHECK ON COMMIT**]

unique-key-constraint :
UNIQUE (*ordered-column-list* **)**

ordered-column-list :
(*column-name* [**ASC** | **DESC**], ... **)**

パラメータ

add-clause　テーブルに新しいカラム制約またはテーブル制約を追加します。

- **ADD *column-definition* 句**　テーブルに新しいカラムを追加します。カラムにデフォルト値がある場合は、新しいカラムのすべてのローにそのデフォルト値が移植されます。この句のキーワードとサブ句の詳細については、「Ultra Light CREATE TABLE 文」 489 ページを参照してください。
- **ADD *table-constraint* 句**　テーブルに制約を追加します。オプションの制約名を使用すると、テーブル全体の制約を修正することなく、後で個々の制約を修正したり削除したりできます。この句のキーワードとサブ句の詳細については、「Ultra Light CREATE TABLE 文」 489 ページを参照してください。

> **注意**
> Ultra Light では、プライマリ・キーを追加することはできません。

modify-clause　1 つのカラム制約を変更します。*column-definition* が ALTER 文の一部の場合には、プライマリ・キーは使用できません。*column-definition* の詳細については、「Ultra Light CREATE TABLE 文」 489 ページを参照してください。

drop-clause　カラム制約またはテーブル制約を削除します。

- **DROP *column-name***　テーブルからカラムを削除します。カラムがインデックス、一意性制約、外部キー、またはプライマリ・キーに含まれている場合は、それらのオブジェクトを削除**しないと**、Ultra Light がカラムを削除できません。
- **DROP CONSTRAINT *table-constraint***　テーブル定義から指定された制約を削除します。*table-constraint* の詳細については、「Ultra Light CREATE TABLE 文」 489 ページを参照してください。

> **注意**
> Ultra Light では、プライマリ・キーを削除することはできません。

rename-clause　テーブル名、カラム名、制約名を変更します。

- **RENAME *new-table-name***　テーブルの名前を *new-table-name* に変更します。古いテーブル名を使用しているアプリケーションを修正する必要があることに注意してください。古いテーブル名を自動的に割り当てられた外部キーは、名前を変更しません。
- **RENAME *old-column-name* TO *new-column-name***　カラムの名前を *new-column-name* に変更します。古いカラム名を使用しているアプリケーションを修正する必要があることに注意してください。

- **RENAME *old-constraint-name* TO *new-constraint-name*** 制約の名前を *new-constraint-name* に変更します。古い制約名を使用しているアプリケーションを修正する必要があることに注意してください。

> **注意**
> Ultra Light では、プライマリ・キーの名前を変更することはできません。

column-constraint カラム制約は、データベース内のデータの整合性を保つために、カラムに格納できる値を制限します。カラム制約は UNIQUE である必要があります。

UNIQUE テーブル内の各ローをユニークに識別する 1 つまたは複数のカラムを識別します。テーブル内の異なるローが、指定されているすべてのカラムで同じ値を持つことはできません。1 つのテーブルに複数の一意性制約が存在することがあります。

備考

1 つの ALTER TABLE 文では、1 つの *table-constraint* または *column-constraint* だけを追加、変更、または削除できます。

役割名は外部キーの名前です。*role-name* の主な機能は、同じテーブルに対する 2 つの外部キーを区別することです。また、CONSTRAINT *constraint-name* を使用して外部キーに名前を付けることができます。ただし、両方のメソッドを使用して外部キーに名前を付けないでください。

テーブルまたはカラムの制約は MODIFY (変更) できません。制約を変更するには、古い制約を DELETE (削除) し、新しい制約を ADD (追加) します。

名前の末尾が **nosync** のテーブルは、末尾が **nosync** のテーブル名に変更する必要があります。「Ultra Light の nosync テーブル」 140 ページを参照してください。

テーブルを対象とした文が、別の要求やクエリですでに参照されている場合、ALTER TABLE は実行できません。同様に、テーブルの変更中は、そのテーブルを参照する要求は処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は ALTER TABLE を実行できません。

Ultra Light.NET を使用している場合、すべてのデータ・オブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。「Dispose メソッド」『Ultra Light - .NET プログラミング』を参照してください。

スキーマの変更が同時に開始された場合、文は解放されません。「DDL 文を使用したスキーマ変更」 10 ページを参照してください。

参照

- 「Ultra Light CREATE TABLE 文」 489 ページ
- 「Ultra Light DROP TABLE 文」 498 ページ
- 「Ultra Light のデータ型」 328 ページ
- 「テーブルの変更」 『SQL Anywhere サーバ - SQL の使用法』
- 「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』
- 「オートインクリメント・カラムの分割サイズの上書き」 138 ページ
- 「最後に割り当てられた GLOBAL AUTOINCREMENT の値の割り出し」 137 ページ

例

次の文は、MyEmployees という架空のテーブルから Street カラムを削除します。

```
ALTER TABLE MyEmployees
DROP Street;
```

次の例は、MyCustomers という架空のテーブルの Street カラムを変更して、約 50 文字を格納するようにします。

```
ALTER TABLE MyCustomers
MODIFY Street CHAR(50);
```

Ultra Light CHECKPOINT 文

この文は、データベースにチェックポイントを実行させるために使用します。

構文

CHECKPOINT

備考

CHECKPOINT 文をコミット・フラッシュのトリガとして使用できます。コミット・フラッシュが実行されると、コミットされていないトランザクションが記憶領域に書き込まれます。

Embedded SQL API を使用している場合は、ULCheckpoint メソッドを使用することもできます。C++ コンポーネント・アプリケーションを作成している場合は、接続オブジェクトで Checkpoint メソッドを使用することもできます。その他すべての API では、この文を使用する必要があります。

関連する動作

この文によって、保留中のコミットされたトランザクションが記憶領域にすべてフラッシュされますが、現在のトランザクションはコミットまたはフラッシュされません。

参照

- 「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」 125 ページ
- 「Ultra Light COMMIT 文」 479 ページ
- 「Ultra Light COMMIT_FLUSH 接続パラメータ」 239 ページ
- Ultra Light for 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』

例

次の文は、データベースのチェックポイントを実行します。

```
CHECKPOINT;
```


Ultra Light COMMIT 文

この文は、データベースの変更を確定するために使用します。

構文

COMMIT [WORK]

備考

Ultra Light SQL を使用するとトランザクションが作成されます。トランザクションは、最後の ROLLBACK または COMMIT 後のすべての変更内容 (INSERT、UPDATE、DELETE) から構成されます。COMMIT 文は、現在のトランザクションを終了し、このトランザクション中に行われたすべての変更内容をデータベースに永続化します。

ALTER、CREATE、DROP の各文を使用したデータベース・オブジェクトの変更は、自動的にコミットされます。

参照

- 「Ultra Light CHECKPOINT 文」 478 ページ
- 「Ultra Light ROLLBACK 文」 506 ページ

例

次の文は、現在のトランザクションで行われた変更内容をデータベースに永続化します。

```
COMMIT;
```

Ultra Light CREATE INDEX 文

この文は、指定されたテーブル上にインデックスを作成するために使用します。

構文

CREATE [**UNIQUE**] **INDEX** [*index-name*]
ON *table-name* **(** *ordered-column-list* **)**
[**WITH MAX HASH SIZE** *integer*]

ordered-column-list :
(*column-name* [**ASC** | **DESC**], ... **)**

パラメータ

UNIQUE UNIQUE パラメータによって、インデックス内のすべてのカラムで同じ値を持つローがテーブル内に複数存在しないようにします。各インデックス・キーはユニークであるか、少なくとも 1 つのカラムで NULL を持つ必要があります。

テーブル上の一意性制約とユニーク・インデックスの間には違いがあります。ユニーク・インデックスのカラムは NULL を使用できますが、一意性制約のカラムには使用できません。また、外部キーは、プライマリ・キー、または一意性制約があるカラムを参照できます。ただし、外部キーは、ユニーク・インデックスを参照できません。

ordered-column-list カラムの順序リストです。インデックスのカラム値は、昇順または降順にソートできません。

WITH MAX HASH SIZE このインデックスのハッシュ・サイズをバイト単位で設定します。この値は、データベースに対して有効なデフォルトの MaxHashSize プロパティを上書きします。デフォルトのサイズの詳細については、「Ultra Light データベース・プロパティへのアクセス」 226 ページを参照してください。

備考

Ultra Light では、プライマリ・キーと一意性制約のインデックスは自動的に作成されます。

インデックスによって、Ultra Light が特定のローを検索しやすくし、クエリのパフォーマンスを向上させることができます。ただし、インデックスを管理する必要があるので、同期や INSERT、DELETE、UPDATE の各文の処理は低速になる可能性があります。

インデックスは、データベースに対して発行されたクエリのパフォーマンスを向上させたり、ORDER BY 句を使用してクエリをソートするときに自動的に使用されます。インデックスはいったん作成されると、DROP INDEX を使用して削除するときを除いて、SQL 文では二度と参照されません。

インデックスはデータベース内の領域を占有します。また、インデックスの管理に必要な追加作業は、データ修正操作のパフォーマンスに影響する場合があります。このため、クエリのパフォーマンスが向上しないインデックスの作成は避けてください。

CREATE INDEX 文の処理中は、そのインデックスを参照する要求やクエリは処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は CREATE INDEX を実行できません。

Ultra Light では、実行プランを使用してクエリを最適化することもできます。「Ultra Light の実行プラン」 360 ページを参照してください。

Ultra Light.NET を使用している場合、すべてのデータ・オブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。「Dispose メソッド」『Ultra Light - .NET プログラミング』を参照してください。

スキーマの変更が同時に開始された場合、文は解放されません。「DDL 文を使用したスキーマ変更」 10 ページを参照してください。

関連する動作

- オートコミット。

参照

- 「Ultra Light のパフォーマンスと最適化」 113 ページ
- 「Ultra Light DROP INDEX 文」 495 ページ
- 「Ultra Light のインデックスの操作」 84 ページ

例

次の文は、Employees テーブルで 2 カラムのインデックスを作成します。

```
CREATE INDEX employee_name_index
ON Employees ( Surname, GivenName );
```

次の文は、ProductID カラム用に SalesOrderItems テーブル上でインデックスを作成します。

```
CREATE INDEX item_prod
ON SalesOrderItems ( ProductID );
```


Ultra Light CREATE PUBLICATION 文

この文は、パブリケーションを作成するために使用します。パブリケーションは、Ultra Light リモート・データベース内の同期されたデータを示します。

構文

CREATE PUBLICATION *publication-name*
(**TABLE** *table-name* [**WHERE** *search-condition*], ... **)**

パラメータ

TABLE 句　このテーブルは、パブリケーションに TABLE を含めるときに使用します。TABLE 句の数に制限はありません。

WHERE 句　WHERE 句を指定した場合、同期時に、*search-condition* を満たすローだけが、関連付けられたテーブルからのアップロード対象になります。「Ultra Light の探索条件」 349 ページ を参照してください。

WHERE 句を指定しなかった場合は、最後の同期後にテーブル内で変更されたすべてのローがアップロード対象になります。

備考

パブリケーションは、単一の同期操作で同期されるテーブルを設定し、Mobile Link サーバにアップロードされるデータを決定します。Mobile Link サーバのダウンロード・セッション中には、これらのテーブルだけのローが送り返される可能性がありますが、ダウンロードされるローは、テーブルの WHERE 句を満たす必要はありません。

テーブル全体だけをパブリッシュできます。Ultra Light では、テーブルの特定のカラムはパブリッシュできません。

関連する動作

- オートコミット。

参照

- 「Ultra Light クライアント」 131 ページ
- 「Ultra Light DROP PUBLICATION 文」 496 ページ
- 「Ultra Light ALTER PUBLICATION 文」 471 ページ
- 「Ultra Light の探索条件」 349 ページ

例

次の文は、2 つのテーブルのすべてのカラムとローをパブリッシュします。

```
CREATE PUBLICATION pub_contact (
   TABLE Contacts,
   TABLE Customers
);
```

次の文は、Customers テーブルで、State カラムに MN が含まれているローのみをパブリッシュします。


```
CREATE PUBLICATION pub_customer (
   TABLE Customers
   WHERE State = 'MN'
);
```

Ultra Light CREATE SYNCHRONIZATION PROFILE 文

この文は、Ultra Light 同期プロファイルを作成するために使用します。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。

構文

CREATE SYNCHRONIZATION PROFILE *sync-profile-name sync-option* [;...]

sync-option :
sync-option-name **=** *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

パラメータ

- **sync-profile-name** 同期プロファイルの名前。
- **sync-option** 1 つ以上の同期オプションの値ペアがセミコロンで区切られた文字列です。たとえば、'option1=value1;option2=value2' のように記述します。
- **sync-option-name** 同期プロファイル・オプションの名前。
- **sync-option-value** 同期プロファイル・オプションの値。

備考

既存の同期プロファイルを変更する方法は 2 通りあります。最初の方法では、REPLACE 句を使用します。この場合、同期プロファイルの内容は、新しい sync-option 文字列に含まれている内容に置換されます。この方法は、同期プロファイルを削除してから、同じ名前のプロファイルを作成する方法と同じであり、新しい文字列を使用する点が異なっています。ただし、同期の実行時にパラメータをマージ (または上書き) できるため、同期プロファイルに同期定義が完全に含まれている必要はないことに注意してください。

同期プロファイルを変更するための 2 番目の方法では、MERGE 句を使用します。この句を使用すると、MERGE 句で指定した sync オプションのみが変更されます。同期プロファイルから sync オプションを削除するには、'option1=;' のような sync-option 文字列を作成します (オプションの値を空に設定します)。

同期プロファイル・オプション STREAM は、その値にサブリストが含まれているため他のオプションとは異なっています (例 : 'STREAM=TCPIP{host=192.168.1.1;port=1234}')。この場合、'host=192.168.1.1;port=1234' がサブリストになります。サブリストの値を追加または削除するには、STREAM の sync-option-name と sub-option-name の間にピリオドを使用します。たとえば、MERGE 'stream.port=5678;stream.host=;compression=zlib' のようにすると、同期プロファイルは stream=TCPIP{port=5678;compression=zlib} になります。新しい値にストリームを設定しようとすると、ストリームの値全体が置換されます。たとえば、MERGE 'stream=HTTPS' のようにすると、同期プロファイルは stream=HTTPS{} になります。

次の表に、Ultra Light でサポートされる同期プロファイル・オプションを示します。

同期プロファイル・オプション	有効な値	説明
AllowDownloadDupRows	ブール式	このオプションを指定すると、プライマリ・キーの同じローが複数ダウンロードされたときにエラーが発生しません。このオプションによって、同期を失敗させることなく、不整合なデータを同期できます。デフォルト値は "no" です。「Additional Parameters 同期パラメータ」 159 ページを参照してください。
AuthParms	文字列 (カンマ区切り)	Mobile Link サーバに送信される認証パラメータのリストを指定します。認証パラメータを使用すると、Mobile Link スクリプトでカスタム認証を実行できます。「Authentication Parameters 同期パラメータ」 161 ページを参照してください。
CheckpointStore	ブール式	同期中にデータベースのチェックポイントを追加して、同期プロセス中のデータベースの拡張を制限します。「Additional Parameters 同期パラメータ」 159 ページを参照してください。
ContinueDownload	ブール式	以前に失敗したダウンロードを再起動します。ダウンロードを続行すると、失敗した同期でダウンロードするように選択されていた変更のみを受信します。デフォルトでは、Ultra Light はダウンロードを続行しません。「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。
DisableConcurrency	ブール式	この同期中は他のスレッドからデータベースへのアクセスを禁止します。「Additional Parameters 同期パラメータ」 159 ページを参照してください。
DownloadOnly	ブール式	ダウンロード専用同期を実行します。「Download Only 同期パラメータ」 163 ページを参照してください。
KeepPartialDownload	ブール式	接続エラーが発生した場合に、部分的なダウンロードを維持するかどうかを制御します。デフォルトでは、Ultra Light は、部分的にダウンロードされた変更をロールバックしません。「Keep Partial Download 同期パラメータ」 165 ページを参照してください。
MobiLinkPwd	文字列	ユーザ名に対する既存の Mobile Link パスワードを指定します。「MobiLinkPwd (mp) 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。

同期プロファイル・オプション	有効な値	説明
MobiLinkUid	文字列	Mobile Link ユーザ名を指定します。「-u オプション」『Mobile Link - クライアント管理』を参照してください。「-mn オプション」『Mobile Link - クライアント管理』を参照してください。
NewMobiLinkPwd	文字列	Mobile Link ユーザの新しいパスワードを指定します。このオプションは、既存のパスワードを変更する場合に使用します。「-mn オプション」『Mobile Link - クライアント管理』を参照してください。
Ping	ブール式	サーバとの通信の確認のみを行います。同期は実行しません。「Ping 同期パラメータ」 169 ページを参照してください。
Publications	文字列 (カンマ区切り)	同期させるパブリケーションを指定します。パブリケーションによって、同期に関与するリモート上のテーブルが決定されます。このパラメータが空白 (デフォルト) の場合は、すべてのテーブルが同期されます。このパラメータがアスタリスク (*) の場合は、すべてのパブリケーションが同期されます。「Ultra Light のパブリケーション」 141 ページを参照してください。
ScriptVersion	文字列	Mobile Link スクリプトのバージョンを指定します。スクリプト・バージョンは、同期中に統合データベースの Mobile Link によって実行されるスクリプトを決定します。スクリプト・バージョンを指定しない場合、「デフォルト」が使用されます。「ScriptVersion (sv) 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。
SendColumnNames	文字列	同期時にカラム名がアップロード・ファイルの一部として Mobile Link サーバに送信されるように指定します。デフォルトでは、カラム名は送信されません。「Send Column Names 同期パラメータ」 172 ページを参照してください。
SendDownloadACK	ブール式	クライアントからサーバにダウンロード確認が送信されるように指定します。デフォルトでは、Mobile Link サーバはダウンロード確認を提供しません。「Send Download Acknowledgement 同期パラメータ」 173 ページを参照してください。

同期プロファイル・オプション	有効な値	説明
Stream	文字列 (サブリスト付き)	Mobile Link のネットワーク同期プロトコルを指定します。「Stream Type 同期パラメータ」 175 ページを参照してください。
TableOrder	文字列 (カンマ区切り)	アップロードでのテーブルの順序を指定します。デフォルトでは、Ultra Light は、外部キーの関係に基づいて順序を選択します。「Additional Parameters 同期パラメータ」 159 ページを参照してください。
UploadOnly	文字列	同期にはアップロードのみが含まれ、ダウンロードは発生されないよう指定します。「Upload Only 同期パラメータ」 178 ページを参照してください。

ブール値に指定できるのは、Yes/No、1/0、True/False、On/Off です。いずれのブールでも、デフォルトは "No" です。他のすべての値では、デフォルトは単純に未指定です。

関連する動作

なし

参照

- 「Ultra Light ALTER SYNCHRONIZATION PROFILE 文」 472 ページ
- 「Ultra Light DROP SYNCHRONIZATION PROFILE 文」 497 ページ
- 「Ultra Light SYNCHRONIZE 文」 512 ページ

例

次の文は、Test1 という同期プロファイルを作成します。

```
CREATE SYNCHRONIZATION PROFILE Test1
'MobiLinkUid=mary;Stream=TCPIP{host=192.168.1.1;port=1234}'
```

次の例は、別のオプションで一連の ALTER SYNCHRONIZATION PROFILE コマンドを実行した後に発生する変更を示します。

myProfile1='MobiLinkUID=mary;ScriptVersion=default' とします。

ALTER SYNCHRONIZATION PROFILE myProfile1 REPLACE
'MobiLinkPwd=sql;ScriptVersion=1' を実行した後、myProfile1 は
'MobiLinkPwd=sql;ScriptVersion=1' になります。

ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE
'MobiLinkUID=mary;STREAM=tcpip' を実行した後、myProfile1 は
'MobiLinkPwd=sql;ScriptVersion=1;MobiLinkUID=mary;STREAM=tcpip' になります。

ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'MobiLinkUID=;STREAM.host=192.168.1.1;STREAM.port=1234;ScriptVersion=;' を実行した後、myProfile1 は 'MobiLinkPwd=sql;STREAM=tcpip{192.168.1.1;port=1234}' になります。

ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'MobiLinkPwd=;Ping=yes;STREAM=HTTP' を実行した後、myProfile1 は 'Ping=yes;STREAM=HTTP' になります。

ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'STREAM=HTTP{host=192.168.1.1}' を実行した後、myProfile1 は 'Ping=yes;STREAM=HTTP{host=192.168.1.1}' になります。

Ultra Light CREATE TABLE 文

この文は、テーブルを作成するために使用します。

構文

CREATE TABLE *table-name* **(**
{ *column-definition* | *table-constraint* }, ...
)

column-definition :
column-name data-type
[[**NOT**] **NULL**]
[**DEFAULT** *column-default*]
[*column-constraint*]

column-default :
GLOBAL AUTOINCREMENT [**(** *number* **)**]
| **AUTOINCREMENT**
| **CURRENT DATE**
| **CURRENT TIME**
| **CURRENT TIMESTAMP**
| **NULL**
| **NEWID( )**
| *constant-value*

column-constraint :
PRIMARY KEY
| **UNIQUE**

table-constraint :
{ [**CONSTRAINT** *constraint-name*]
pkey-constraint
| *fkey-constraint*
| *unique-key-constraint* }
[**WITH MAX HASH SIZE** *integer*]

pkey-constraint :
PRIMARY KEY [*ordered-column-list*]

fkey-constraint :
[**NOT NULL**] **FOREIGN KEY** [*role-name*] **(** *ordered-column-list* **)**
REFERENCES *table-name* **(** *column-name*, ... **)**
[**CHECK ON COMMIT**]

unique-key-constraint :
UNIQUE (*ordered-column-list* **)**

ordered-column-list :
(*column-name* [**ASC** | **DESC**], ... **)**

パラメータ

column-definition テーブル内のカラムを定義します。この句には、次のパラメータを使用できます。

- **column-name**　カラム名は識別子です。同じテーブル内の 2 つのカラムが同じ名前を持つことはできません。「Ultra Light の識別子」 319 ページを参照してください。
- **data-type**　カラムのデータ型です。「Ultra Light のデータ型」 328 ページを参照してください。
- **[NOT] NULL**　NOT NULL が指定されているか、カラムが PRIMARY KEY または UNIQUE 制約下にある場合、カラムはいずれのローでも NULL を持つことはできません。それ以外の場合は、NULL は許可されます。
- **column-default**　カラムのデフォルト値を設定します。DEFAULT 値を指定する場合、カラムの値を指定しない INSERT 文のカラムの値としてこのデフォルト値が使用されます。INSERT 文はカラムの値を指定しません。DEFAULT を指定しない場合、DEFAULT NULL と等しくなります。デフォルトのオプションを以下に示します。
  - **AUTOINCREMENT**　AUTOINCREMENT を使用する場合、カラムは整数データ型の 1 つ、または真数値型にします。テーブルに挿入する場合、AUTOINCREMENT カラムの値を指定しないと、カラム内の任意の値より大きいユニーク値が生成されます。INSERT で、カラムの現在の最大値より大きい値を指定した場合、この値が後続の挿入処理の開始ポイントとして使用されます。

    ヒント
    Ultra Light では、テーブルが作成された時点でのオートインクリメントの初期値は 0 ではありません。カラムに符号付きデータ型が指定されている場合は、オートインクリメントによって負の値が生成されます。このため、オートインクリメントを適用するカラムを符号なし整数として宣言し、負の値が生成されないようにしてください。

  - **GLOBAL AUTOINCREMENT**　ドメインが分割されるという点を除いて、AUTOINCREMENT と同じです。各分割には同じ数の値が含まれます。データベースの各コピーにユニークなグローバル・データベース ID 番号を割り当てます。Ultra Light では、データベースのデフォルト値は、そのデータベース番号でユニークに識別された分割からのみ設定されます。「Ultra Light での GLOBAL AUTOINCREMENT の使用」 135 ページと「Ultra Light global_database_id オプション」 231 ページを参照してください。
  - **[NOT] NULL**　カラムに NULL を格納できるかどうかを制御します。
  - **NEWID()**　ユニークな識別子の値を生成する関数です。「NEWID 関数 [その他]」 427 ページを参照してください。
  - **CURRENT TIMESTAMP**　CURRENT DATE と CURRENT TIME を結合して、TIMESTAMP 値を形成します。年、月、日、時、分、秒、秒の小数位で構成されます。秒は小数第 3 位まで格納されます。精度はシステム・クロックの精度によって制限されます。「CURRENT TIMESTAMP 特別値」 325 ページを参照してください。
  - **CURRENT DATE**　現在の年、月、日を格納します。「CURRENT DATE 特別値」 324 ページを参照してください。
  - **CURRENT TIME**　現在の時、分、秒 (小数位あり) で構成される時刻を格納します。「CURRENT TIME 特別値」 324 ページを参照してください。
  - **constant-value**　カラムのデータ型の定数です。通常、この定数は数値または文字列です。

- **column-constraint 句**　カラム制約を指定して、カラムに許容される値を制限します。カラム制約は次のいずれかです。
  - **PRIMARY KEY**　*column-constraint* の一部として設定するときに、PRIMARY KEY 句はテーブルのプライマリ・キーとしてカラムを設定します。プライマリ・キーは、テーブル内の各ローをユニークに識別します。デフォルトでは、プライマリ・キーに含まれるカラムには NULL を使用できません。
  - **UNIQUE**　テーブル内の各ローをユニークに識別する 1 つまたは複数のカラムを識別します。テーブル内の異なるローが、指定されているすべてのカラムで同じ値を持つことはできません。1 つのテーブルに複数の一意性制約が存在することがあります。NULL 値は使用できません。

table-constraint 句　テーブル制約を指定して、テーブル内の 1 つまたは複数のカラムに格納できる値を制限します。CONSTRAINT 句を使用して、テーブル制約の識別子を指定します。以下に定義するように、テーブル制約は、プライマリ・キー制約、外部キー制約、一意性制約のいずれかの形式にできます。

- **pkey-constraint 句**　テーブルのプライマリ・キーとして、指定したカラムを設定します。プライマリ・キーは、テーブル内の各ローをユニークに識別します。プライマリ・キーに含まれるカラムには NULL を使用できません。
- **fkey-constraint 句**　外部キー制約を指定して、1 つまたは複数のカラムの値を、別のテーブルのプライマリ・キー (または一意性制約) の値と一致するように制限します。
  - **NOT NULL 句**　NOT NULL を指定して、外部キー・カラムを NULL 入力不可にします。外部キー内の NULL は、外部テーブルのこのローに該当するローがプライマリ・テーブル内にないことを意味します。複数カラムの外部キー内の少なくとも 1 つの値が NULL である場合、キーの他のカラムに保持できる値に関する制限はありません。
  - **role-name 句**　*role-name* を指定して、外部キーに名前を付けます。*role-name* は、同じテーブル内で外部キーを区別する場合に使用します。また、CONSTRAINT *constraint-name* を使用して外部キーに名前を付けることができます。ただし、両方のメソッドを使用して外部キーに名前を付けないでください。
  - **REFERENCES 句**　REFERENCES 句を指定し、プライマリ・テーブルで、外部キー制約として使用する 1 つまたは複数のカラムを定義します。REFERENCES カラム制約に *column-name* を指定する場合、一意性制約またはプライマリ・キーの制約を受ける、プライマリ・テーブルのカラム名を指定します。
  - **CHECK ON COMMIT**　データベース・サーバで、COMMIT を待ってから外部キー制約を実行するように、CHECK ON COMMIT を指定します。デフォルトでは、挿入、更新、削除の各操作中に、外部キー制約がただちに実行されます。ただし、CHECK ON COMMIT を設定すると、データベースの変更が外部キー制約に違反している場合でも、次の COMMIT 前にデータの不整合が解決されればデータベースの変更を任意の順序で実行できます。
- **unique-key-constraint 句**　一意性制約を指定して、テーブル内の各ローをユニークに識別する 1 つ以上のカラムを指定します。テーブル内の異なるローが、指定されているすべてのカラムで同じ値を持つことはできません。1 つのテーブルに複数の一意性制約が存在することがあります。

- **WITH MAX HASH SIZE**　このインデックスのハッシュ・サイズをバイト単位で設定します。この値は、データベースに対して有効なデフォルトの MaxHashSize プロパティを上書きします。デフォルトのサイズの詳細については、「Ultra Light データベース・プロパティへのアクセス」 226 ページを参照してください。

備考

通常、カラム制約を使うのは、制約がテーブル内で複数の他カラムを参照しない場合です。複数のカラムを参照する場合は、テーブル制約を使用します。この文によって制約違反が発生した場合、文の実行は完了しません。エラーの検出前に文が行った変更が完了していない場合は、エラーがレポートされます。

テーブル内の各ローは、ユニークなプライマリ・キー値を持ちます。

役割名を指定しない場合、役割名は以下のように設定されます。

1. テーブル名と同じ役割名を含む外部キーが存在しない場合、テーブル名が役割名として割り当てられます。
2. テーブル名がすでに使用されている場合、役割名は、0 が埋め込まれた、テーブルに固有の 3 桁の数字と結合されたテーブル名になります。

スキーマの変更　スキーマの変更が同時に開始された場合、文は解放されません。「DDL 文を使用したスキーマ変更」 10 ページを参照してください。

CREATE TABLE 文の処理中は、そのテーブルを参照する要求やクエリは処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は CREATE TABLE を実行できません。

Ultra Light.NET ユーザの場合、すべてのデータ・オブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。「Dispose メソッド」 『Ultra Light - .NET プログラミング』を参照してください。

関連する動作

オートコミット。

参照

- 「Ultra Light の式」 343 ページ
- 「Ultra Light DROP TABLE 文」 498 ページ
- 「CREATE TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「Ultra Light のデータ型」 328 ページ
- 「オートインクリメント・カラムの分割サイズの上書き」 138 ページ

例

次の文は、図書データベース用にテーブルを作成し、図書情報を保持します。

```
CREATE TABLE library_books (
  isbn CHAR(20)     PRIMARY KEY,
  copyright_date    DATE,
  title             CHAR(100),
  author            CHAR(50),
  location          CHAR(50),
```

```
    FOREIGN KEY location REFERENCES room
);
```

次の文では、図書データベース用にテーブルを作成して、貸し出し図書の情報を保持します。date_borrowed のデフォルト値は、エントリが作成される日に本が貸し出されることを示します。date_returned カラムは、本が返却されるまでは NULL です。

```
CREATE TABLE borrowed_book (
  loaner_name   CHAR(100) PRIMARY KEY,
  date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,
  date_returned DATE,
  book CHAR(20)
  FOREIGN KEY book REFERENCES library_books (isbn)
);
```

次の文は、販売データベース用にテーブルを作成し、注文と注文項目情報を保持します。

```
CREATE TABLE Orders (
  order_num INTEGER NOT NULL PRIMARY KEY,
  date_ordered DATE,
  name CHAR(80)
);
CREATE TABLE Order_item (
  order_num       INTEGER NOT NULL,
  item_num        SMALLINT NOT NULL,
  PRIMARY KEY (order_num, item_num),
  FOREIGN KEY (order_num)
  REFERENCES Orders (order_num)
);
```

Ultra Light DELETE 文

この文は、データベースのテーブルからローを削除するために使用します。

構文

DELETE [**FROM**] *table-name*[[**AS**] *correlation-name*]
[**WHERE** *search-condition*]

パラメータ

correlation-name　文の他の部分からテーブルを参照するときに使用する識別子。

WHERE 句　WHERE 句を指定すると、*search-condition* を満たすローだけが削除されます。「Ultra Light の探索条件」 349 ページを参照してください。

WHERE 句では、非確定の関数 (RAND など) や変数を使用できません。この句では、カラムも制限されません。カラムをサブクエリで使用する場合、別のテーブルを参照する必要がある場合があります。

備考

Ultra Light では、独自の方法でローのステータスがトレースされます。削除やローのステータスの意味を理解している必要があります。「Ultra Light でのローのステータス」 13 ページを参照してください。

参照

- 「Ultra Light START SYNCHRONIZATION DELETE 文」 510 ページ
- 「Ultra Light STOP SYNCHRONIZATION DELETE 文」 511 ページ

例

次の文は、Employees テーブルから従業員 105 を削除します。

```
DELETE
FROM Employees
WHERE EmployeeID = 105;
```

次の文は、FinancialData テーブルから、2000 年より前のデータをすべて削除します。

```
DELETE
FROM FinancialData
WHERE Year < 2000;
```

Ultra Light DROP INDEX 文

この文は、インデックスを削除するために使用します。

構文

DROP INDEX [*table-name.*]*index-name*

備考

テーブルのプライマリ・インデックスを削除することはできません。

DROP INDEX 文の処理中は、そのインデックスを参照する要求やクエリは処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は DROP INDEX を実行できません。

Ultra Light.NET を使用している場合、すべてのデータ・オブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。「Dispose メソッド」『Ultra Light - .NET プログラミング』を参照してください。

スキーマの変更が同時に開始された場合、文は解放されません。「DDL 文を使用したスキーマ変更」 10 ページを参照してください。

参照

- 「Ultra Light CREATE INDEX 文」 480 ページ
- 「Ultra Light のインデックスの操作」 84 ページ

例

次の文は、FinancialData テーブルで、架空のインデックス fin_codes_idx を削除します。

```
DROP INDEX FinancialData.fin_codes_idx;
```

Ultra Light DROP PUBLICATION 文

この文は、パブリケーションを削除するために使用します。

構文

DROP PUBLICATION *publication-name*, ...

参照

- 「Ultra Light での同期の設計」 139 ページ
- 「Ultra Light ALTER PUBLICATION 文」 471 ページ
- 「Ultra Light CREATE PUBLICATION 文」 482 ページ

例

次の文は、pub_contact パブリケーションを削除します。

```
DROP PUBLICATION pub_contact;
```

Ultra Light DROP SYNCHRONIZATION PROFILE 文

この文は、Ultra Light 同期プロファイルを削除するために使用します。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。

構文

DROP SYNCHRONIZATION PROFILE *sync-profile-name*

パラメータ

- **sync-profile-name**　同期プロファイルの名前。

備考

なし

関連する動作

なし

参照

- 「Ultra Light CREATE SYNCHRONIZATION PROFILE 文」 484 ページ
- 「Ultra Light ALTER SYNCHRONIZATION PROFILE 文」 472 ページ
- 「Ultra Light SYNCHRONIZE 文」 512 ページ

例

次の例は、Test1 という同期プロファイルを削除するための構文を示します。

```
DROP SYNCHRONIZATION PROFILE Test1;
```


Ultra Light DROP TABLE 文

この文は、テーブルとテーブル内のすべてのデータをデータベースから削除するために使用します。

構文

DROP TABLE *table-name*

備考

DROP TABLE 文は、データベースから指定したテーブルを削除します。テーブル内にあるすべてのデータ、およびすべてのインデックスとキーも削除されます。

DROP TABLE 文の処理中は、そのテーブルやテーブルのインデックスを参照する要求やクエリは処理されません。また、アクティブなクエリやコミットされていないトランザクションがある場合は DROP TABLE を実行できません。

Ultra Light.NET の場合、すべてのデータ・オブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。「Dispose メソッド」 『Ultra Light - .NET プログラミング』を参照してください。

スキーマの変更が同時に開始された場合、文は解放されません。「DDL 文を使用したスキーマ変更」 10 ページを参照してください。

参照

- 「Ultra Light ALTER TABLE 文」 474 ページ
- 「Ultra Light CREATE TABLE 文」 489 ページ

例

次の文は、データベースから架空のテーブル EmployeeBenefits を削除します。

```
DROP TABLE EmployeeBenefits;
```


Ultra Light FROM 句

この句は、SELECT 文に必要なテーブルまたはビューを指定するために使用します。

構文

FROM *table-expression*, ...

table-expression :
table-name [[**AS**] *correlation-name*]
| **(** *select-list* **)** [**AS**] *derived-table-name* **(** *column-name*, ... **)**
| **(** *table-expression* **)**
| *table-expression join-operator table-expression* [**ON** *search-condition*] ...

join-operator :
,
| **INNER JOIN**
| **CROSS JOIN**
| **LEFT OUTER JOIN**
| **JOIN**

パラメータ

table-name　ベース・テーブルまたはテンポラリ・テーブル。Ultra Light では、異なるユーザがテーブルを所有できません。ユーザ ID でテーブルを修飾した場合、ID は無視されます。

correlation-name　文の他の部分からテーブルを参照するときに使用する識別子。たとえば、次の文で、a は Contacts テーブルの相関名、b は Customers テーブルの相関名として定義されています。

```
SELECT *
FROM Contacts a, Customers b
WHERE a.CustomerID=b.ID;
```

derived-table-name　派生テーブルは、FROM 句でネストされている SELECT 文です。

派生テーブルの Select リストの項目は、派生テーブル名 (オプション) に続いてピリオド (.) とカラム名を指定して参照します。カラム名があいまいではない場合は、カラム名だけを使用できます。

SELECT 文内から派生テーブルを参照することはできません。「式のサブクエリ」 347 ページを参照してください。

join-operator　ジョインのタイプを指定します。カンマ (,) または CROSS JOIN を指定した場合は、ON サブ句を指定できません。JOIN を指定した場合は、ON サブ句を指定する必要があります。INNER JOIN と LEFT OUTER JOIN の場合は、ON サブ句はオプションです。

備考

FROM 句を指定しない場合、SELECT 文の式は定数式である必要があります。

派生テーブル
この説明はテーブルについてのものですが、特に注意書きがなければ派生テーブルにも適用します。

FROM 句は、指定した全テーブルのすべてのカラムで構成される結果セットを作成します。最初に、指定したテーブルのすべてのローの組み合わせが結果セットの中に入ります。次に、JOIN 条件か WHERE 条件、またはその両方の分だけ、通常は組み合わせの数が減ります。

ジョインのタイプを指定しないで、カンマ区切りリストとしてテーブルをリストする場合、デフォルトでは CROSS JOIN が使用されます。

INNER ジョインの場合、ON 句または WHERE 句を使用してジョインの結果を制限すると、同じ結果が返されます。OUTER ジョインの場合、これらは同じではありません。

注意
Ultra Light では、KEY JOIN と NATURAL JOIN の各ジョインはサポートされていません。

参照

- 「ジョイン：複数テーブルからのデータ検索」 『SQL Anywhere サーバ - SQL の使用法』
- 「Ultra Light DELETE 文」 494 ページ
- 「Ultra Light SELECT 文」 507 ページ
- 「Ultra Light UPDATE 文」 517 ページ

例

次は、有効な FROM 句です。

```
...
FROM Employees
...

...
FROM Customers
CROSS JOIN SalesOrders
CROSS JOIN SalesOrderItems
CROSS JOIN Products
...
```

次のクエリは、派生テーブルを使用して、SalesOrders テーブルに 4 件以上の注文がある Customers テーブルの顧客名を返します。

```
SELECT Surname, GivenName, number_of_orders
FROM Customers JOIN
    ( SELECT CustomerID, COUNT(*)
      FROM SalesOrders
      GROUP BY CustomerID )
    AS sales_order_counts( CustomerID, number_of_orders )
ON ( Customers.id = sales_order_counts.CustomerID )
WHERE number_of_orders > 3;
```


Ultra Light INSERT 文

この文は、テーブルにローを挿入するために使用します。

構文

INSERT [**INTO**]
table-name [**(** *column-name*, ... **)**]
{ **VALUES (** *expression, ...* **)** | *select-statement* }

備考

INSERT 文を使用すると、1 つのローを挿入するか、クエリの結果セットから複数のローを挿入できます。

カラムを指定した場合は、指定したカラムに 1 対 1 で値が挿入されます。カラム名のリストを指定しないと、テーブルでの表示順 (SELECT * を使用して取り出すときと同じ順序) に値がテーブル・カラムに挿入されます。ローは、任意の順序でテーブルに挿入されます。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字の状態のままで格納されます。

参照

- 「Ultra Light SELECT 文」 507 ページ

例

次の文は、データベースに Eastern Sales 部を追加します。

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 230, 'Eastern Sales' );
```

Ultra Light LOAD TABLE 文

この文は、外部ファイルからデータベース・テーブルの中へバルク・データをインポートするために使用します。またこの文は、SQL Anywhere の dbunload ユーティリティ (reload.sql ファイル) の出力処理もサポートします。

LOAD [**INTO**] **TABLE** [*owner.*]*table-name*
(*column-name*, ... **)**
FROM *stringfilename*
[*load-option* ...]

load-option :
CHECK CONSTRAINTS { **ON** | **OFF** }
| **COMPUTES** { **ON** | **OFF**}
| **DEFAULTS** { **ON** | **OFF** }
| **DELIMITED BY** *string*
| **ENCODING** *encoding*
| **ESCAPES** { **ON** }
| **FORMAT** { **ASCII** | **TEXT**}
| **ORDER** { **ON** | **OFF**}|
| **QUOTES** { **ON** | **OFF** }
| **SKIP** *integer*
| **STRIP** { **ON** | **OFF** | **BOTH** }
| **WITH CHECKPOINT** { **ON** | **OFF** }

comment-prefix : string

encoding : string

パラメータ

- **column-name**　この句は、データのロード先となる 1 つ以上のカラムを指定するために使用します。DEFAULTS が OFF に設定されている場合、カラム・リストにないカラムは NULL になります。DEFAULTS オプションが ON で、カラムにデフォルト値が入っている場合は、その値が使用されます。DEFAULTS オプションが OFF で、NULL 入力不可のカラムがカラム・リストから省かれている場合は、データベース・サーバは、空の文字列をカラムの型に変換しようとします。

  カラム・リストが指定されていると、ファイルに存在すると思われるカラムと、想定されるファイル内でのカラムの順序がカラム・リストにリストされます。カラム名を繰り返すことはできません。

- **FROM string-filename**　この句は、データのロード元となるファイルを指定するために使用します。*string-filename* は文字列としてデータベース・サーバに渡されます。したがって、文字列は他の SQL 文字列と同じデータベースのフォーマット要件に従います。特に、次の点に注意してください。

  ○ ディレクトリ・パスを示すには、円記号 (¥) を 2 つの円記号で表してください。したがって、ファイル c:¥temp¥input.dat から Employees テーブルにデータをロードする文は、次のようになります。

```
LOAD TABLE Employees
FROM 'c:¥¥temp¥¥input.dat' ...
```

- ○ パス名はデータベース・サーバを基準にした相対パスを指定します。クライアント・アプリケーションではありません。
- ○ UNC パス名を使用すると、データベース・サーバ以外のコンピュータ上のファイルからデータをロードできます。

● **load-option 句** さまざまなロード・オプションを使用して、データのロード方法を制御できます。次のリストは、サポートされているロード・オプションを示します。

- ○ **CHECK CONSTRAINTS 句** この句は、ロード中に制約をチェックするかどうかを制御します。CHECK CONSTRAINTS はデフォルトで ON に設定されますが、アンロード・ユーティリティ (ulunload) は CHECK CONSTRAINTS を OFF に設定して LOAD TABLE 文を書き出します。CHECK CONSTRAINTS を OFF に設定すると、データベースの再構築などの場合に役立つ検査制約が無効になります。
- ○ **COMPUTES 句** このオプションは処理されますが、Ultra Light では無視されます。
- ○ **DEFAULTS 句** デフォルトでは、DEFAULTS は OFF に設定されています。DEFAULTS が OFF の場合は、カラム・リストにないカラムすべてに NULL が割り当てられます。DEFAULTS が OFF に設定されており、NULL 入力不可のカラムがカラム・リストから省かれている場合は、データベース・サーバは、空の文字列をカラムの型に変換しようとします。DEFAULTS が ON に設定されており、カラムにデフォルト値が入っている場合は、その値が使用されます。
- ○ **DELIMITED BY 句** この句は、カラム・デリミタ文字を指定するために使用します。デフォルトのカラム・デリミタ文字列はカンマです。ただし、最長で 255 バイトの文字列を指定できます。たとえば、... DELIMITED BY '###' ... などです。他の SQL 文字列と同じフォーマット要件が適用されます。タブで区切った値を指定する場合、タブ文字を表す 16 進のエスケープ・シーケンス (9) を使用して、... DELIMITED BY '¥x09' ... のように指定します。
- ○ **ENCODING 句** この句は、データベースにロードするデータに使用する文字エンコードを指定します。
- ○ **ESCAPES 句** ESCAPES は常に ON であるため、データベース・サーバによって円記号に続く文字が認識され、特殊文字として解釈されます。改行文字は ¥n との組み合わせとしてインクルードされ、他の文字はタブ文字の ¥x09 のような 16 進の ASCII のコードとしてデータにインクルードされます。2 つの円記号 (¥) は 1 つの円記号として解釈されます。円記号 (¥) の後に n、x、X、¥ 以外の文字がある場合、それらは別々の文字と解釈されます。たとえば、¥q であれば、円記号と q が挿入されます。
- ○ **FORMAT 句** この句は、データのロード元となるデータ・ソースのフォーマットを指定します。TEXT の場合、(ENCODING オプションで定義したとおり) 入力行は文字であり、1 行あたり 1 つのローで構成され、カラム・デリミタ文字列によって値が区切られているものとみなされます。ASCII もサポートされています。
- ○ **QUOTES 句** この句は、文字列を引用符で囲むかどうかを指定します。Ultra Light は ON のみをサポートしているため、LOAD TABLE 文は引用符文字で囲まれた文字列を探します。引用符文字はアポストロフィ (一重引用符) です。文字列の中で最初に出てくるこのような文字は、その文字列の引用符文字として処理されます。文字列の終わりには先頭にあるものと同じ引用符が必要です。

カラム・デリミタ文字列をカラム値の中に入れることができます。また、引用符文字は値の一部とはみなされません。したがって、次の行はアドレスにカンマがあるかどうかは関係ありません。また、アドレスを囲む引用符は、データベースへは挿入されません。

```
'123 High Street, Anytown',(715)398-2354
```

値の中に引用符文字をインクルードするには、2 つの引用符を使用する必要があります。次の行は、第 3 カラムの中へ一重引用符文字である値を入れます。

```
'123 High Street, Anytown','(715)398-2354',""
```

- **SKIP 句**　この句は、ファイルの最初の数行を無視するかどうかを指定するために使用します。*integer* 引数では、スキップする行数を指定します。たとえば、この句を使用して、カラム見出しを含む行をスキップできます。
- **STRIP 句**　この句は処理されますが、無視されます。この句は、引用符がない値に、値を挿入する前に削除された先行ブランクまたは後続ブランクがあるかどうかを指定します。STRIP オプションは次のオプションを受け入れます。
  - **STRIP ON**　先行ブランクを削除します。
  - **STRIP OFF**　先行ブランクまたは後続ブランクを削除しません。
  - **STRIP BOTH**　先行ブランクと後続ブランクの両方を削除します。
- **WITH CHECKPOINT 句**　この句は、チェックポイントを実行するかどうかを指定するために使用します。デフォルト設定は OFF です。この句を ON に設定すると、文が正常に完了した後、チェックポイントが発行されます。

備考

LOAD TABLE を使うと、ファイルからデータベース・テーブルの中へ効率よく大量の挿入を行えます。この文は主に、SQL Anywhere の dbunload ユーティリティ (reload.sql ファイル) の出力処理をサポートする手段です。

LOAD TABLE は、Windows と Linux でのみサポートされており、Palm OS や Windows Mobile ではサポートされていません。

FORMAT TEXT の場合、値を指定しないことが、NULL 値を指定することになります。たとえば、1,,'Fred', という値を含むファイルに 3 つの値が予想される場合、挿入される値は 1、NULL、Fred です。ファイルに 1,2, が含まれる場合、値 1、2、NULL が挿入されます。空白のみで構成される値は、NULL 値と扱われます。ファイルに 1, ,'Fred', が含まれる場合、値 1、NULL、Fred が挿入されます。他の値はすべて NULL 以外と扱われます。たとえば、'' (一重引用符が 2 つ) は空の文字列です。'NULL' は 4 文字の文字列です。

LOAD TABLE でロードしていないカラムが NULL 値を許容しておらず、ファイル値が NULL の場合、数値カラムには値 0 (ゼロ)、文字カラムには空の文字列 ('') が指定されます。LOAD TABLE でロードされたカラムが NULL 値を許容し、ファイル値が NULL の場合、カラム値は (すべての型で) NULL になります。

テーブルに a、b、c の各列が含まれ、入力データに a、b、c が含まれているときに、LOAD 文がデータのロード先として a と b のみを指定した場合は、カラム c に次の値が挿入されます。

- DEFAULTS ON が指定されると、カラム c にデフォルト値が入っている場合は、デフォルト値が使用されます。
- カラム c にデフォルト値がなく、NULL が許容されている場合は、NULL が使用されます。
- カラム c にデフォルト値がなく、NULL が許容されていない場合は、カラムのデータ型に応じて、ゼロ (0) または空の文字列 ('') が使用されるかエラーが返されます。

関連する動作

オートコミット。

参照

- 「Ultra Light INSERT 文」 501 ページ
- 「Ultra Light データベースのアンロード・ユーティリティ (ulunload)」 298 ページ
- 「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』

標準と互換性

- **SQL/2003** ベンダ拡張。

例

次に、LOAD TABLE の例を示します。まずテーブルを作成し、次に *input.txt* というファイルを使用してテーブルにデータをロードします。

```
CREATE TABLE t( a CHAR(100) primary key, let_me_default INT DEFAULT 1, c CHAR(100) );
```

次に、ファイル *input.txt* の内容を示します。

```
'this_is_for_column_c', 'this_is_for_column_a', ignore_me
```

次の LOAD 文は、*input.txt* というファイルをロードします。

```
LOAD TABLE T ( c, a ) FROM 'input.txt' FORMAT TEXT DEFAULTS ON;
```

コマンド SELECT * FROM t は、次の結果セットを返します。

a	let_me_default	c
this_is_for_column_a	1	this_is_for_column_c

Ultra Light ROLLBACK 文

この文は、トランザクションを終了し、COMMIT 文または ROLLBACK 文が最後に実行されて以降の変更を元に戻すときに使用します。

構文

ROLLBACK [WORK]

備考

Ultra Light SQL を使用するとトランザクションが作成されます。トランザクションは、最後の ROLLBACK または COMMIT 後のすべての変更内容 (INSERT、UPDATE、DELETE) から構成されます。ROLLBACK 文は、現在のトランザクションを終了し、前回の COMMIT または ROLLBACK 以降にデータベースに対して行われたすべての変更を元に戻します。

参照

- 「Ultra Light COMMIT 文」 479 ページ

例

次の文は、前回コミット時の状態にデータベースをロールバックします。

```
ROLLBACK;
```


Ultra Light SELECT 文

この文は、データベースから情報を取り出すために使用します。

構文

SELECT [**DISTINCT**] [*row-limitation*]
select-list
[**FROM** *table-expression*, ...]
[**WHERE** *search-condition*]
[**GROUP BY** *group-by-expression*, ...]
[**ORDER BY** *order-by-expression*, ...]
[**FOR** { **UPDATE** | **READ ONLY** }]
[**OPTION (FORCE ORDER)**]

row-limitation :
FIRST
| **TOP** *n* [**START AT** *m*]

select-list :
expression [[**AS**] *alias-name*], ...

order-by-expression :
{ *integer* | *expression* } [**ASC** | **DESC**]

パラメータ

DISTINCT 句　DISTINCT を指定して、重複するローを結果から削除します。DISTINCT を指定しない場合は、SELECT 文の句を満たすすべてのローを返します。ここには重複するローも含まれます。多くの場合、DISTINCT を指定すると、文の実行時間が非常に長くなります。したがって、DISTINCT を使用するのは、必要な場合だけにしてください。

row-limitation 句　ロー制限を使用して、結果のサブセットを返します。たとえば、結果セットの最初のローを取得するには、FIRST を指定します。結果の最初の *n* 個のローを返すには、TOP*n* を使用します。TOP*n* 個のローを取得するときの開始ローの位置を制御するには、START AT*m* を指定します。これらの句が意味のある結果を返すようにローの順序を指定するには、SELECT 文に ORDER BY 句を指定します。

select-list　式のリストであり、データベースから何を取り出すかを指定します。通常の場合、select リスト内の式はカラム名です。ただし、関数のように、他のタイプの式にすることもできます。アスタリスク (*) を使用すると、FROM 句にリストされた全テーブルのすべてのカラムを選択できます。オプションとして、*select-list* 内のそれぞれの式に対して、エイリアスを定義できます。エイリアスを使用すると、クエリ内の別の位置 (WHERE 句や ORDER BY 句など) から、*select-list* の式を参照できるようになります。

FROM 句　*table-expression* の中で指定されるテーブルとビューからローを取り出します。「Ultra Light FROM 句」 499 ページを参照してください。

WHERE 句　WHERE 句を指定すると、*search-condition* を満たすローだけが選択されます。「Ultra Light の探索条件」 349 ページを参照してください。

GROUP BY 句　GROUP BY 句を使用したクエリの結果には、GROUP BY 式の中の個別の値の各セットに対し 1 つのローが入ります。テーブル・リストのローの各グループに対する結果には

ローが 1 つずつ含まれるため、結果ローはグループとして頻繁に参照されます。これらのグループ内のローには、集合関数を適用できます。NULL があった場合、ユニークな値と見なされます。

ORDER BY 句　この句で指定した式に従って、クエリの結果がソートされます。ORDER BY 句の各式は、昇順 (ASC) または降順 (DESC) (デフォルト) でソートすることができます。式が整数 *n* である場合、クエリの結果は select リストの *n* 番目の式でソートされます。

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。ORDER BY 句がない場合は、Ultra Light が最も効率のよい順序でローを返します。

FOR 句　この句には、クエリの動作を制御する 2 つの形式があります。

- **FOR READ ONLY**　この句は、クエリが更新に使用されないことを示します。クエリが更新に使用されないことがわかっている場合は、Ultra Light のパフォーマンスが向上することがあるため、この句はできるだけ指定してください。たとえば、読み込み専用アクセスが必要であることがわかると、Ultra Light はダイレクト・テーブル・スキャンを実行できます。FOR READ ONLY はデフォルトの動作です。「ダイレクト・ページ・スキャンの使用」 123 ページを参照してください。
- **FOR UPDATE**　この句によってクエリを更新に使用できるようになります。この句は、明示的に指定する必要があります。明示的に指定しないと、更新は許可されません (FOR READ ONLY がデフォルトの動作)。

OPTION (FORCE ORDER) 句　この句は一般的には使用しないことをおすすめします。この句は、Ultra Light で選択されたテーブル・アクセス順序を上書きし、クエリに出現する順序でテーブルにアクセスするよう Ultra Light に要求します。この句は、クエリの順序が Ultra Light の順序よりも確実に効率的である場合にのみ使用してください。

Ultra Light では、実行プランを使用してクエリを最適化することもできます。「Ultra Light の実行プラン」 360 ページを参照してください。

備考

クエリは必ず閉じてください。そうしないと、メモリが解放されず、存在し続けるテンポラリ・テーブルの数が不必要に増加することになります。

参照

- 「Ultra Light のパフォーマンスと最適化」 113 ページ
- 「SELECT 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「データのクエリ」 『SQL Anywhere サーバ - SQL の使用法』

例

次の文は、Employees テーブルから従業員の数を選択します。

```
SELECT COUNT(*)
FROM Employees;
```

次の文は、Employees テーブルで 40 番目から 49 番目までの 10 個のローを選択します。

```
SELECT TOP 10 START AT 40 * FROM Employees;
```


Ultra Light SET OPTION 文

この文は、データベース・オプションの値を変更するために使用します。

構文

SET OPTION *option-name*=*option-value*

option-name: identifier

option-value: string, identifier, or number

備考

この文ではデータベース・オプションの設定しかできません。データベースが作成された後にプロパティを変更することはできません。**ml_remote_id** は、これらのルールの例外となります (以下を参照)。

オプションが永続的かどうかの指定はできません。オプションが一時的か永続的かは、Ultra Light でのオプションの実装方法で決まります。永続的なオプションは sysuldata テーブルに格納されます。一時的なオプションが使用されるのは、データベースの実行が停止するまでです。

設定解除が可能なデータベース・オプションは **ml_remote_id** だけです。次に例を示します。

```
SET OPTION ml_remote_id=;
```

この結果、ID が NULL に設定されます。

参照

- 「sysuldata システム・テーブル」 314 ページ
- 「Ultra Light データベース・オプション」 227 ページ
- 「DB_PROPERTY 関数 [システム]」 398 ページ
- 「Ultra Light ml_remote_id オプション」 232 ページ

例

次の文を実行すると、global_database_id オプションが 100 に設定されます。

```
SET OPTION global_database_id=100;
```

Ultra Light START SYNCHRONIZATION DELETE 文

この文は、Mobile Link 同期で削除されたローのロギングを再起動するために使用します。

構文

START SYNCHRONIZATION DELETE

備考

Ultra Light データベースでは、同期が必要なローへの変更のログを自動的に取ります。これらの変更は次の同期時に統合データベースにアップロードされます。この文を使用すると、その前に STOP SYNCHRONIZATION DELETE 文で停止されていた、削除されたローのロギングを再起動できます。

STOP SYNCHRONIZATION DELETE 文を実行すると、その接続に対してそれ以降に実施された削除操作は同期されません。この効果は、START SYNCHRONIZATION DELETE 文が実行されるまで継続します。

アプリケーションでデータを同期しない場合は、START SYNCHRONIZATION DELETE を使用しないでください。

Ultra Light では、独自の方法でローのステータスがトレースされます。削除やローのステータスの意味を理解している必要があります。「Ultra Light でのローのステータス」 13 ページを参照してください。

参照

- 「Ultra Light STOP SYNCHRONIZATION DELETE 文」 511 ページ

例

次の一連の SQL 文は、START SYNCHRONIZATION DELETE と STOP SYNCHRONIZATION DELETE の使用方法を示します。

```
STOP SYNCHRONIZATION DELETE;
DELETE FROM PROPOSAL
  WHERE last_modified < months( CURRENT TIMESTAMP, -1 );
START SYNCHRONIZATION DELETE;
COMMIT;
```


Ultra Light STOP SYNCHRONIZATION DELETE 文

この文は、Mobile Link 同期で削除されたローのロギングを停止するために使用します。

構文

STOP SYNCHRONIZATION DELETE

備考

Ultra Light データベースでは、同期が必要なローへの変更のログを自動的に取ります。これらの変更は次の同期時に統合データベースにアップロードされます。この文を使用すると、その前に START SYNCHRONIZATION DELETE 文で起動されていた、削除されたローのロギングを停止できます。このコマンドは、リモート・データベースに対して訂正を行うには便利ですが、Mobile Link 同期を事実上無効にしてしまうので、使用の際には注意してください。削除のロギングを停止するのは、一時的のみにしてください。

STOP SYNCHRONIZATION DELETE 文を実行すると、その接続に対してそれ以降に実施された削除操作は同期されません。この効果は、START SYNCHRONIZATION DELETE 文が実行されるまで継続します。

アプリケーションでデータを同期しない場合は、STOP SYNCHRONIZATION DELETE を使用しないでください。

Ultra Light では、独自の方法でローのステータスがトレースされます。削除やローのステータスの意味を理解している必要があります。「Ultra Light でのローのステータス」 13 ページを参照してください。

参照

- 「Ultra Light START SYNCHRONIZATION DELETE 文」 510 ページ

例

次の一連の SQL 文は、START SYNCHRONIZATION DELETE と STOP SYNCHRONIZATION DELETE の使用方法を示します。

```
STOP SYNCHRONIZATION DELETE;
DELETE FROM PROPOSAL
WHERE last_modified < months( CURRENT TIMESTAMP, -1 );
START SYNCHRONIZATION DELETE;
COMMIT;
```


Ultra Light SYNCHRONIZE 文

この文は、Mobile Link サーバを介して Ultra Light データベースを同期するために使用します。同期は、同期プロファイルのパラメータに応じて設定します。または、文自体の中でパラメータを指定できます。

構文

SYNCHRONIZE {
PROFILE *sync-profile-name* [**MERGE** *sync-option* [**;**...]]
| **USING** *sync-option* [**;**...]
}

sync-option :
sync-option-name **=** *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

パラメータ

- **sync-profile-name** 同期プロファイルの名前。
- **MERGE 句** この句は、同期プロファイルに指定されているオプションを追加または上書きするために使用します。
- **USING 句** この句は、同期プロファイルを参照しないで同期オプションを指定するために使用します。
- **sync-option** 1 つ以上の同期オプションの値ペアがセミコロンで区切られた文字列です。たとえば、'option1=value1;option2=value2' のように記述します。
- **sync-option-name** 同期オプションの名前
- **sync-option-value** 同期オプションの値。

備考

Ultra Light でサポートされる同期プロファイル・オプションのリストについては、「Ultra Light CREATE SYNCHRONIZATION PROFILE 文」 484 ページを参照してください。

同期プロファイルで sync オプションを既存のオプションとマージする方法については、「Ultra Light ALTER SYNCHRONIZATION PROFILE 文」 472 ページを参照してください。

sync オプションをマージできると、開発者は、(MobiLinkPwd などのように) 一部のオプションをデータベースに格納する作業を省略できます。

同期コールバック関数が定義され、Ultra Light に登録されている場合は、SYNCHRONIZE 文を実行するたびに、その同期の進行状況情報がコールバック関数に渡されます。コールバック関数が登録されていない場合、進行状況情報は渡されません。

関連する動作

なし

参照

- 「Ultra Light ALTER SYNCHRONIZATION PROFILE 文」 472 ページ
- 「Ultra Light DROP SYNCHRONIZATION PROFILE 文」 497 ページ
- 「ULRegisterSynchronizationCallback」 『Ultra Light - C/C++ プログラミング』

例

次の例は、Test1 という同期プロファイルを削除するための構文を示します。ここでは、MobiLinkPwd がプロファイルの一部として格納されていません。

```
SYNCHRONIZE PROFILE Test1 MERGE "MobiLinkPwd=sql'
```

次の例は、Test1 という同期プロファイルに、publication オプションと uploadonly オプションを追加するための構文を示します。

```
SYNCHRONIZE PROFILE Test1
  MERGE'publication=p2;uploadonly=on';
```

次の例は、USING の使用方法を示します。

```
SYNCHRONIZE USING
"MobiLinkUid=joe;MobiLinkPwd=sql;ScriptVersion=1;Stream=TCPIP{host=localhost}'
```

次の例は、publication オプションと uploadonly オプションを同期するための構文を示します。

```
SYNCHRONIZE
  USING 'publication=p2;uploadonly=on';
```

Ultra Light TRUNCATE TABLE 文

この文は、テーブルは削除せずにテーブルからすべてのローを削除するために使用します。

構文

TRUNCATE TABLE *table-name*

備考

TRUNCATE TABLE 文によってテーブル内のローがすべて削除され、この後の同期時に Mobile Link サーバにこの削除について通知されません。これは、次の文と同義です。

```
STOP SYNCHRONIZATION DELETE;
DELETE FROM TABLE;
START SYNCHRONIZATION DELETE;
```

注意
この文は、同期またはレプリケーション対象のデータベースに対して慎重に使用する必要があります。Mobile Link サーバに通知されないので、この削除によって整合性が失われ、その結果、同期またはレプリケーションに失敗する可能性があります。

TRUNCATE TABLE 文の実行後、テーブル構造体、すべてのインデックス、制約およびカラム定義は存在し続けます。削除されるのはデータのみです。

テーブルを対象とした文が、別の要求やクエリですでに参照されている場合、TRUNCATE TABLE は実行できません。同様に、テーブルの変更中は、そのテーブルを参照する要求は処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は TRUNCATE TABLE を実行できません。

Ultra Light.NET を使用している場合、すべてのデータ・オブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。「Dispose メソッド」『Ultra Light - .NET プログラミング』を参照してください。

スキーマの変更　スキーマの変更が同時に開始された場合、文は解放されません。「DDL 文を使用したスキーマ変更」 10 ページを参照してください。

関連する動作

テーブルに DEFAULT AUTOINCREMENT または DEFAULT GLOBAL AUTOINCREMENT と定義されたカラムがある場合、TRUNCATE TABLE はそのカラムの次に使用可能な値をリセットします。

TRUNCATE TABLE でローが削除済みとマーク付けされると、この処理を実行したユーザは、ROLLBACK 文を発行しないかぎり、ローにアクセスできなくなります。ただし、他の接続からはアクセスできます。COMMIT を使用すると削除が確定し、すべての接続からデータにアクセスできなくなります。

トランケート対象のテーブルを同期すると、テーブルに適用されているすべての INSERT 文が、TRUNCATE TABLE 文より優先されます。

参照

- 「Ultra Light DELETE 文」 494 ページ
- 「Ultra Light START SYNCHRONIZATION DELETE 文」 510 ページ
- 「Ultra Light STOP SYNCHRONIZATION DELETE 文」 511 ページ

例

次の文は Departments テーブルからすべてのローを削除します。

```
TRUNCATE TABLE Departments;
```

この例を実行する場合は、ROLLBACK 文を実行して、変更内容を元に戻すようにしてください。「Ultra Light ROLLBACK 文」 506 ページを参照してください。

Ultra Light UNION 文

この文は、2 つ以上の SELECT 文の結果を結合するために使用します。

構文

select-statement-without-ordering
[**UNION** [**ALL** | **DISTINCT**] *select-statement-without-ordering*]...
[**ORDER BY** [*number* [**ASC** | **DESC**] , ...]

備考

いくつかの SELECT 文の結果を、UNION を使ってより大きな結果へと結合することができます。各 SELECT 文には、それぞれの select リストに同じ数の式を指定します。各文には ORDER BY 句を含めることはできません。

UNION ALL の結果は、統合された SELECT 文の結果を結合したものです。ローが重複しないようにして結果を取得するには、UNION または UNION DISTINCT を指定します。ただし、重複するローを削除することにより、文の合計実行時間が増えることに注意してください。ローの重複を許容するには、UNION ALL を指定します。

データ型の異なる対応する式を結合しようとすると、Ultra Light は、結合された値を表すデータ型を検索しようとします。この検索ができない場合、統合操作は失敗し、エラーが返されます (例 :「値 'Surname' をデータ型数値に変換できません。」)。

結果に表示されるカラム名は、最初の SELECT 文に対して表示されるカラム名 (またはエイリアス) です。

ORDER BY 句では、整数を使用して順序が確立されます。ここで整数は結果をソートするクエリ式を示します。

参照

- 「Ultra Light SELECT 文」 507 ページ

例

次の例は、結合された Employees テーブルと Customers テーブルで見つかったそれぞれの姓すべてをリストします。

```
SELECT Surname FROM Employees
UNION
SELECT Surname FROM Customers;
```


Ultra Light UPDATE 文

この文は、テーブルのローを変更するために使用します。

構文

UPDATE *table-name*[[**AS**] *correlation-name*]
SET *column-name* **=** *expression*, ...
[**WHERE** *search-condition*]

パラメータ

table-name　*table-name* は、更新するテーブルの名前を指定します。使用できるのは単一テーブルのみです。

correlation-name　文の他の部分からテーブルを参照するときに使用する識別子。

SET 句　それぞれ指定したカラムを、等号の右側の式の値に設定します。式には制限がありません。式が *column-name* である場合は、古い値が使われます。

SET 句で指定されたカラムの値のみ変更されます。特に、UPDATE を使用して、カラムの値をデフォルトに設定することはできません。

WHERE 句　WHERE 句を指定すると、*search-condition* を満たすローだけが更新されます。「Ultra Light の探索条件」 349 ページを参照してください。

備考

UPDATE 文は、テーブル内の値を修正します。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字の状態のままで格納されます。

参照

- 「Ultra Light INSERT 文」 501 ページ
- 「Ultra Light DELETE 文」 494 ページ
- 「Ultra Light の探索条件」 349 ページ

例

次の文は、従業員 Philip Chin (従業員 129) を販売部からマーケティング部 (部門 400) に移動します。

```
UPDATE Employees
SET DepartmentID = 400
WHERE EmployeeID = 129;
```

correlation-name を使用した例を次に示します。

```
UPDATE Employee E
SET salary = salary * 1.05
WHERE EXISTS( SELECT 1 FROM Sales S HAVING E.Sales > Avg( S.sales)
GROUP BY S.dept_no )
```


Ultra Light のトラブルシューティング

目次

Ultra Light エンジンを起動できない

現象

START 接続パラメータを使用し、次の定義で Ultra Light エンジンを起動しましたが、クライアントが SQLE_UNABLE_TO_CONNECT_OR_START を返します。

```
START="¥Program Files¥uleng11.exe"
```

説明

引用符の位置が正しくありません。

推奨事項

このパラメータが機能するには、最初の引用符の前に ¥ 記号がある必要があります。たとえば、次のようにしてこのパス内のスペースを明確に記述できます。

```
START=¥ :Program Files¥uleng11.exe"
```

または

```
START='"¥Program Files¥uleng11.exe"'
```


アップグレード後にデータベースに接続できない

現象

Ultra Light をアップグレードしました。管理ツールを使用して空の Ultra Light データベースを作成できることがわかりました。しかし、Sybase Central を使用して空のデータベースまたはその他の Ultra Light データベース (*CustDB.udb* を含む) に接続しようとすると、エラーが発生します。ただし、SQL Anywhere データベースとの接続は問題なく動作します。

説明

SQL Anywhere のすべてのアプリケーションとプロセスを閉じていませんでした。このため、Ultra Light のプラグインが正しくインストールされていません。

推奨事項

SQL Anywhere を削除して再インストールします。

1. Sybase Central、Interactive SQL、実行中のすべてのデータベース・エンジンを閉じます。
2. 次のコマンドを実行します。

   ```
   dbisql -terminate
   scjview -terminate
   ```

3. Windows の **[タスク マネージャ]** を開き、*scjview.exe* プロセスと *dbisql.exe* プロセスを閉じます。
4. 最新バージョンの Ultra Light を再インストールします。

参照

- 「Ultra Light のアップグレード」 『SQL Anywhere 11 - 変更点とアップグレード』

データベースの破損

現象

次の場合は、データベースが破損している可能性があります。

- 次のエラーが生成される。
  - SQLE_DEVICE_ERROR
  - SQLE_DATABASE_ERROR (他の問題による現象の可能性もあり)
  - SQLE_MEMORY_ERROR (他の問題による現象の可能性もあり)
- クラッシュするか、無効なクエリ結果が返される。

説明

より一般的な破損の原因は 2 つあります。

- 頻繁の高い原因としては、ファイルの格納時にデバイスに問題が発生したため、ファイルの内容が誤って変更されることを挙げることができます。通常、この問題が発生すると、データベースはすぐに機能を停止します。
- 頻度の低い原因としては、Ultra Light コードのエラーによってインデックスの正常な管理に失敗することを挙げることができます。クエリの結果に対する変更を予期することが難しいため、これらの問題は長期間発見されない可能性があります。

推奨事項

チェックサムは、オフラインの破損を検出するために使用します。チェックサムによって、重要なページの破損が原因で他のデータが破損するのを防ぐことができます。チェックサムが一致しなかった場合、データベースでページがロードされるときに、Ultra Light によってすぐにデータベースが停止され、致命的なエラーがレポートされます。このエラーは訂正できません。代わりに、次の手順を実行してください。

1. iAnywhere にエラーをレポートします。これは、破損の原因となる一連のイベントがわかっている場合、およびエラーに再現性がある場合に役立ちます。
2. データが必要な場合は、データベースの内容をファイルにアンロードします。
3. 新しいデータベースを作成します。
4. アンロードしたデータを同期またはロードすることによって、データを再移植します。

参照

- 「Ultra Light checksum_level 作成パラメータ」 191 ページ
- 「ULSQLCode 列挙体」 『Ultra Light - .NET プログラミング』

データベースのサイズが安定しない

現象

アプリケーションが複数のクライアント・デバイス間のラージ・バイナリ・オブジェクトを数多く収集して、この情報を統合データベースに同期し、その後、各クライアント・デバイスから同期されたデータが削除されます。ただし、データベースからデータが削除されても、データベースのサイズは大きいままです。このことは問題となります。デバイスのリソースは限られているため、ファイル・サイズは慎重に管理する必要があります。

説明

データベースのサイズが大きくなるのは、データベース内のデータ量が増える場合のみです。ただし、いったん拡大すると、データベース・ファイルはそのサイズを維持し、自動的に縮小しません。ファイルの空き領域は、内部的に管理されます。

推奨事項

同期しないテーブルに STOP SYNCHRONIZATION DELETE 文または TRUNCATE 文を使用しないようにします。代わりに、同期しないテーブルには FROM *table-name* 句を含む DELETE 文を使用します。

同期後のデータベースを再作成します。

1. デバイスに配備する Ultra Light データベースを作成します。
2. クライアント・デバイスで必要とされるスキーマを定義する DDL 文の SQL スクリプトを作成します。「Ultra Light スキーマのアップグレードの配備」 70 ページを参照してください。
3. データを同期します。
4. データベースを削除します。
5. 新しい空のデータベースを作成し、ALTER DATABASE SCHEMA FROM FILE 文で標準のデータベース・スキーマを使用します。

参照

- 「Ultra Light STOP SYNCHRONIZATION DELETE 文」 511 ページ
- 「Ultra Light TRUNCATE TABLE 文」 514 ページ
- 「Ultra Light DELETE 文」 494 ページ
- 「Ultra Light ALTER DATABASE SCHEMA FROM FILE 文」 470 ページ

新しいデータベースへの ASCII データのインポート

現象

新しい Ultra Light データベースを作成しましたが、インポートできない *.csv* ASCII データ・ファイルがあります。

説明

Ultra Light の管理ツールでは、*.csv* フォーマットはサポートされていません。

推奨事項

次のいずれかの方法を試すことができます。

- Interactive SQL (dbisql) を使用してデータをインポートします。Ultra Light データベースに接続し、**[データ] - [インポート]** を選択します。または、Ultra Light データベースに接続し、INPUT 文を実行します (この文は、Ultra Light の PreparedStatement オブジェクトでは使用できません)。

  注意
  Ultra Light にはプライマリ・キーが必要です。Interactive SQL ではユーザのテーブルを作成できますが、そのプライマリ・キーは自動的に作成されません。この目的のために作成した空の Ultra Light データベースに必ず接続してください。

- バッチ・プロセスの一部としてこの機能を組み込む場合は、独自のコードを作成してください。

参照

- 「INPUT 文 [Interactive SQL]」 『SQL Anywhere サーバ - SQL リファレンス』
- 「Ultra Light 用 Interactive SQL ユーティリティ (dbisql)」 263 ページ

以前のバージョンのユーティリティが動作する

現象

Ultra Light 11 のインストールを完了しました。しかし、Ultra Light のユーティリティを実行しようとすると、以前のバージョンが起動します。

説明

Ultra Light 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、管理ツールを使用するとき、システムのパスに注意してください。インストールでは、最も新しくインストールされたバージョンの実行プログラムのディレクトリがシステム・パスの最後に追加されるため、ソフトウェアの最新バージョンをインストールしたにもかかわらず、以前にインストールしたバージョンが実行されている場合があります。

推奨事項

この問題にはさまざまな対処方法があります。「ユーティリティの使用」 『SQL Anywhere 11 - 変更点とアップグレード』を参照してください。

結果セットに予測できない変更が生じる

現象

クエリを実行すると、実行するたびに結果セットが変わります。

説明

得られる結果セットを注意深く確認します。セットの結果は本当に違いますか。または、実行のたびに最も効率的な順序で結果が返されているだけではありませんか。ローに最後にアクセスした日時やその他の要因によって、クエリを実行するたびに選択される順序が変わることがあります。

推奨事項

予測可能な順序または一貫性のある順序で結果セットが返されるようにする場合は、SELECT 文に ORDER BY 句が含まれていることを確認します。結果セットで返される結果が依然として不適切な場合は、データベースが破損している可能性があります。

参照

- 「Ultra Light SELECT 文」 507 ページ
- 「データベースの破損」 522 ページ

Ultra Light エンジン・クライアントでエラー -764 が発生する

適用対象

Windows Mobile

現象

Windows Mobile デバイスで Ultra Light エンジンを実行しようとして、クライアントから -764 エラーが返されます。

説明

-764 のエラーは、エンジンが見つからないため、起動できなかったことを意味します。

推奨事項

次のいずれかのアクションを検討します。

- 推奨される配備ロケーション *¥Windows* ディレクトリにエンジンを再配備することを検討します。Ultra Light は、自動的にこのロケーションでエンジンのファイルを検索します。
- 別のロケーションにエンジンをインストールした場合は、接続コードに START 接続パラメータを使用していることを確認します。
- START 接続パラメータを使用しており、エンジンへのパスが正しい場合は、パス名の特殊文字に正しいエスケープ・シーケンスを使用していることを確認します。

  たとえば、次のコードは変更が必要となる場合があります。

  ```
  ULConnection conn = new ULConnection(@"dbf=¥Program Files¥HelloEngine¥HelloEngine.udb;
  START=¥Windows¥uleng11.exe")
  ```

  場合によっては、次のようなコードに変更します。

  ```
  ULConnection conn = new ULConnection(@"dbf=¥¥¥"Program Files ¥"¥¥HelloEngine¥
  ¥HelloEngine.udb;
  START=¥¥Windows¥¥uleng11.exe");
  ```

参照

- 「Ultra Light エンジンを持つ複数の Ultra Light アプリケーションの配備」 59 ページ
- 「Ultra Light START 接続パラメータ」 258 ページ

用語解説

用語解説

Adaptive Server Anywhere (ASA)

SQL Anywhere Studio のリレーショナル・データベース・サーバ・コンポーネントであり、主に、モバイル環境と埋め込み環境、または小規模および中規模のビジネス用のサーバとして使用されます。バージョン 10.0.0 で、Adaptive Server Anywhere は SQL Anywhere サーバに、SQL Anywhere Studio は SQL Anywhere にそれぞれ名前が変更されました。

参照：「SQL Anywhere」 536 ページ。

Carrier

Mobile Link システム・テーブルまたは Notifier プロパティ・ファイルに保存される Mobile Link オブジェクトで、システム起動同期で使用される通信業者に関する情報が含まれます。

参照：「サーバ起動同期」 541 ページ。

DB 領域

データ用の領域をさらに作成する追加のデータベース・ファイルです。1 つのデータベースは 13 個までのファイルに保管されます (初期ファイル 1 つと 12 の DB 領域)。各テーブルは、そのインデックスとともに、単一のデータベース・ファイルに含まれている必要があります。CREATE DBSPACE という SQL コマンドで、新しいファイルをデータベースに追加できます。

参照：「データベース・ファイル」 545 ページ。

DBA 権限

ユーザに、データベース内の管理作業を許可するレベルのパーミッションです。DBA ユーザにはデフォルトで DBA 権限が与えられています。

参照：「データベース管理者 (DBA)」 545 ページ。

EBF

Express Bug Fix の略です。Express Bug Fix は、1 つ以上のバグ・フィックスが含まれる、ソフトウェアのサブセットです。これらのバグ・フィックスは、更新のリリース・ノートにリストされます。バグ・フィックス更新を適用できるのは、同じバージョン番号を持つインストール済みのソフトウェアに対してだけです。このソフトウェアについては、ある程度のテストが行われているとはいえ、完全なテストが行われたわけではありません。自分自身でソフトウェアの妥当性を確かめるまでは、アプリケーションとともにこれらのファイルを配布しないでください。

Embedded SQL

C プログラム用のプログラミング・インタフェースです。SQL Anywhere の Embedded SQL は ANSI と IBM 規格に準拠して実装されています。

FILE

SQL Remote のレプリケーションでは、レプリケーション・メッセージのやりとりのために共有ファイルを使うメッセージ・システムのことです。これは特定のメッセージ送信システムに頼らずにテストやインストールを行うのに便利です。

参照：「レプリケーション」 553 ページ。

grant オプション

他のユーザにパーミッションを許可できるレベルのパーミッションです。

iAnywhere JDBC ドライバ

iAnywhere JDBC ドライバでは、pure Java である jConnect JDBC ドライバに比べて何らかの有利なパフォーマンスや機能を備えた JDBC ドライバが提供されます。ただし、このドライバは pure Java ソリューションではありません。iAnywhere JDBC ドライバは一般に推奨されるドライバです。

参照：

- 「JDBC」 533 ページ
- 「jConnect」 533 ページ

InfoMaker

レポート作成とデータ管理用のツールです。洗練されたフォーム、レポート、グラフ、クロスタブ、テーブルを作成できます。また、これらを基本的な構成要素とするアプリケーションも作成できます。

Interactive SQL

データベース内のデータの変更や問い合わせ、データベース構造の修正ができる、SQL Anywhere のアプリケーションです。Interactive SQL では、SQL 文を入力するためのウィンドウ枠が表示されます。また、クエリの進捗情報や結果セットを返すウィンドウ枠も表示されます。

JAR ファイル

Java アーカイブ・ファイルです。Java のアプリケーションで使用される 1 つ以上のパッケージの集合からなる圧縮ファイルのフォーマットです。Java プログラムをインストールしたり実行したりするのに必要なリソースが 1 つの圧縮ファイルにすべて収められています。

Java クラス

Java のコードの主要な構造単位です。これはプロシージャや変数の集まりで、すべてがある一定のカテゴリに関連しているためグループ化されたものです。

jConnect

JavaSoft JDBC 標準を Java で実装したものです。これにより、Java 開発者は多層／異機種環境でもネイティブなデータベース・アクセスができます。iAnywhere JDBC ドライバは一般に推奨されるドライバです。

参照：

- 「JDBC」 533 ページ
- 「iAnywhere JDBC ドライバ」 532 ページ

JDBC

Java Database Connectivity の略です。Java アプリケーションからリレーショナル・データにアクセスすることを可能にする SQL 言語プログラミング・インタフェースです。推奨 JDBC ドライバは、iAnywhere JDBC ドライバです。

参照：

- 「jConnect」 533 ページ
- 「iAnywhere JDBC ドライバ」 532 ページ

Listener

Mobile Link サーバ起動同期に使用される、dblsn という名前のプログラムです。Listener はリモート・デバイスにインストールされ、Push 通知を受け取ったときにデバイス上でアクションが開始されるように設定されます。

参照：「サーバ起動同期」 541 ページ。

LTM

LTM (Log Transfer Manager) は、Replication Agent とも呼ばれます。Replication Server と併用することで、LTM はデータベース・トランザクション・ログを読み込み、コミットされた変更を Sybase Replication Server に送信します。

参照：「Replication Server」 536 ページ。

Mobile Link

Ultra Light と SQL Anywhere のリモート・データベースを統合データベースと同期させるために設計された、セッションベース同期テクノロジです。

参照：

- 「統合データベース」 560 ページ
- 「同期」 560 ページ
- 「Ultra Light」 537 ページ

Mobile Link クライアント

2 種類の Mobile Link クライアントがあります。SQL Anywhere リモート・データベース用の Mobile Link クライアントは、dbmlsync コマンド・ライン・ユーティリティです。Ultra Light リモート・データベース用の Mobile Link クライアントは、Ultra Light ランタイム・ライブラリに組み込まれています。

Mobile Link サーバ

Mobile Link 同期を実行する、mlsrv11 という名前のコンピュータ・プログラムです。

Mobile Link システム・テーブル

Mobile Link の同期に必要なシステム・テーブルです。Mobile Link 設定スクリプトによって、Mobile Link 統合データベースにインストールされます。

Mobile Link モニタ

Mobile Link の同期をモニタするためのグラフィカル・ツールです。

Mobile Link ユーザ

Mobile Link ユーザは、Mobile Link サーバに接続するのに使用されます。Mobile Link ユーザをリモート・データベースに作成し、統合データベースに登録します。Mobile Link ユーザ名はデータベース・ユーザ名から完全に独立しています。

Notifier

Mobile Link サーバ起動同期に使用されるプログラムです。Notifier は Mobile Link サーバに統合されており、統合データベースに Push 要求がないか確認し、Push 通知を送信します。

参照：

- 「サーバ起動同期」 541 ページ
- 「Listener」 533 ページ

ODBC

Open Database Connectivity の略です。データベース管理システムに対する Windows の標準的なインタフェースです。ODBC は、SQL Anywhere がサポートするインタフェースの 1 つです。

ODBC アドミニストレータ

Windows オペレーティング・システムに付属している Microsoft のプログラムです。ODBC データ・ソースの設定に使用します。

ODBC データ・ソース

ユーザが ODBC からアクセスするデータと、そのデータにアクセスするために必要な情報の仕様です。

PDB

Palm のデータベース・ファイルです。

PowerDesigner

データベース・モデリング・アプリケーションです。これを使用すると、データベースやデータ・ウェアハウスの設計に対する構造的なアプローチが可能となります。SQL Anywhere には、PowerDesigner の Physical Data Model コンポーネントが付属します。

PowerJ

Java アプリケーション開発に使用する Sybase 製品です。

Push 通知

QAnywhere では、メッセージ転送を開始するよう QAnywhere クライアントに対して指示するために、サーバから QAnywhere クライアントに配信される特殊なメッセージです。Mobile Link サーバ起動同期では、Push 要求データや内部情報を含むデバイスに Notifer から配信される特殊なメッセージです。

参照：

- 「QAnywhere」 535 ページ
- 「サーバ起動同期」 541 ページ

Push 要求

Mobile Link サーバ起動同期において、Push 通知をデバイスに送信する必要があるかどうかを判断するために Notifier が確認する、結果セット内の値のローです。

参照：「サーバ起動同期」 541 ページ。

QAnywhere

アプリケーション間メッセージング (モバイル・デバイス間メッセージングやモバイル・デバイスとエンタープライズの間のメッセージングなど) を使用すると、モバイル・デバイスや無線デバイスで動作しているカスタム・プログラムと、集中管理されているサーバ・アプリケーションとの間で通信できます。

QAnywhere Agent

QAnywhere では、クライアント・デバイス上で動作する独立のプロセスのことです。クライアント・メッセージ・ストアをモニタリングし、メッセージを転送するタイミングを決定します。

REMOTE DBA 権限

SQL Remote では、Message Agent (dbremote) で必要なパーミッションのレベルを指します。Mobile Link では、SQL Anywhere 同期クライアント (dbmlsync) で必要なパーミッションのレベルを指します。Message Agent (dbremote) または同期クライアントがこの権限のあるユーザとして接続した場合、DBA のフル・アクセス権が与えられます。Message Agent (dbremote) または同期クライアント (dbmlsync) から接続しない場合、このユーザ ID にはパーミッションは追加されません。

参照：「DBA 権限」 531 ページ。

Replication Agent

参照：「LTM」 533 ページ。

Replication Server

SQL Anywhere と Adaptive Server Enterprise で動作する、Sybase による接続ベースのレプリケーション・テクノロジです。Replication Server は、少数のデータベース間でほぼリアルタイムのレプリケーションを行うことを目的に設計されています。

参照：「LTM」 533 ページ。

SQL

リレーショナル・データベースとの通信に使用される言語です。SQL は ANSI により標準が定義されており、その最新版は SQL-2003 です。SQL は、公認されてはいませんが、Structured Query Language の略です。

SQL Anywhere

SQLAnywhere のリレーショナル・データベース・サーバ・コンポーネントであり、主に、モバイル環境と埋め込み環境、または小規模および中規模のビジネス用のサーバとして使用されます。SQL Anywhere は、SQL Anywhere RDBMS、Ultra Light RDBMS、Mobile Link 同期ソフトウェア、その他のコンポーネントを含むパッケージの名前でもあります。

SQL Remote

統合データベースとリモート・データベース間で双方向レプリケーションを行うための、メッセージベースのデータ・レプリケーション・テクノロジです。統合データベースとリモート・データベースは、SQL Anywhere である必要があります。

SQL ベースの同期

Mobile Link では、Mobile Link イベントを使用して、テーブル・データを Mobile Link でサポートされている統合データベースに同期する方法のことです。SQL ベースの同期では、SQL を直接使用したり、Java と .NET 用の Mobile Link サーバ API を使用して SQL を返すことができます。

SQL 文

DBMS に命令を渡すために設計された、SQL キーワードを含む文字列です。

参照：

- 「スキーマ」 543 ページ
- 「SQL」 536 ページ
- 「データベース管理システム (DBMS)」 545 ページ

Sybase Central

SQL Anywhere データベースのさまざまな設定、プロパティ、ユーティリティを使用できる、グラフィカル・ユーザ・インタフェースを持つデータベース管理ツールです。Mobile Link などの他の iAnywhere 製品を管理する場合にも使用できます。

SYS

システム・オブジェクトの大半を所有する特別なユーザです。一般のユーザは SYS でログインできません。

Ultra Light

小型デバイス、モバイル・デバイス、埋め込みデバイス用に最適化されたデータベースです。対象となるプラットフォームとして、携帯電話、ポケットベル、パーソナル・オーガナイザなどが挙げられます。

Ultra Light ランタイム

組み込みの Mobile Link 同期クライアントを含む、インプロセス・リレーショナル・データベース管理システムです。Ultra Light ランタイムは、Ultra Light の各プログラミング・インタフェースで使用されるライブラリと、Ultra Light エンジンの両方に含まれます。

Windows

Windows Vista、Windows XP、Windows 200x などの、Microsoft Windows オペレーティング・システムのファミリのことです。

Windows CE

「Windows Mobile」 537 ページを参照してください。

Windows Mobile

Microsoft がモバイル・デバイス用に開発したオペレーティング・システムのファミリです。

アーティクル

Mobile Link または SQL Remote では、テーブル全体もしくはテーブル内のカラムとローのサブセットを表すデータベース・オブジェクトを指します。アーティクルの集合がパブリケーションです。

参照：

- 「レプリケーション」 553 ページ
- 「パブリケーション」 548 ページ

アップロード

同期中に、リモート・データベースから統合データベースにデータが転送される段階です。

アトミックなトランザクション

完全に処理されるかまったく処理されないことが保証される 1 つのトランザクションです。エラーによってアトミックなトランザクションの一部が処理されなかった場合は、データベースが一貫性のない状態になるのを防ぐために、トランザクションがロールバックされます。

アンロード

データベースをアンロードすると、データベースの構造かデータ、またはその両方がテキスト・ファイルにエクスポートされます (構造は SQL コマンド・ファイルに、データはカンマ区切りの ASCII ファイルにエクスポートされます)。データベースのアンロードには、アンロード・ユーティリティを使用します。

また、UNLOAD 文を使って、データから抜粋した部分だけをアンロードできます。

イベント・モデル

Mobile Link では、同期を構成する、begin_synchronization や download_cursor などの一連のイベントのことです。イベントは、スクリプトがイベント用に作成されると呼び出されます。

インクリメンタル・バックアップ

トランザクション・ログ専用のバックアップです。通常、フル・バックアップとフル・バックアップの間に使用します。

参照：「トランザクション・ログ」 547 ページ。

インデックス

ベース・テーブルにある 1 つ以上のカラムに関連付けられた、キーとポインタのソートされたセットです。テーブルの 1 つ以上のカラムにインデックスが設定されていると、パフォーマンスが向上します。

ウィンドウ

分析関数の実行対象となるローのグループです。ウィンドウには、ウィンドウ定義内のグループ化指定に従って分割されたデータの、1 つ、複数、またはすべてのローが含まれます。ウィンドウは、入力の現在のローについて計算を実行する必要があるローの数や範囲を含むように移動します。ウィンドウ構成の主な利点は、追加のクエリを実行しなくても、結果をグループ化して分析する機会が増えることです。

エージェント ID

参照：「クライアント・メッセージ・ストア ID」 540 ページ。

エンコード

文字コードとも呼ばれます。エンコードは、文字セットの各文字が情報の 1 つまたは複数のバイトにマッピングされる方法のことで、一般的に 16 進数で表現されます。UTF-8 はエンコードの例です。

参照：

- 「文字セット」 561 ページ
- 「コード・ページ」 541 ページ
- 「照合」 558 ページ

オブジェクト・ツリー

Sybase Central では、データベース・オブジェクトの階層を指します。オブジェクト・ツリーの最上位には、現在使用しているバージョンの Sybase Central がサポートするすべての製品が表示されます。それぞれの製品を拡張表示すると、オブジェクトの下位ツリーが表示されます。

参照：「Sybase Central」 537 ページ。

カーソル

結果セットへの関連付けに名前を付けたもので、プログラミング・インタフェースからローにアクセスしたり更新したりするときに使用します。SQL Anywhere では、カーソルはクエリ結果内で前方や後方への移動をサポートします。カーソルは、カーソル結果セット (通常 SELECT 文で定義される) とカーソル位置の 2 つの部分から構成されます。

参照：

- 「カーソル結果セット」 540 ページ
- 「カーソル位置」 539 ページ

カーソル位置

カーソル結果セット内の 1 つのローを指すポインタ。

参照：

- 「カーソル」 539 ページ
- 「カーソル結果セット」 540 ページ

カーソル結果セット

カーソルに関連付けられたクエリから生成されるローのセットです。

参照：

- 「カーソル」 539 ページ
- 「カーソル位置」 539 ページ

クエリ

データベースのデータにアクセスしたり、そのデータを操作したりする SQL 文や SQL 文のグループです。

参照：「SQL」 536 ページ。

クライアント／サーバ

あるアプリケーション (クライアント) が別のアプリケーション (サーバ) に対して情報を送受信するソフトウェア・アーキテクチャのことです。通常この 2 種類のアプリケーションは、ネットワークに接続された異なるコンピュータ上で実行されます。

クライアント・メッセージ・ストア

QAnywhere では、メッセージを保管するリモート・デバイスにある SQL Anywhere データベースのことです。

クライアント・メッセージ・ストア ID

QAnywhere では、Mobile Link リモート ID のことです。これによって、クライアント・メッセージ・ストアがユニークに識別されます。

グローバル・テンポラリ・テーブル

明示的に削除されるまでデータ定義がすべてのユーザに表示されるテンポラリ・テーブルです。グローバル・テンポラリ・テーブルを使用すると、各ユーザが、1 つのテーブルのまったく同じインスタンスを開くことができます。デフォルトでは、コミット時にローが削除され、接続終了時にもローが削除されます。

参照：

- 「テンポラリ・テーブル」 546 ページ
- 「ローカル・テンポラリ・テーブル」 554 ページ

ゲートウェイ

Mobile Link システム・テーブルまたは Notifier プロパティ・ファイルに保存される Mobile Link オブジェクトで、システム起動同期用のメッセージの送信方法に関する情報が含まれます。

参照：「サーバ起動同期」 541 ページ。

コード・ページ

コード・ページは、文字セットの文字を数値表示 (通常 0 ～ 255 の整数) にマッピングするエンコードです。Windows Code Page 1252 などのコード・ページがあります。このマニュアルの目的上、コード・ページとエンコードは同じ意味で使用されます。

参照：

- 「文字セット」 561 ページ
- 「エンコード」 539 ページ
- 「照合」 558 ページ

コマンド・ファイル

SQL 文で構成されたテキスト・ファイルです。コマンド・ファイルは手動で作成できますが、データベース・ユーティリティによって自動的に作成することもできます。たとえば、dbunload ユーティリティを使うと、指定されたデータベースの再構築に必要な SQL 文で構成されたコマンド・ファイルを作成できます。

サーバ・メッセージ・ストア

QAnywhere では、サーバ上のリレーショナル・データベースです。このデータベースは、メッセージを、クライアント・メッセージ・ストアまたは JMS システムに転送されるまで一時的に格納します。メッセージは、サーバ・メッセージ・ストアを介して、クライアント間で交換されます。

サーバ管理要求

XML 形式の QAnywhere メッセージです。サーバ・メッセージ・ストアを管理したり、QAnywhere アプリケーションをモニタリングするために QAnywhere システム・キューに送信されます。

サーバ起動同期

Mobile Link サーバから Mobile Link 同期を開始する方法です。

サービス

Windows オペレーティング・システムで、アプリケーションを実行するユーザ ID がログオンしていないときにアプリケーションを実行する方法です。

サブクエリ

別の SELECT 文、INSERT 文、UPDATE 文、DELETE 文、または別のサブクエリの中にネストされた SELECT 文です。

相関とネストの 2 種類のサブクエリがあります。

サブスクリプション

Mobile Link 同期では、パブリケーションと Mobile Link ユーザ間のクライアント・データベース内のリンクであり、そのパブリケーションが記述したデータの同期を可能にします。

SQL Remote レプリケーションでは、パブリケーションとリモート・ユーザ間のリンクのことで、これによりリモート・ユーザはそのパブリケーションの更新内容を統合データベースとの間で交換できます。

参照：

- 「パブリケーション」 548 ページ
- 「Mobile Link ユーザ」 534 ページ

システム・オブジェクト

SYS または dbo が所有するデータベース・オブジェクトです。

システム・テーブル

SYS または dbo が所有するテーブルです。メタデータが格納されています。システム・テーブル (データ辞書テーブルとしても知られています) はデータベース・サーバが作成し管理します。

システム・ビュー

すべてのデータベースに含まれているビューです。システム・テーブル内に格納されている情報をわかりやすいフォーマットで示します。

ジョイン

指定されたカラムの値を比較することによって 2 つ以上のテーブルにあるローをリンクする、リレーショナル・システムでの基本的な操作です。

ジョイン・タイプ

SQL Anywhere では、クロス・ジョイン、キー・ジョイン、ナチュラル・ジョイン、ON 句を使ったジョインの 4 種類のジョインが使用されます。

参照：「ジョイン」 542 ページ。

ジョイン条件

ジョインの結果に影響を及ぼす制限です。ジョイン条件は、JOIN の直後に ON 句か WHERE 句を挿入して指定します。ナチュラル・ジョインとキー・ジョインについては、SQL Anywhere がジョイン条件を生成します。

参照：

- 「ジョイン」 542 ページ
- 「生成されたジョイン条件」 559 ページ

スキーマ

テーブル、カラム、インデックス、それらの関係などを含んだデータベース構造です。

スクリプト

Mobile Link では、Mobile Link のイベントを処理するために記述されたコードです。スクリプトは、業務上の要求に適合するように、データ交換をプログラム的に制御します。

参照：「イベント・モデル」 538 ページ。

スクリプト・バージョン

Mobile Link では、同期を作成するために同時に適用される、一連の同期スクリプトです。

スクリプトベースのアップロード

Mobile Link では、ログ・ファイルを使用した方法の代わりとなる、アップロード処理のカスタマイズ方法です。

ストアド・プロシージャ

ストアド・プロシージャは、データベースに保存され、データベース・サーバに対する一連の操作やクエリを実行するために使用される SQL 命令のグループです。

スナップショット・アイソレーション

読み込み要求を発行するトランザクション用のデータのコミットされたバージョンを返す、独立性レベルの種類です。SQL Anywhere では、スナップショット、文のスナップショット、読み込み専用文のスナップショットの 3 つのスナップショットの独立性レベルがあります。スナップショット・アイソレーションが使用されている場合、読み込み処理は書き込み処理をブロックしません。

参照：「独立性レベル」 561 ページ。

セキュア機能

データベース・サーバが起動されたときに、そのデータベース・サーバで実行されているデータベースでは使用できないように -sf オプションによって指定される機能です。

セッション・ベースの同期

統合データベースとリモート・データベースの両方でデータ表現の一貫性が保たれる同期です。Mobile Link はセッション・ベースです。

ダイレクト・ロー・ハンドリング

Mobile Link では、テーブル・データを Mobile Link でサポートされている統合データベース以外のソースに同期する方法のことです。アップロードとダウンロードの両方をダイレクト・ロー・ハンドリングで実装できます。

参照：

- 「統合データベース」 560 ページ
- 「SQL ベースの同期」 537 ページ

ダウンロード

同期中に、統合データベースからリモート・データベースにデータが転送される段階です。

チェックサム

データベース・ページを使用して記録されたデータベース・ページのビット数の合計です。チェックサムを使用すると、データベース管理システムは、ページがディスクに書き込まれるときに数が一致しているかを確認することで、ページの整合性を検証できます。数が一致した場合は、ページが正常に書き込まれたとみなされます。

チェックポイント

データベースに加えたすべての変更内容がデータベース・ファイルに保存されるポイントです。通常、コミットされた変更内容はトランザクション・ログだけに保存されます。

データ・キューブ

同じ結果を違う方法でグループ化およびソートされた内容を各次元に反映した、多次元の結果セットです。データ・キューブは、セルフジョイン・クエリと相関サブクエリを必要とするデータの複雑な情報を提供します。データ・キューブは OLAP 機能の一部です。

データベース

プライマリ・キーと外部キーによって関連付けられているテーブルの集合です。これらのテーブルでデータベース内の情報が保管されます。また、テーブルとキーによってデータベースの構造が定義されます。データベース管理システムでこの情報にアクセスします。

参照：

- 「外部キー」 555 ページ
- 「プライマリ・キー」 550 ページ
- 「データベース管理システム (DBMS)」 545 ページ
- 「リレーショナル・データベース管理システム (RDBMS)」 553 ページ

データベース・オブジェクト

情報を保管したり受け取ったりするデータベース・コンポーネントです。テーブル、インデックス、ビュー、プロシージャ、トリガはデータベース・オブジェクトです。

データベース・サーバ

データベース内にある情報へのすべてのアクセスを規制するコンピュータ・プログラムです。SQL Anywhere には、ネットワーク・サーバとパーソナル・サーバの 2 種類のサーバがあります。

データベース・ファイル

データベースは 1 つまたは複数のデータベース・ファイルに保持されます。まず、初期ファイルがあり、それに続くファイルは DB 領域と呼ばれます。各テーブルは、それに関連付けられているインデックスとともに、単一のデータベース・ファイルに含まれている必要があります。

参照：「DB 領域」 531 ページ。

データベース管理システム (DBMS)

データベースを作成したり使用したりするためのプログラムの集合です。

参照：「リレーショナル・データベース管理システム (RDBMS)」 553 ページ。

データベース管理者 (DBA)

データベースの管理に必要なパーミッションを持つユーザです。DBA は、データベース・スキーマのあらゆる変更や、ユーザやグループの管理に対して、全般的な責任を負います。データベース管理者のロールはデータベース内に自動的に作成されます。その場合、ユーザ ID は DBA であり、パスワードは sql です。

データベース所有者 (dbo)

SYS が所有しないシステム・オブジェクトを所有する特別なユーザです。

参照：

- 「データベース管理者 (DBA)」 545 ページ
- 「SYS」 537 ページ

データベース接続

クライアント・アプリケーションとデータベース間の通信チャネルです。接続を確立するためには有効なユーザ ID とパスワードが必要です。接続中に実行できるアクションは、そのユーザ ID に付与された権限によって決まります。

データベース名

サーバがデータベースをロードするとき、そのデータベースに指定する名前です。デフォルトのデータベース名は、初期データベース・ファイルのルート名です。

参照：「データベース・ファイル」 545 ページ。

データ型

CHAR や NUMERIC などのデータのフォーマットです。ANSI SQL 規格では、サイズ、文字セット、照合に関する制限もデータ型に組み込めます。

参照：「ドメイン」 546 ページ。

データ操作言語 (DML)

データベース内のデータの操作に使う SQL 文のサブセットです。DML 文は、データベース内のデータを検索、挿入、更新、削除します。

データ定義言語 (DDL)

データベース内のデータの構造を定義するときに使う SQL 文のサブセットです。DDL 文は、テーブルやユーザなどのデータベース・オブジェクトを作成、変更、削除できます。

デッドロック

先へ進めない場所に一連のトランザクションが到達する状態です。

デバイス・トラッキング

Mobile Link サーバ起動同期において、デバイスを特定する Mobile Link のユーザ名を使用して、メッセージのアドレスを指定できる機能です。

参照：「サーバ起動同期」 541 ページ。

テンポラリ・テーブル

データを一時的に保管するために作成されるテーブルです。グローバルとローカルの 2 種類があります。

参照：

- 「ローカル・テンポラリ・テーブル」 554 ページ
- 「グローバル・テンポラリ・テーブル」 540 ページ

ドメイン

適切な位置に精度や小数点以下の桁数を含み、さらにオプションとしてデフォルト値や CHECK 条件などを含んでいる、組み込みデータ型のエイリアスです。ドメインには、通貨データ型のように SQL Anywhere が事前に定義したものもあります。ユーザ定義データ型とも呼ばれます。

参照：「データ型」 546 ページ。

トランザクション

作業の論理単位を構成する一連の SQL 文です。1 つのトランザクションは完全に処理されるかまったく処理されないかのどちらかです。SQL Anywhere は、ロック機能のあるトランザクション処理をサポートしているので、複数のトランザクションが同時にデータベースにアクセスしてもデータを壊すことがありません。トランザクションは、データに加えた変更を永久なものにする COMMIT 文か、トランザクション中に加えられたすべての変更を元に戻す ROLLBACK 文のいずれかで終了します。

トランザクション・ログ

データベースに対するすべての変更内容が、変更された順に格納されるファイルです。パフォーマンスを向上させ、データベース・ファイルが破損した場合でもデータをリカバリできます。

トランザクション・ログ・ミラー

オプションで設定できる、トランザクション・ログ・ファイルの完全なコピーのことで、トランザクション・ログと同時に管理されます。データベースの変更がトランザクション・ログへ書き込まれると、トランザクション・ログ・ミラーにも同じ内容が書き込まれます。

ミラー・ファイルは、トランザクション・ログとは別のデバイスに置いてください。一方のデバイスに障害が発生しても、もう一方のログにリカバリのためのデータが確保されます。

参照：「トランザクション・ログ」 547 ページ。

トランザクション単位の整合性

Mobile Link で、同期システム全体でのトランザクションの管理を保証します。トランザクション全体が同期されるか、トランザクション全体がまったく同期されないかのどちらかになります。

トリガ

データを修正するクエリをユーザが実行すると、自動的に実行されるストアド・プロシージャの特別な形式です。

参照：

- 「ロー・レベルのトリガ」 554 ページ
- 「文レベルのトリガ」 561 ページ
- 「整合性」 558 ページ

ネットワーク・サーバ

共通ネットワークを共有するコンピュータからの接続を受け入れるデータベース・サーバです。

参照：「パーソナル・サーバ」 548 ページ。

ネットワーク・プロトコル

TCP/IP や HTTP などの通信の種類です。

パーソナル・サーバ

クライアント・アプリケーションが実行されているコンピュータと同じマシンで実行されているデータベース・サーバです。パーソナル・データベース・サーバは、単一のコンピュータ上で単一のユーザが使用しますが、そのユーザからの複数の同時接続をサポートできます。

パッケージ

Java では、それぞれが互いに関連のあるクラスの集合を指します。

ハッシュ

ハッシュは、インデックスのエントリをキーに変換する、インデックスの最適化のことです。インデックスのハッシュの目的は、必要なだけの実際のロー・データをロー ID に含めることで、インデックスされた値を特定するためのローの検索、ロード、アンパックという負荷の高い処理を避けることです。

パフォーマンス統計値

データベース・システムのパフォーマンスを反映する値です。たとえば、CURRREAD 統計値は、データベース・サーバが要求したファイル読み込みのうち、現在まだ完了していないものの数を表します。

パブリケーション

Mobile Link または SQL Remote では、同期されるデータを識別するデータベース・オブジェクトのことです。Mobile Link では、クライアント上にのみ存在します。1 つのパブリケーションは複数のアーティクルから構成されています。SQL Remote ユーザは、パブリケーションに対してサブスクリプションを作成することによって、パブリケーションを受信できます。Mobile Link ユーザは、パブリケーションに対して同期サブスクリプションを作成することによって、パブリケーションを同期できます。

参照：

- 「レプリケーション」 553 ページ
- 「アーティクル」 538 ページ
- 「パブリケーションの更新」 548 ページ

パブリケーションの更新

SQL Remote レプリケーションでは、単一のデータベース内の 1 つまたは複数のパブリケーションに対して加えられた変更のリストを指します。パブリケーションの更新は、レプリケーション・メッセージの一部として定期的にリモート・データベースへ送られます。

参照：

- 「レプリケーション」 553 ページ
- 「パブリケーション」 548 ページ

パブリッシャ

SQL Remote レプリケーションでは、レプリケートできる他のデータベースとレプリケーション・メッセージを交換できるデータベースの単一ユーザを指します。

参照：「レプリケーション」 553 ページ。

ビジネス・ルール

実世界の要求に基づくガイドラインです。通常ビジネス・ルールは、検査制約、ユーザ定義データ型、適切なトランザクションの使用により実装されます。

参照：

- 「制約」 558 ページ
- 「ユーザ定義データ型」 552 ページ

ヒストグラム

ヒストグラムは、カラム統計のもっとも重要なコンポーネントであり、データ分散を表します。SQL Anywhere は、ヒストグラムを維持して、カラムの値の分散に関する統計情報をオプティマイザに提供します。

ビット配列

ビット配列は、一連のビットを効率的に保管するのに使用される配列データ構造の種類です。ビット配列は文字列に似てますが、使用される要素は文字ではなく 0 (ゼロ) と 1 になります。ビット配列は、一般的にブール値の文字列を保持するのに使用されます。

ビュー

データベースにオブジェクトとして格納される SELECT 文です。ビューを使用すると、ユーザは 1 つまたは複数のテーブルのローやカラムのサブセットを参照できます。ユーザが特定のテーブルやテーブルの組み合わせのビューを使うたびに、テーブルに保持されているデータから再計算されます。ビューは、セキュリティの目的に有用です。またデータベース情報の表示を調整して、データへのアクセスが簡単になるようにする場合も役立ちます。

ファイルベースのダウンロード

Mobile Link では、ダウンロードがファイルとして配布されるデータの同期方法であり、同期変更のオフライン配布を可能にします。

ファイル定義データベース

Mobile Link では、ダウンロード・ファイルの作成に使用される SQL Anywhere データベースのことです。

参照：「ファイルベースのダウンロード」 549 ページ。

フェールオーバ

アクティブなサーバ、システム、またはネットワークで障害や予定外の停止が発生したときに、冗長な (スタンバイ) サーバ、システム、またはネットワークに切り替えることです。フェールオーバは自動的に発生します。

プライマリ・キー

テーブル内のすべてのローをユニークに識別する値を持つカラムまたはカラムのリストです。

参照：「外部キー」 555 ページ。

プライマリ・キー制約

プライマリ・キーのカラムに対する一意性制約です。テーブルにはプライマリ・キー制約を 1 つしか設定できません。

参照：

- 「制約」 558 ページ
- 「検査制約」 557 ページ
- 「外部キー制約」 556 ページ
- 「一意性制約」 555 ページ
- 「整合性」 558 ページ

プライマリ・テーブル

外部キー関係でプライマリ・キーを含むテーブルです。

プラグイン・モジュール

Sybase Central で、製品にアクセスしたり管理したりする方法です。プラグインは、通常、インストールすると Sybase Central にもインストールされ、自動的に登録されます。プラグインは、多くの場合、Sybase Central のメイン・ウィンドウに最上位のコンテナとして、その製品名 (たとえば SQL Anywhere) で表示されます。

参照：「Sybase Central」 537 ページ。

フル・バックアップ

データベース全体をバックアップすることです。オプションでトランザクション・ログのバックアップも可能です。フル・バックアップには、データベース内のすべての情報が含まれており、システム障害やメディア障害が発生した場合の保護として機能します。

参照：「インクリメンタル・バックアップ」 538 ページ。

プロキシ・テーブル

メタデータを含むローカル・テーブルです。リモート・データベース・サーバのテーブルに、ローカル・テーブルであるかのようにアクセスするときに使用します。

参照：「メタデータ」 551 ページ。

ベース・テーブル

データを格納する永久テーブルです。テーブルは、テンポラリ・テーブルやビューと区別するために、「ベース・テーブル」と呼ばれることがあります。

参照：

- 「テンポラリ・テーブル」 546 ページ
- 「ビュー」 549 ページ

ポーリング

Mobile Link サーバ起動同期において、Mobile Link Listerner などのライト・ウェイト・ポーラが Notifier から Push 通知を要求する方法です。

参照：「サーバ起動同期」 541 ページ。

ポリシー

QAnywhere では、メッセージ転送の発生時期を指定する方法のことです。

マテリアライズド・ビュー

計算され、ディスクに保存されたビューのことです。マテリアライズド・ビューは、ビュー (クエリ指定を使用して定義される) とテーブル (ほとんどのテーブルの操作をそのテーブル上で実行できる) の両方の特性を持ちます。

参照：

- 「ベース・テーブル」 551 ページ
- 「ビュー」 549 ページ

ミラー・ログ

参照：「トランザクション・ログ・ミラー」 547 ページ。

メタデータ

データについて説明したデータです。メタデータは、他のデータの特質と内容について記述しています。

参照：「スキーマ」 543 ページ。

メッセージ・システム

SQL Remote のレプリケーションでは、統合データベースとリモート・データベースの間でのメッセージのやりとりに使用するプロトコルのことです。SQL Anywhere では、FILE、FTP、SMTP のメッセージ・システムがサポートされています。

参照：

- 「レプリケーション」 553 ページ
- 「FILE」 532 ページ

メッセージ・ストア

QAnywhere では、メッセージを格納するクライアントおよびサーバ・デバイスのデータベースのことです。

参照：

- 「クライアント・メッセージ・ストア」 540 ページ
- 「サーバ・メッセージ・ストア」 541 ページ

メッセージ・タイプ

SQL Remote のレプリケーションでは、リモート・ユーザと統合データベースのパブリッシャとの通信方法を指定するデータベース・オブジェクトのことを指します。統合データベースには、複数のメッセージ・タイプが定義されていることがあります。これによって、リモート・ユーザはさまざまなメッセージ・システムを使って統合データベースと通信できることになります。

参照：

- 「レプリケーション」 553 ページ
- 「統合データベース」 560 ページ

メッセージ・ログ

データベース・サーバや Mobile Link サーバなどのアプリケーションからのメッセージを格納できるログです。この情報は、メッセージ・ウィンドウに表示されたり、ファイルに記録されたりすることもあります。メッセージ・ログには、情報メッセージ、エラー、警告、MESSAGE 文からのメッセージが含まれます。

メンテナンス・リリース

メンテナンス・リリースは、同じメジャー・バージョン番号を持つ旧バージョンのインストール済みソフトウェアをアップグレードするための完全なソフトウェア・セットです (バージョン番号のフォーマットは、*メジャー.マイナー.パッチ.ビルド*です)。バグ・フィックスとその他の変更については、アップグレードのリリース・ノートにリストされます。

ユーザ定義データ型

参照：「ドメイン」 546 ページ。

ライト・ウェイト・ポーラ

Mobile Link サーバ起動同期において、Mobile Link サーバからの Push 通知をポーリングするデバイス・アプリケーションです。

参照：「サーバ起動同期」 541 ページ。

リダイレクタ

クライアントと Mobile Link サーバ間で要求と応答をルート指定する Web サーバ・プラグインです。このプラグインによって、負荷分散メカニズムとフェールオーバ・メカニズムも実装されます。

リファレンス・データベース

Mobile Link では、Ultra Light クライアントの開発に使用される SQL Anywhere データベースです。開発中は、1 つの SQL Anywhere データベースをリファレンス・データベースとしても統合データベースとしても使用できます。他の製品によって作成されたデータベースは、リファレンス・データベースとして使用できません。

リモート ID

SQL Anywhere と Ultra Light データベース内のユニークな識別子で、Mobile Link によって使用されます。リモート ID は NULL に初期設定されていますが、データベースの最初の同期時に GUID に設定されます。

リモート・データベース

Mobile Link または SQL Remote では、統合データベースとデータを交換するデータベースを指します。リモート・データベースは、統合データベース内のすべてまたは一部のデータを共有できます。

参照：

- 「同期」 560 ページ
- 「統合データベース」 560 ページ

リレーショナル・データベース管理システム (RDBMS)

関連するテーブルの形式でデータを格納するデータベース管理システムです。

参照：「データベース管理システム (DBMS)」 545 ページ。

レプリケーション

物理的に異なるデータベース間でデータを共有することです。Sybase では、Mobile Link、SQL Remote、Replication Server の 3 種類のレプリケーション・テクノロジを提供しています。

レプリケーション・メッセージ

SQL Remote または Replication Server では、パブリッシュするデータベースとサブスクリプションを作成するデータベース間で送信される通信内容を指します。メッセージにはデータを含み、レプリケーション・システムで必要なパススルー文、情報があります。

参照：

- 「レプリケーション」 553 ページ
- 「パブリケーションの更新」 548 ページ

レプリケーションの頻度

SQL Remote レプリケーションでは、リモート・ユーザに対する設定の 1 つで、パブリッシャの Message Agent がレプリケーション・メッセージを他のリモート・ユーザに送信する頻度を定義します。

参照：「レプリケーション」 553 ページ。

ロー・レベルのトリガ

変更されているローごとに一回実行するトリガです。

参照：

- 「トリガ」 547 ページ
- 「文レベルのトリガ」 561 ページ

ローカル・テンポラリ・テーブル

複合文を実行する間だけ存在したり、接続が終了するまで存在したりするテンポラリ・テーブルです。データのセットを 1 回だけロードする必要がある場合にローカル・テンポラリ・テーブルが便利です。デフォルトでは、COMMIT を実行するとローが削除されます。

参照：

- 「テンポラリ・テーブル」 546 ページ
- 「グローバル・テンポラリ・テーブル」 540 ページ

ロール

概念データベース・モデルで、ある視点からの関係を説明する動詞またはフレーズを指します。各関係は 2 つのロールを使用して表すことができます。"contains (A は B を含む)" や "is a member of (B は A のメンバ)" などのロールがあります。

ロールバック・ログ

コミットされていない各トランザクションの最中に行われた変更のレコードです。ROLLBACK 要求やシステム障害が発生した場合、コミットされていないトランザクションはデータベースから破棄され、データベースは前の状態に戻ります。各トランザクションにはそれぞれロールバック・ログが作成されます。このログは、トランザクションが完了すると削除されます。

参照：「トランザクション」 547 ページ。

ロール名

外部キーの名前です。この外部キーがロール名と呼ばれるのは、外部テーブルとプライマリ・テーブル間の関係に名前を指定するためです。デフォルトでは、テーブル名がロール名になります。ただし、別の外部キーがそのテーブル名を使用している場合、デフォルトのロール名はテーブル名に 3 桁のユニークな数字を付けたものになります。ロール名は独自に作成することもできます。

参照：「外部キー」 555 ページ。

ログ・ファイル

SQL Anywhere によって管理されているトランザクションのログです。ログ・ファイルを使用すると、システム障害やメディア障害が発生してもデータベースを回復させることができます。また、データベースのパフォーマンスを向上させたり、SQL Remote を使用してデータをレプリケートしたりする場合にも使用できます。

参照：

- 「トランザクション・ログ」 547 ページ
- 「トランザクション・ログ・ミラー」 547 ページ
- 「フル・バックアップ」 550 ページ

ロック

複数のトランザクションを同時に実行しているときにデータの整合性を保護する同時制御メカニズムです。SQL Anywhere では、2 つの接続によって同じデータが同時に変更されないようにするために、また変更処理の最中に他の接続によってデータが読み込まれないようにするために、自動的にロックが適用されます。

ロックの制御は、独立性レベルを設定して行います。

参照：

- 「独立性レベル」 561 ページ
- 「同時性 (同時実行性)」 561 ページ
- 「整合性」 558 ページ

ワーク・テーブル

クエリの最適化の最中に中間結果を保管する内部保管領域です。

一意性制約

NULL 以外のすべての値が重複しないことを要求するカラムまたはカラムのセットに対する制限です。テーブルには複数の一意性制約を指定できます。

参照：

- 「外部キー制約」 556 ページ
- 「プライマリ・キー制約」 550 ページ
- 「制約」 558 ページ

解析ツリー

クエリを代数で表現したものです。

外部キー

別のテーブルにあるプライマリ・キーの値を複製する、テーブルの 1 つ以上のカラムです。テーブル間の関係は、外部キーによって確立されます。

参照：

- 「プライマリ・キー」 550 ページ
- 「外部テーブル」 556 ページ

外部キー制約

カラムまたはカラムのセットに対する制約で、テーブルのデータが別のテーブルのデータとどのように関係しているかを指定するものです。カラムのセットに外部キー制約を加えると、それらのカラムが外部キーになります。

参照：

- 「制約」 558 ページ
- 「検査制約」 557 ページ
- 「プライマリ・キー制約」 550 ページ
- 「一意性制約」 555 ページ

外部ジョイン

テーブル内のすべてのローを保護するジョインです。SQL Anywhere では、左外部ジョイン、右外部ジョイン、全外部ジョインがサポートされています。左外部ジョインは JOIN 演算子の左側にあるテーブルのローを保護し、右側にあるテーブルのローがジョイン条件を満たさない場合には NULL を返します。全外部ジョインは両方のテーブルに含まれるすべてのローを保護します。

参照：

- 「ジョイン」 542 ページ
- 「内部ジョイン」 561 ページ

外部テーブル

外部キーを持つテーブルです。

参照：「外部キー」 555 ページ。

外部ログイン

リモート・サーバとの通信に使用される代替のログイン名とパスワードです。デフォルトでは、SQL Anywhere は、クライアントに代わってリモート・サーバに接続するときは、常にそのクライアントの名前とパスワードを使用します。外部ログインを作成することによって、このデフォルトを上書きできます。外部ログインは、リモート・サーバと通信するときに使用する代替のログイン名とパスワードです。

競合

リソースについて対立する動作のことです。たとえば、データベース用語では、複数のユーザがデータベースの同じローを編集しようとした場合、そのローの編集権についての競合が発生します。

競合解決

Mobile Link では、競合解決は 2 人のユーザが別々のリモート・データベースの同じローを変更した場合にどう処理するかを指定するロジックのことです。

検査制約

指定された条件をカラムやカラムのセットに課す制約です。

参照：

- 「制約」 558 ページ
- 「外部キー制約」 556 ページ
- 「プライマリ・キー制約」 550 ページ
- 「一意性制約」 555 ページ

検証

データベース、テーブル、またはインデックスについて、特定のタイプのファイル破損をテストすることです。

作成者 ID

Ultra Light の Palm OS アプリケーションでは、アプリケーションが作成されたときに割り当てられる ID のことです。

参照元オブジェクト

テーブルなどのデータベースの別のオブジェクトをオブジェクト定義が直接参照する、ビューなどのオブジェクトです。

参照：「外部キー」 555 ページ。

参照整合性

データの整合性、特に異なるテーブルのプライマリ・キー値と外部キー値との関係を管理する規則を厳守することです。参照整合性を備えるには、それぞれの外部キーの値が、参照テーブルにあるローのプライマリ・キー値に対応するようにします。

参照：

- 「プライマリ・キー」 550 ページ
- 「外部キー」 555 ページ

参照先オブジェクト

ビューなどの別のオブジェクトの定義で直接参照される、テーブルなどのオブジェクトです。

参照：「プライマリ・キー」 550 ページ。

識別子

テーブルやカラムなどのデータベース・オブジェクトを参照するときに使う文字列です。A ～ Z、a ～ z、0 ～ 9、アンダースコア (_)、アットマーク (@)、シャープ記号 (#)、ドル記号 ($) のうち、任意の文字を識別子として使用できます。

述部

条件式です。オプションで論理演算子 AND や OR と組み合わせて、WHERE 句または HAVING 句に条件のセットを作成します。SQL では、unknown と評価される述部が false と解釈されます。

照合

データベース内のテキストのプロパティを定義する文字セットとソート順の組み合わせのことです。SQL Anywhere データベースでは、サーバを実行しているオペレーティング・システムと言語によって、デフォルトの照合が決まります。たとえば、英語版 Windows システムのデフォルトの照合は 1252LATIN1 です。照合は、照合順とも呼ばれ、文字列の比較とソートに使用します。

参照：

- 「文字セット」 561 ページ
- 「コード・ページ」 541 ページ
- 「エンコード」 539 ページ

世代番号

Mobile Link では、リモート・データベースがデータをアップロードしてからダウンロード・ファイルを適用するようにするためのメカニズムのことです。

参照：「ファイルベースのダウンロード」 549 ページ。

制約

テーブルやカラムなど、特定のデータベース・オブジェクトに含まれた値に関する制約です。たとえば、一意性制約があるカラム内の値は、すべて異なっている必要があります。テーブルに、そのテーブルの情報と他のテーブルのデータがどのように関係しているのかを指定する外部キー制約が設定されていることもあります。

参照：

- 「検査制約」 557 ページ
- 「外部キー制約」 556 ページ
- 「プライマリ・キー制約」 550 ページ
- 「一意性制約」 555 ページ

整合性

データが適切かつ正確であり、データベースの関係構造が保たれていることを保証する規則を厳守することです。

参照：「参照整合性」 557 ページ。

正規化

データベース・スキーマを改善することです。リレーショナル・データベース理論に基づく規則に従って、冗長性を排除したり、編成を改良します。

正規表現

正規表現は、文字列内で検索するパターンを定義する、一連の文字、ワイルドカード、演算子です。

生成されたジョイン条件

自動的に生成される、ジョインの結果に対する制限です。キーとナチュラルの 2 種類があります。キー・ジョインは、KEY JOIN を指定したとき、またはキーワード JOIN を指定したが、CROSS、NATURAL、または ON を使用しなかった場合に生成されます。キー・ジョインの場合、生成されたジョイン条件はテーブル間の外部キー関係に基づいています。ナチュラル・ジョインは NATURAL JOIN を指定したときに生成され、生成されたジョイン条件は、2 つのテーブルの共通のカラム名に基づきます。

参照：

- 「ジョイン」 542 ページ
- 「ジョイン条件」 543 ページ

接続 ID

クライアント・アプリケーションとデータベース間の特定の接続に付けられるユニークな識別番号です。現在の接続 ID を確認するには、次の SQL 文を使用します。

```
SELECT CONNECTION_PROPERTY( 'Number' );
```

接続プロファイル

ユーザ名、パスワード、サーバ名などの、データベースに接続するために必要なパラメータのセットです。便宜的に保管され使用されます。

接続起動同期

Mobile Link のサーバ起動同期の 1 つの形式で、接続が変更されたときに同期が開始されます。

参照：「サーバ起動同期」 541 ページ。

相関名

クエリの FROM 句内で使用されるテーブルやビューの名前です。テーブルやビューの元の名前か、FROM 句で定義した代替名のいずれかになります。

抽出

SQL Remote レプリケーションでは、統合データベースから適切な構造とデータをアンロードする動作を指します。この情報は、リモート・データベースを初期化するときに使用されます。

参照：「レプリケーション」 553 ページ。

通信ストリーム

Mobile Link では、Mobile Link クライアントと Mobile Link サーバ間での通信にネットワーク・プロトコルが使用されます。

転送ルール

QAnywhere では、メッセージの転送を発生させる時期、転送するメッセージ、メッセージを削除する時期を決定する論理のことです。

統合データベース

分散データベース環境で、データのマスタ・コピーを格納するデータベースです。競合や不一致が発生した場合、データのプライマリ・コピーは統合データベースにあるとみなされます。

参照：

- 「同期」 560 ページ
- 「レプリケーション」 553 ページ

統合化ログイン

オペレーティング・システムへのログイン、ネットワークへのログイン、データベースへの接続に、同一のユーザ ID とパスワードを使用するログイン機能の 1 つです。

動的 SQL

実行される前に作成したプログラムによって生成される SQL です。Ultra Light の動的 SQL は、占有容量の小さいデバイス用に設計された変形型です。

同期

Mobile Link テクノロジを使用してデータベース間でデータをレプリケートする処理です。

SQL Remote では、同期はデータの初期セットを使ってリモート・データベースを初期化する処理を表すために特に使用されます。

参照：

- 「Mobile Link」 533 ページ
- 「SQL Remote」 536 ページ

同時性 (同時実行性)

互いに独立し、場合によっては競合する可能性のある 2 つ以上の処理を同時に実行することです。SQL Anywhere では、自動的にロックを使用して各トランザクションを独立させ、同時に稼働するそれぞれのアプリケーションが一貫したデータのセットを参照できるようにします。

参照：

- 「トランザクション」 547 ページ
- 「独立性レベル」 561 ページ

独立性レベル

あるトランザクションの操作が、同時に処理されている別のトランザクションの操作からどの程度参照できるかを示します。独立性レベルには 0 から 3 までの 4 つのレベルがあります。最も高い独立性レベルには 3 が設定されます。デフォルトでは、レベルは 0 に設定されています。SQL Anywhere では、スナップショット、文のスナップショット、読み込み専用文のスナップショットの 3 つのスナップショットの独立性レベルがあります。

参照：「スナップショット・アイソレーション」 543 ページ。

内部ジョイン

2 つのテーブルがジョイン条件を満たす場合だけ、結果セットにローが表示されるジョインです。内部ジョインがデフォルトです。

参照：

- 「ジョイン」 542 ページ
- 「外部ジョイン」 556 ページ

物理インデックス

インデックスがディスクに保存されるときの実際のインデックス構造です。

文レベルのトリガ

トリガ付きの文の処理が完了した後に実行されるトリガです。

参照：

- 「トリガ」 547 ページ
- 「ロー・レベルのトリガ」 554 ページ

文字セット

文字セットは記号、文字、数字、スペースなどから成ります。"ISO-8859-1" は文字セットの例です。Latin1 とも呼ばれます。

参照：

- 「コード・ページ」 541 ページ
- 「エンコード」 539 ページ
- 「照合」 558 ページ

文字列リテラル

文字列リテラルとは、一重引用符 (') で囲まれ、シーケンスで並べられた文字のことです。

論理インデックス

物理インデックスへの参照 (ポインタ) です。ディスクに保存される論理インデックス用のインデックス構造はありません。

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

は

ひ

ふ

ヘ

ほ

ま

み

め

も

ゆ

よ

ら

り

れ

ろ

わ

