iAnywhere SOLUTIONS™
A SYBASE COMPANY

Ultra Light データベース
ユーザーズ・ガイド

パート番号：DC37717-01-0902-01
改訂：2005 年 3 月

目次

はじめに

このマニュアルの内容 このマニュアルでは、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 Studio のマニュアル

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

SQL Anywhere Studio のマニュアル

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

- **『SQL Anywhere Studio の紹介』** このマニュアルでは、SQL Anywhere Studio のデータベース管理と同期テクノロジの概要について説明します。また、SQL Anywhere Studio を構成する各部分について説明するチュートリアルも含まれています。
- **『SQL Anywhere Studio 新機能ガイド』** このマニュアルは、SQL Anywhere Studio のこれまでのリリースのユーザを対象としています。ここでは、製品の今回のリリースと以前のリリースで導入された新機能をリストし、アップグレード手順を説明しています。
- **『Adaptive Server Anywhere データベース管理ガイド』** このマニュアルでは、データベースおよびデータベース・サーバの実行、管理、設定について説明しています。
- **『Adaptive Server Anywhere SQL ユーザーズ・ガイド』** このマニュアルでは、データベースの設計と作成の方法、データのインポート・エクスポート・変更の方法、データの検索方法、ストアド・プロシージャとトリガの構築方法について説明します。
- **『Adaptive Server Anywhere SQL リファレンス・マニュアル』** このマニュアルは、Adaptive Server Anywhere で使用する SQL 言語の完全なリファレンスです。また、Adaptive Server Anywhere のシステム・テーブルとシステム・プロシージャについても説明しています。
- **『Adaptive Server Anywhere プログラミング・ガイド』** このマニュアルでは、C、C++、Java プログラミング言語を使用してデータベース・アプリケーションを構築、配備する方法につい

て説明します。Visual Basic や PowerBuilder などのツールのユーザは、それらのツールのプログラミング・インタフェースを使用できます。また、Adaptive Server Anywhere ADO.NET データ・プロバイダについても説明します。

- **『Adaptive Server Anywhere SNMP Extension Agent ユーザーズ・ガイド』** このマニュアルでは、Adaptive Server Anywhere SNMP Extension Agent を SNMP 管理アプリケーションとともに使用できるように設定して、Adaptive Server Anywhere データベースを管理できるようにする方法を説明します。

- **『Adaptive Server Anywhere エラー・メッセージ』** このマニュアルでは、Adaptive Server Anywhere エラー・メッセージの完全なリストを、その診断情報とともに説明します。

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

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

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

- **『Mobile Link サーバ起動同期ユーザーズ・ガイド』** このマニュアルでは、Mobile Link のサーバによって開始される同期について説明します。サーバによって開始される同期とは、統合データベースから同期の開始を可能にする Mobile Link の機能です。

- **『Mobile Link チュートリアル』** このマニュアルには、Mobile Link アプリケーションの設定と実行を行う方法を説明するチュートリアルがいくつか用意されています。
- **『QAnywhere ユーザーズ・ガイド』** このマニュアルでは、Mobile Link QAnywhere について説明します。Mobile Link QAnywhere は、従来のデスクトップ・クライアントやラップトップ・クライアントだけでなく、モバイル・クライアントや無線クライアント用のメッセージング・アプリケーションの開発と展開を可能にするメッセージング・プラットフォームです。
- **『Mobile Link およびリモート・データ・アクセスの ODBC ドライバ』** このマニュアルでは、Mobile Link 同期サーバから、または Adaptive Server Anywhere リモート・データ・アクセスによって、Adaptive Server Anywhere 以外の統合データベースにアクセスするための ODBC ドライバの設定方法について説明します。
- **『SQL Remote ユーザーズ・ガイド』** このマニュアルでは、モバイル・コンピューティング用の SQL Remote データ・レプリケーション・システムについて、あらゆる角度から説明します。このシステムによって、Adaptive Server Anywhere または Adaptive Server Enterprise の単一データベースと Adaptive Server Anywhere の複数データベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。
- **『SQL Anywhere Studio ヘルプ』** このマニュアルには、Sybase Central や Interactive SQL、その他のグラフィカル・ツールに関するコンテキスト別のヘルプが含まれています。これは、製本版マニュアル・セットには含まれていません。
- **『Ultra Light データベース・ユーザーズ・ガイド』** このマニュアルは、Ultra Light 開発者を対象としています。ここでは、Ultra Light データベース・システムの概要について説明します。また、すべての Ultra Light プログラミング・インタフェースに共通する情報を提供します。
- **Ultra Light のインタフェースに関するマニュアル** 各 Ultra Light プログラミング・インタフェースには、それぞれに対応するマニュアルを用意しています。これらのインタフェースは、RAD (

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

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

マニュアルの形式

SQL Anywhere Studio のマニュアルは、次の形式で提供されています。

- **オンライン・マニュアル** オンライン・マニュアルには、SQL Anywhere Studio の完全なマニュアルがあり、SQL Anywhere ツールに関する印刷マニュアルとコンテキスト別のヘルプの両方が含まれています。オンライン・マニュアルは、製品のメンテナンス・リリースごとに更新されます。これは、最新の情報を含む最も完全なマニュアルです。

  Windows オペレーティング・システムでオンライン・マニュアルにアクセスするには、[スタート] ー [プログラム] ー [SQL Anywhere 9] ー [オンライン・マニュアル] を選択します。オンライン・マニュアルをナビゲートするには、左ウィンドウ枠で HTML ヘルプの目次、索引、検索機能を使用し、右ウィンドウ枠でリンク情報とメニューを使用します。

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

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

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

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

表記の規則

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

SQL 構文の表記規則

SQL 構文の表記には、次の規則が適用されます。

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

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

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

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

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

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

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

- **オプション部分** 文のオプション部分は角カッコで囲みます。

  RELEASE SAVEPOINT [*savepoint-name*]

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

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

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

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

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

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

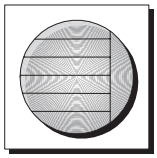
  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

グラフィック・アイコン

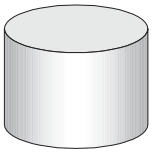
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

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



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



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

- プログラミング・インタフェース

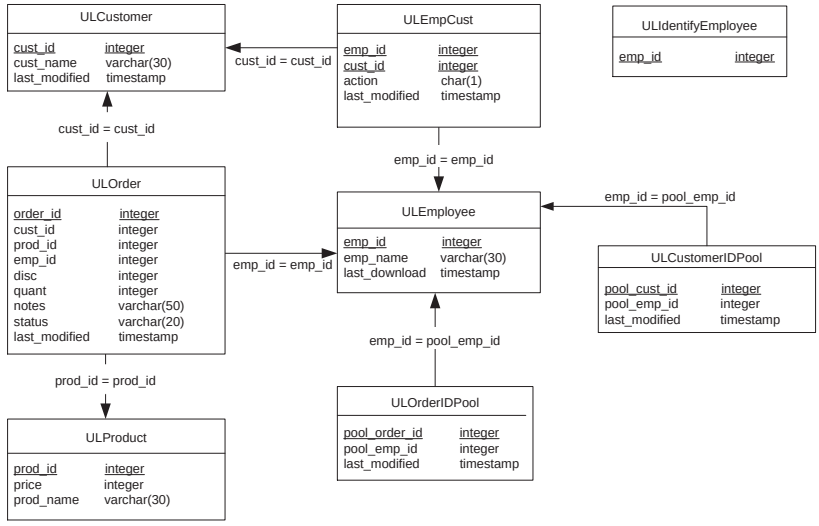
CustDB サンプル・データベース

Mobile Link と Ultra Light のマニュアルでは、多くの例で Ultra Light のサンプル・データベースが使用されています。

Ultra Light サンプル・データベースのリファレンス・データベースは、*custdb.db* という名前のファイルに保存され、SQL Anywhere ディレクトリのサブディレクトリ *Samples¥UltraLite¥CustDB* に置かれています。全面的にこのデータベースを使用して構築したアプリケーションも提供されています。

サンプル・データベースは、あるハードウェア販売会社の販売管理データベースです。データベースには、この販売会社の顧客、製品、営業戦力に関する情報が入っています。

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



詳細情報の検索／フィードバックの提供

詳細情報の検索

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

ご質問がある場合や支援が必要な場合は、次に示す iAnywhere Solutions 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 Studio バージョンのビルド番号を明記し、現在発生している問題について詳しくお知らせくださいますようお願いいたします。バージョン情報は、コマンド・プロンプトで **dbeng9 -v** と入力して確認できます。

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

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

ニュースグループに関するお断り

iAnywhere Solutions は、ニュースグループ上に解決策、情報、または意見を提供する義務を負うものではありません。また、システム・オペレータ以外のスタッフにこのサービスを監視させて、操作状況や可用性を保証する義務もありません。

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

フィードバック

このマニュアルに関するご意見、ご提案、フィードバックをお寄せください。

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

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

第 1 部　Ultra Light データベース

第 1 部では、小型デバイス用の Ultra Light リレーショナル・データベース・システムの概要を説明し、Ultra Light データベースの一般的な機能について説明します。

第 1 章

Ultra Light へようこそ

この章の内容

この章では、Ultra Light の特徴、プラットフォーム、アーキテクチャ、機能を紹介します。

Ultra Light の概要

Ultra Light は、小型デバイス、モバイル・デバイス、埋め込みデバイスを対象としたリレーショナル・データベース管理および同期システムです。Ultra Light には、次の利点があります。

- **堅牢なデータ管理** 小型デバイスに格納されているデータは、エンタープライズ・データベースに格納されているデータと同様に重要です。Ultra Light によって、リレーショナル・データベース・システムのトランザクション処理、参照整合性などの利点を小型デバイスで使用できます。

  Ultra Light データベース機能の詳細については、「Ultra Light データベース機能」7 ページを参照してください。

- **強力な同期** Ultra Light を使用すると、中央のデータベースでデータを同期できます。

  Ultra Light は、SQL Anywhere Studio に含まれる Mobile Link 同期テクノロジを使用して、業界標準のデータベース管理システムと同期を行います。Mobile Link 同期は、Sybase Adaptive Server Anywhere、Sybase Adaptive Server Enterprise、IBM DB2、Microsoft SQL Server、Oracle と動作できます。Mobile Link は、柔軟性が高く、プログラム可能でスケーラブルな、数千の Ultra Light データベースを管理できる同期システムです。

  詳細については、『Mobile Link クライアント』> 「Ultra Light クライアント」を参照してください。

- **プログラミング・インタフェースの選択** Ultra Light コンポーネントには、データに簡単にアクセスできるオブジェクトベースのプログラミング・インタフェースを使用できるオプションが用意されています。Visual Studio .NET、AppForge MobileVB と Crossfire、Borland JBuilder、eMbedded Visual Basic などの一般的な開発ツールの統合によって、開発者の生産性が向上します。グラフィカル・ツールを使用して、Ultra Light データベースのスキーマをすばやく設計したり変更したりできます。

詳細については、「Ultra Light プログラミング・インタフェース」 5 ページと「Ultra Light プログラミング・インタフェースの選択」 12 ページを参照してください。

- **マルチプラットフォームのサポート**　Windows CE、Palm OS、Windows XP、Java ベースのデバイスに対応する Ultra Light データベース・アプリケーションを開発して配備できます。

  詳細については、『SQL Anywhere Studio の紹介』>「Ultra Light 開発プラットフォーム」を参照してください。

Ultra Light プログラミング・インタフェース

Ultra Light には、さまざまなプログラミング・インタフェースが用意されており、これらは各種一般的なプログラミング・ツールに統合されます。各インタフェースには、同じ基本 Ultra Light ランタイム・ライブラリが使用されます。

これらのインタフェースは、「**コンポーネント**」と「**静的インタフェース**」の 2 つのカテゴリに分けられます。各インタフェースには独自の長所があり、それぞれに適した用途があります。プログラミング・インタフェースの選択に関するヒントについては、「Ultra Light プログラミング・インタフェースの選択」 12 ページを参照してください。

- **Ultra Light コンポーネント** Ultra Light コンポーネントは、リレーショナル・データベースおよび同期機能を開発ツールのユーザに提供します。これらのコンポーネントには、サポートされている開発ツールごとに使い慣れたインタフェースが用意されています。Ultra Light コンポーネントは、簡単なテーブル・ベースのデータ・アクセス・インタフェースのほか、より複雑なクエリ用として動的 SQL も備えています。

  次のコンポーネントを使用できます。

  - **Ultra Light for MobileVB** Microsoft Visual Basic に対する AppForge MobileVB 拡張または Microsoft Visual Studio .NET に対する AppForge Crossfire 拡張を使用した開発。

『Ultra Light for MobileVB ユーザーズ・ガイド』> 「Ultra Light for MobileVB ユーザーズ・ガイド」を参照してください。

- **Ultra Light ActiveX** eMbedded Visual Basic または JScript と Pocket IE を使用した開発。

  『Ultra Light ActiveX ユーザーズ・ガイド』> 「Ultra Light ActiveX ユーザーズ・ガイド」を参照してください。

- **Native Ultra Light for Java** サポートされている JDK を使用した開発。Ultra Light コンポーネント自体は、改善したパフォーマンス用としてネイティブ (C++) メソッドにアクセスします。

  『Native Ultra Light for Java ユーザーズ・ガイド』> 「Native Ultra Light for Java ユーザーズ・ガイド」を参照してください。

- **Ultra Light.NET** ADO.NET プログラミング・インタフェースを含む Visual Studio .NET を使用した開発。

  『Ultra Light.NET ユーザーズ・ガイド』> 「Ultra Light.NET ユーザーズ・ガイド」を参照してください。

- **Ultra Light C++ コンポーネント** C++ インタフェースを使用した開発。

  『Ultra Light C/C++ ユーザーズ・ガイド』> 「Ultra Light C/C++ ユーザーズ・ガイド」を参照してください。

- **Ultra Light for M-Business Anywhere** M-Business Anywhere を使用した開発。

  『UltraLite for M-Business Anywhere User's Guide』> 「UltraLite for M-Business Anywhere User's Guide」を参照してください。

- **ODBC** Ultra Light は、ODBC プログラミング・インタフェースのサブセットをサポートします。

『Ultra Light C/C++ ユーザーズ・ガイド』> 「チュートリアル：ODBC を使用したアプリケーションの構築」と『Ultra Light C/C++ ユーザーズ・ガイド』> 「Ultra Light ODBC API リファレンス」を参照してください。

- **静的インタフェース** 静的インタフェースは、プリプロセッサ・ベースのインタフェースに慣れた C/C++ および Java 開発者に対してリッチな SQL インタフェースを提供します。アプリケーションで使用するすべての SQL 文は、コンパイル時に定義します。

  次の静的インタフェースを使用できます。

  - **Embedded SQL と静的型 C++ API** C/C++ と Embedded SQL 文を使用した開発。

    『Ultra Light C/C++ ユーザーズ・ガイド』> 「Ultra Light C/C++ ユーザーズ・ガイド」を参照してください。

  - **Ultra Light 静的型 Java** JDBC インタフェースを使用した pure Java での開発。このインタフェースは、他の Ultra Light インタフェースに対しては別のランタイム・ライブラリを使用します。

    『Ultra Light 静的型 Java ユーザーズ・ガイド』> 「Ultra Light 静的型 Java ユーザーズ・ガイド」を参照してください。

Ultra Light データベース機能

Ultra Light の機能は次のとおりです。

- **テーブル** 単一の Ultra Light データベース・ファイルに、多数のテーブルを保持できます。テーブルのカラムの数と型は設計時に固定されますが、各テーブルに任意の数のロー (最大 64 K) を置くことができます。各ローには、カラムごとに 1 つのエントリがあります。エントリの値がない場合は、特殊な NULL エントリが使用されます。

データベースを設計するときは、顧客や従業員など、各テーブルが異なる型の項目を表すようにしてください。

- **データ型** Ultra Light データベースは、あらゆるデータ型、デフォルト値、NULL 値を保持できます。
- **インデックス** リレーショナル・データベース・テーブルのローは、順序付けされていません。インデックスを作成すると、順番にローにアクセスし、データに迅速にアクセスできます。通常、インデックスは 1 つのカラムと関連付けられますが、Ultra Light にはマルチカラム・インデックスもあります。
- **キー** 各テーブルには、「**プライマリ・キー**」という特殊なインデックスがあります。プライマリ・キー・カラムのエントリはユニークにします。

  「**外部キー**」は、1 つのテーブルのデータを他のテーブルのデータと関連付けます。外部キー・カラムの各エントリを、他のテーブルのプライマリ・キーのエントリと対応させます。

  テーブル間のプライマリ・キーと外部キーはデータベースの「**参照整合性**」を維持します。Ultra Light データベースでは、参照整合性が確保されているため、たとえば、該当する顧客がデータベースに存在しない場合は、その顧客の注文を入力できません。

  参照整合性を確保することで、Ultra Light データベース内のデータが正しく、同様にエンタープライズの他の場所にあるデータも正しいことが保証されます。参照整合性が操作のコミット時にチェックされるため、データを変更する順序の柔軟性が高まります。
- **パブリケーション** Ultra Light データベースのデータを他のデータベースと同期する場合は、有効な SQL Anywhere Studio ライセンスが必要です。SQL Anywhere Studio の Mobile Link 同期テクノロジは、Ultra Light データベースをデスクトップ、ワークグループ、またはエンタープライズのデータベースと同期します。

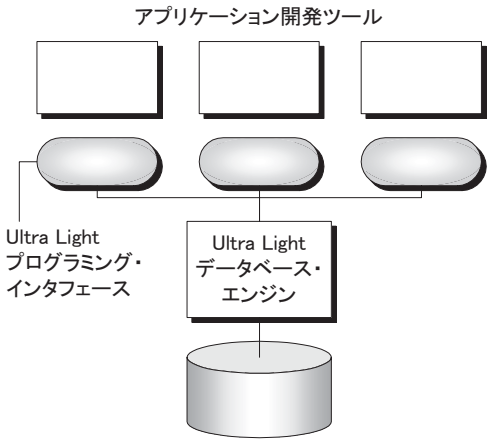
パブリケーションは、同期するデータ・セットを定義します。通常は、Ultra Light データベースのすべてのデータを同期した方が効果的ですが、パブリケーションを使用すると、より柔軟に制御できます。パブリケーションにより、優先度による同期を実行できます。つまり、特定のテーブルまたはテーブルのグループだけを同期の対象として指定できます。

- **トランザクションとリカバリ** Ultra Light には、コミット機能、ロールバック機能、デバイスが故障した場合の自動リカバリ機能があるので、トランザクションは完全に実行されるか、またはまったく実行されないことが保証されます。
- **セキュリティ** Ultra Light のユーザ認証とデータベース暗号化、同期中の暗号化によって、安全なアプリケーションを構築できます。
- **パフォーマンスと小さい専有容量** Ultra Light のターゲット・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 が採用するアルゴリズムとデータ構造により、パフォーマンスは高く、メモリ使用量は低く抑えることができます。たとえば、Ultra Light は、小型のデバイス向けに設計されたキャッシング・アルゴリズムを採用しています。
- **マルチスレッド・アプリケーション** マルチスレッド・アプリケーションをサポートするプラットフォームでは、Ultra Light はマルチスレッド・アプリケーションと、単一データベースに接続する複数のアプリケーションをサポートします。

Ultra Light データベースの使用については、『Ultra Light データベース・ユーザーズ・ガイド』>「Ultra Light データベース」を参照してください。

Ultra Light アプリケーションのアーキテクチャ

すべての Ultra Light アプリケーションは、同じ基本データベース管理コード上に構築されます。Ultra Light コンポーネントの場合、このコードはコンポーネント自体に含まれます。一方、静的インタフェースの場合、このコードは個々の Ultra Light ランタイム・ライブラリとして使用されます。



Ultra Light アプリケーションの構成は、次のとおりです。

- アプリケーション・コード
- Ultra Light コンポーネントまたはランタイム・ライブラリ
- Ultra Light データベースまたはスキーマ・ファイル

Ultra Light のマニュアルの使用

Ultra Light プログラミング・インタフェースを選択したら、次のマニュアルから必要な情報を見つけることができます。これらすべてのマニュアルは、SQL Anywhere オンライン・マニュアルに含まれています。

プログラミング・インタフェースの選択に関するヒントについては、「Ultra Light プログラミング・インタフェースの選択」12 ページを参照してください。

- ***Ultra Light データベース・ユーザーズ・ガイド*** **(このマニュアル)** このマニュアルには、Ultra Light データベース管理、SQL、同期などの情報を含む、すべての Ultra Light インタフェースで役に立つ情報が記載されています。

- **インタフェースのマニュアル** 各 Ultra Light インタフェースには個別のマニュアルがあります。これらのマニュアルには、そのインタフェースを使用してアプリケーションを開発するために必要なすべての情報が記載されています。
  - 『Native Ultra Light for Java ユーザーズ・ガイド』>「Native Ultra Light for Java ユーザーズ・ガイド」
  - 『Ultra Light ActiveX ユーザーズ・ガイド』>「Ultra Light ActiveX ユーザーズ・ガイド」
  - 『Ultra Light C/C++ ユーザーズ・ガイド』>「Ultra Light C/C++ ユーザーズ・ガイド」
  - 『Ultra Light for MobileVB ユーザーズ・ガイド』>「Ultra Light for MobileVB ユーザーズ・ガイド」
  - 『Ultra Light 静的型 Java ユーザーズ・ガイド』>「Ultra Light 静的型 Java ユーザーズ・ガイド」
  - 『Ultra Light.NET ユーザーズ・ガイド』>「Ultra Light.NET ユーザーズ・ガイド」
  - 『UltraLite for M-Business Anywhere User's Guide』>「UltraLite for M-Business Anywhere User's Guide」
- **Mobile Link のマニュアル** アプリケーションに同期が含まれる場合、同期システムに関する完全なマニュアルについては、『Mobile Link 管理ガイド』>「Mobile Link 管理ガイド」と『Mobile Link クライアント』>「Mobile Link クライアント」を参照してください。

Ultra Light プログラミング・インタフェースの選択

どの Ultra Light プログラミング・インタフェースを使用するかの選択は、基本的に次の質問に対する回答によって決まります。

- ターゲット・プラットフォームは ?
- 使用するプログラミング言語は ?

C/C++ 開発者および Java 開発者に対して複数のインタフェースを提供できるため、柔軟性が増します。

各インタフェースについては、個々のマニュアルで説明します。詳細については、「Ultra Light のマニュアルの使用」10 ページを参照してください。

Palm OS、Windows XP、Windows CE のクロス・プラットフォーム開発

オプションは次のとおりです。

- **C/C++** 以下から選択できます。
  - Ultra Light C++ コンポーネント
  - 静的型 C++ API
  - Embedded SQL (静的インタフェース)

  これらのインタフェースの選択方法については、「コンポーネントと静的インタフェースの選択」14 ページを参照してください。
- **Visual Basic .NET と Visual Basic** Ultra Light for MobileVB を AppForge MobileVB または AppForge Crossfire と共に使用して、Microsoft 開発環境からクロスプラットフォーム・アプリケーションを開発できます。
- **Web 開発** Ultra Light for M-Business Anywhere を使用して、クロス・プラットフォームの Web アプリケーションを開発できます。

Palm OS のみに対応した開発

オプションは次のとおりです。

- **C/C++** 以下から選択できます。
  - Ultra Light C++ コンポーネント

- 静的型 C++ API
- Embedded SQL (静的インタフェース)

これらのインタフェースの選択方法については、「コンポーネントと静的インタフェースの選択」14 ページを参照してください。

- **Visual Basic** Ultra Light for MobileVB を使用して、Palm OS に対応した Visual Basic アプリケーションを開発できます。
- **Web 開発** Ultra Light for M-Business Anywhere を使用して、Palm OS に対応したブラウザベースのアプリケーションを開発できます。

Windows CE と Windows XP での開発

オプションは次のとおりです。

, **C/C++** 以下から選択できます。

- Ultra Light C++ コンポーネント
- ODBC
- 静的型 C++ API
- Embedded SQL (静的インタフェース)

これらのインタフェースの選択方法については、「コンポーネントと静的インタフェースの選択」14 ページを参照してください。

- **Java** 以下から選択できます。
  - Native Ultra Light for Java (コンポーネント)
  - 静的型 Java API

  これらのインタフェースの選択方法については、「コンポーネントと静的インタフェースの選択」14 ページを参照してください。
- **Visual Basic** 以下から選択できます。
  - Visual Basic .NET からのインタフェースを提供する Ultra Light.NET

- Visual Basic からのインタフェースを提供する Ultra Light for MobileVB (AppForge MobileVB の拡張子を使用)
- eMbedded Visual Basic からのインタフェースを提供する Ultra Light ActiveX

- **C#** Ultra Light.NET を使用して、Windows CE または Windows XP に対応した C# アプリケーションを開発できます。
- **Web 開発** 以下を使用して JavaScript アプリケーションを構築できます。
  - Ultra Light for M-Business Anywhere
  - Ultra Light ActiveX

他のプラットフォームに対応した開発

Ultra Light 静的型 Java は、JDK 1.1.8 以降をサポートするプラットフォームに対して pure Java ソリューションを提供します。

コンポーネントと静的インタフェースの選択

Ultra Light アプリケーションは、Ultra Light 「**コンポーネント**」または「**静的インタフェース**」を使用して構築されます。

どちらを使用するかは、ある程度までは使用する言語によって決まります。C#、Visual Basic、または JavaScript のプログラマである場合、Ultra Light コンポーネントを選択してください。C/C++ または Java のプログラマである場合、コンポーネントと静的インタフェースのどちらかを選択できます。この項では、コンポーネントと静的インタフェースを比較します。

データ・アクセス機能

Ultra Light コンポーネントは、動的 SQL とテーブル・ベース API のどちらかを使用してデータにアクセスできます。

- 動的 SQL には、マルチテーブルのジョインを含む、多くの一般的な SQL 機能が用意されています。静的インタフェースとは対照的に、動的 SQL では、SQL 文をランタイム時に作成できます。

  詳細については、「動的 SQL」201 ページを参照してください。

- Ultra Light コンポーネントのテーブル・ベース API は、一度に 1 つずつローにアクセスします。これは単純な論理ですが、テーブルのジョインなどの論理が必要な場合は、このような論理を独自にアプリケーションに実装する必要があります。

動的 SQL とテーブル・ベース API は単一アプリケーションで組み合わせることができます。

静的インタフェース (Embedded SQL、静的型 C++ API、静的型 Java API) は、SQL を使用してデータにアクセスします。静的インタフェースは動的 SQL より広範囲の SQL をサポートしていますが、すべてのクエリはコンパイル時に指定します。たとえば、UNION と FULL OUTER JOIN クエリは現在静的インタフェースでのみサポートされています。静的 SQL と動的 SQL の両方で、クエリはパラメータを使用でき、これらのパラメータに対してランタイム時に値を指定できます。

また、静的型 C++ API にはテーブル・ベース API も用意されており、これには、コンポーネントのテーブル・ベース API の利点と制約があります。詳細については、『Ultra Light C/C++ ユーザーズ・ガイド』>「静的型 C++ API を使用したアプリケーションの開発」を参照してください。

アプリケーション・サイズ

Ultra Light コンポーネントには、任意のクエリを解析、最適化、実行するコードが含まれています。これに対し、静的インタフェースは、特定のクエリを実行するコードを生成しますが、クエリを解析または最適化するコードを生成する必要はありません。このため、通常、静的インタフェースを使用して構築されたアプリケーションは、Ultra Light コンポーネントを使用して構築されたアプリケーションよりもサイズが小さくなります。

データベース内のクエリとテーブルの数が増えると、静的インタフェースのサイズの利点は失われます。多くのクエリが使用され、多くのテーブルが含まれるデータベースに対応する複雑なアプリケーションの場合、コンポーネントは個々のクエリごとにコードを使用する必要がないため、コンポーネントの方が小さくなる場合があります。

開発モデル

各 Ultra Light コンポーネントには、このコンポーネントによってサポートされている言語のユーザが使いやすいよう設計されたオブジェクト指向の API が採用されています。この開発モデルは、他の多くの種類のアプリケーションと同じです。

Ultra Light 静的インタフェースには、より複雑な開発モデルが必要です。この場合、前処理手順によって、リファレンス・データベースからアプリケーション・コードが生成されます。

多くのユーザにとって、静的インタフェースよりも Ultra Light コンポーネントの方が習得が楽です。

詳細については、次の項を参照してください。

- 「Ultra Light コンポーネントを使用したアプリケーションの開発」17 ページ
- 『Ultra Light C/C++ ユーザーズ・ガイド』>「静的型 C++ アプリケーションの開発」
- 『Ultra Light C/C++ ユーザーズ・ガイド』>「Embedded SQL アプリケーションの開発」
- 『Ultra Light 静的型 Java ユーザーズ・ガイド』>「静的型 Java API を使用したデータ・アクセス」

パフォーマンス

静的インタフェースを使用するアプリケーションの一部として含まれているクエリは、すでに解析および最適化されています。このため、これらのクエリの方が Ultra Light コンポーネントのクエリよりパフォーマンスがよい場合があります。静的インタフェースのクエリの最適化は、リファレンス・データベースにおけるデータの分散状況によって決まります。リファレンス・データベースのデータが Ultra Light データベースのデータに近いほど、パフォーマンスは向上します。

Adaptive Server Anywhere SQL のサポートの詳細については、『ASA SQL リファレンス・マニュアル』>「SQL 文」を参照してください。

Adaptive Server Anywhere との互換性

Embedded SQL には、Ultra Light データベースと Adaptive Server Anywhere データベースに共通の静的プログラミング・インタフェースが用意されています。ADO.NET と ODBC には、Ultra Light コンポーネントと Adaptive Server Anywhere 間で共有されるプログラミング・モデルが用意されています。

特に、両方のデータベースを使用できる Windows CE 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 から完全な機能を備えたより強力な Adaptive Server Anywhere データベースに移行する必要がある場合、Embedded SQL、ODBC、または ADO.NET を使用すると、アプリケーションの移行作業がより簡単になります。

Ultra Light コンポーネントを使用したアプリケーションの開発

❖ Ultra Light コンポーネント・アプリケーションを開発するには、次の手順に従います。

1　データベース・スキーマ・ファイルを設計します。

Ultra Light スキーマ・ペインタを使用するか XML ファイルを記述して、データベース・スキーマを作成します。
SQL Anywhere Studio ユーザは、Adaptive Server Anywhere データベースから Ultra Light データベース・スキーマを生成できます。

詳細については、「Ultra Light スキーマ・ペインタ」158 ページ、「ulxml ユーティリティ」161 ページ、「ulinit ユーティリティ」143 ページを参照してください。

2　Ultra Light プロジェクトの開発環境を設定します。

詳細については、次の項目を参照してください。

- Ultra Light for MobileVB：『Ultra Light for MobileVB ユーザーズ・ガイド』>「レッスン 1：プロジェクト・アーキテクチャの作成」
- Ultra Light.NET：『Ultra Light.NET ユーザーズ・ガイド』>「レッスン 1：Visual Studio プロジェクトの作成」

- Ultra Light C++ コンポーネント：『Ultra Light C/C++ ユーザーズ・ガイド』> 「チュートリアル：C++ コンポーネントを使用したアプリケーションの構築」
- Ultra Light ActiveX：『Ultra Light ActiveX ユーザーズ・ガイド』> 「eMbedded Visual Basic 設計環境への Ultra Light ActiveX の追加」
- Native Ultra Light for Java：『Native Ultra Light for Java ユーザーズ・ガイド』> 「Borland JBuilder でのアプリケーション開発」
- Ultra Light for M-Business Anywhere：『UltraLite for M-Business Anywhere User's Guide』> 「UltraLite for M-Business Anywhere Quick Start」

3 Ultra Light API を使用して特定の言語に対応したアプリケーション・コードを記述します。

4 アプリケーションとデータベース・スキーマを配備します。

この手順は、ターゲット・デバイスと、使用しているコンポーネントによって異なります。

第 2 章

チュートリアル：CustDB サンプル Ultra Light アプリケーション

この章の内容

この章では、CustDB サンプル・アプリケーションを使用して Ultra Light の主な機能について説明します。

これらの技術については、CustDB のデスクトップ・バージョンを使用して説明します。また、サポートされているインタフェースごとに、CustDB のバージョンを用意しています。詳細については、「次の手順」33 ページを参照してください。

この章の内容は、マニュアル全体を通して例として取り上げられています。

概要

この章では、CustDB (顧客データベース) Ultra Light サンプル・アプリケーションについて説明します。CustDB は、販売管理アプリケーションです。

CustDB サンプル・アプリケーションを使用して、Ultra Light アプリケーションを開発するときに必要な、多くの方法の実行例を紹介します。

この章では、Windows NT/2000/XP 用にコンパイルされたアプリケーションを使用します。

各 Ultra Light プログラミング・インタフェースには、CustDB アプリケーションの個別バージョンを用意しています。各バージョンには同様の機能がありますが、場合によっては、各プラットフォームの規則に応じて機能が変更されていることがあります。

Ultra Light の機能

この章では、次の Ultra Light の機能について説明します。

- Ultra Light データベース・アプリケーションは、使用できるリソースが限られた小型のデバイスで実行できます。
- Ultra Light アプリケーションには、リレーショナル・データベース・エンジンも含まれます。
- Ultra Light アプリケーションでは、双方向同期スキームにより、中央の統合データベースとデータを共有します。Ultra Light データベースは、「**リモート・データベース**」とも呼ばれます。
- 各リモート・データベースには、統合データベースに格納されているデータのサブセットが含まれます。
- SQL Anywhere Studio に付属の Mobile Link 同期サーバは、統合データベースとインストールされた各 Ultra Light アプリケーションとの間でデータの同期を行います。
- 統合データベースに格納される SQL スクリプトは、同期論理を実装します。
- 同期スクリプトの参照と編集には、Sybase Central を使用します。

シナリオ

CustDB のシナリオは、次のようになります。

統合データベースは本社に設置されています。

リモート・データベースには、販売担当者用とモバイル・マネージャ用の 2 つのタイプがあります。各販売担当者の Ultra Light リモート・データベースには、すべての製品と、その販売担当者に割り当てられている注文のみが含まれています。モバイル・マネージャの Ultra Light リモート・データベースには、すべての製品と注文が含まれています。

サンプル・アプリケーションで実行される内容は次のとおりです。

- 顧客および製品のリスト表示
- 新規顧客の追加
- 注文の追加や削除
- 未処理の注文リストのスクロール
- 注文の承認や拒否
- 変更内容に関する統合データベースとの同期

CustDB Ultra Light アプリケーションを実行して 1 つのリモート・データベースを操作すると、その変更内容は統合データベースと同期されます。

Ultra Light を一般的な方法でインストールすると、複数のリモート・データベースがインストールされます。各リモート・データベースはハンドヘルド・デバイス上で動作し、統合データベースからのデータのサブセットを含みます。

ファイル・ロケーション

CustDB サンプル・アプリケーションは、インストールされている Ultra Light 内の次のロケーションに格納されています。

ランタイム・ファイルのロケーション

CustDB サンプル・アプリケーションを実行するには、次のコンポーネントが必要です。

- **統合データベース** CustDB データベースの Adaptive Server Anywhere バージョンは、*Samples¥UltraLite¥Custdb¥custdb.db* に格納されています。スキーマの詳細と、サポートされている他の統合データベース・タイプの使用方法については、『Mobile Link チュートリアル』> 「CustDB サンプル・アプリケーション」を参照してください。

  このデータベースは、統合データベースとして実行されます。これに含まれる情報は次のとおりです。

  - 同期されるメタデータを格納する Mobile Link システム・テーブル
  - ベース・テーブルのローに格納される CustDB データ
  - 同期スクリプト

  インストール中、このデータベース用に UltraLite 9.0 Sample という名前の ODBC データ・ソースが作成されます。

- **Ultra Light アプリケーション実行プログラムとソース・コード** サンプルは、インタフェースごとのバージョンが用意されています。

  実行プログラムとソース・コードは、SQL Anywhere Studio のインストール環境の次のサブディレクトリにあります。

コンポーネント	場所
Ultra Light for MobileVB	*Samples¥UltraLiteForMobileVB* と *Samples¥UltraLiteForCrossfire*
Ultra Light ActiveX	*Samples¥UltraLiteActiveX*
Ultra Light.NET	*Samples¥UltraLite.NET*
Native Ultra Light for Java	*Samples¥NativeUltraLiteForJava*
C++ コンポーネント	*Samples¥UltraLite*
Embedded SQL	*Samples¥UltraLite*
静的型 C++ API	*Samples¥UltraLite*

コンポーネント	場所
静的型 Java API	*Samples¥UltraLite*
Ultra Light for M-Business Anywhere	*Samples¥UltraLiteForMBusinessAnywhere*

[スタート] メニューからサンプルをブラウズするには、[プログラム] ー [SQL Anywhere 9] ー [サンプル・アプリケーションおよびプロジェクト] を選択します。

サンプル・アプリケーションにおける同期テクニック

CustDB サンプル・データベースとの同期は、デスクトップ・コンピュータ上で実行する Mobile Link 同期サーバによって実行されます。

CustDB サンプル・アプリケーションでは、便利な同期テクニックを紹介しています。同期の実行方法の詳細については、Mobile Link のマニュアルの次の項を参照してください。

- 同期処理の概要については、『Mobile Link 管理ガイド』> 「同期処理」を参照してください。
- 同期を制御する同期スクリプトの記述方法については、『Mobile Link 管理ガイド』> 「同期スクリプトの作成」を参照してください。
- CustDB サンプル・アプリケーションで使用する同期テクニックについては、『Mobile Link チュートリアル』> 「CustDB サンプル・アプリケーション」を参照してください。

レッスン 1：Mobile Link 同期サーバの起動

Ultra Light のサンプル・アプリケーションを初めて起動したときには、リモート データベースにデータは含まれていません。アプリケーションは同期を実行し、統合データベースからデータの初期コピーをダウンロードします。

同期を実行するには、データベース・サーバを起動し、Ultra Light サンプル・データベースに対して実行される Mobile Link 同期サーバを起動します。

次の手順では、SQL Anywhere Studio によって [スタート] メニューに追加されているいくつかのショートカットを使用します。

❖ Mobile Link 同期サーバを起動するには、次の手順に従います。

1. Adaptive Server Anywhere CustDB サンプル・データベースを起動します。

   [スタート] メニューから、[プログラム] － [SQL Anywhere 9] － [Ultra Light] － [Ultra Light のパーソナル・サーバのサンプル] を選択します。

   このチュートリアルでは、CustDB サンプル・データベースは統合データベースです。Ultra Light データベースとこの統合データベース間でデータを同期します。また、Adaptive Server Anywhere 以外のデータベース・サーバも統合データベースとして使用できます。詳細については、『Mobile Link チュートリアル』> 「CustDB サンプル・アプリケーション」を参照してください。

2. Mobile Link 同期サーバを CustDB データベースに対して起動します。

   [スタート] メニューから、[プログラム] － [SQL Anywhere 9] － [Mobile Link] － [同期サーバのサンプル] を選択します。

Mobile Link 同期サーバは、ODBC ドライバを使用して統合データベース・サーバに接続します。データベース・サーバとは別のマシンで実行することもできますが、このチュートリアルでは、同じマシンから実行します。

レッスン 2：サンプル・アプリケーションの起動と同期

次の手順では、サンプル・アプリケーションを起動し、統合データベースとの同期を行って、データの初期セットを取得します。ダウンロードするデータは、アプリケーションの起動時に入力したユーザ ID によって異なります。

❖ 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. サンプル・アプリケーションを起動します。

   [スタート] メニューから、[プログラム] － [SQL Anywhere 9] － [Ultra Light] － [Windows アプリケーションのサンプル] を選択します。

   ここでは、サンプル・アプリケーションは Mobile Link 同期サーバと同じデスクトップ・マシン上で動作しています。運用環境では、Ultra Light アプリケーションは多くの場合、ハンドヘルド・デバイス上で動作します。

2. 従業員 ID を入力します。

   値として 50 を入力し、[Enter] キーを押します。このアプリケーションでは 51、52、または 53 の値も入力できますが、これらの値を入力すると、動作が多少異なります。

   従業員 ID を入力すると、アプリケーションが同期され、顧客、製品、注文情報がアプリケーションにダウンロードされます。Mobile Link 同期サーバのウィンドウに、同期が行われたことを示すメッセージが表示されます。

3. アプリケーションにデータが含まれていることを確認します。

   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ています。

レッスン 3：注文情報の追加

この項では、新規注文をデータベースに追加します。注文の追加は、どのアプリケーション・バージョンでも同様に行います。

データベースはテーブルを保持しており、テーブルの各ローには、特定の注文に関する情報が収められています。それぞれの注文情報には、顧客、製品、数量、価格、必要に応じて値引きに関するデータが含まれています。また、アプリケーションから修正できる [status] フィールドと [notes] フィールドも含まれています。

このリモート・データベースは、統合データベースの ULOrder テーブルにリストされた注文は受け取りません。承認されていない注文のみがダウンロードされます。同期スクリプトを使ってどの情報をアプリケーションに送信するかを指定できます。

注文情報の追加

❖ 注文情報を追加するには、次の手順に従います。

1　未処理の注文情報をスクロールします。

[Next] をクリックして次の顧客を表示します。

2　新規注文を入力します。

[Order] メニューから [New] を選択します。

[Add New Order] 画面が表示されます。

3　顧客を選択します。

ドロップダウン・リストから [**Basements R Us**] を選択します。このリストには、統合データベースから取得した顧客の完全なリストが表示されます。

現在の注文リストには、この顧客からの注文はありません。

4　製品を選択します。

Ultra Light には、統合データベースから取得した製品の完全なリストが保管されています。このリストを表示するには、[Product] ドロップダウン・リスト・ボックスを開きます。

リストから [**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。

5 数量と値引き情報を入力します。

数量に 20、値引きに 5 を入力します。

6 [Enter] キーを押して、新しい注文を追加します。

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。

レッスン 4：既存の注文の承認または拒否

この手順では、1 つの注文を承認し、別の注文を拒否します。注文の承認や拒否によって、ローカル・データベースのカラムが 2 つ更新されます。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。

❖ 注文の承認、拒否、削除を行うには、次の手順に従います。

1 Apple Street Builders からの注文を承認します。

- リストの最初にある Apple Street Builders からの注文を表示します。
- [Approve] をクリックして、注文を承認します。
- 承認したことを示す Good Work というメモを追加します。
- 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。

- 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。
- [Deny] をクリックして、注文を拒否します。
- Discount too high というメモを追加します。

3 Awnings R Us からの注文を削除します。

- 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。
- メニュー項目の [Order] － [Delete] を選択して、この注文を削除します。データのローカル・コピーからこの注文情報が消えます。

レッスン 5：変更内容の同期

この項では、リモート・データベースで変更した内容を統合データベースと同期させます。

同期するには、Mobile Link 同期サーバが実行されている必要があります。Mobile Link 同期サーバを停止した場合は、「レッスン 1：Mobile Link 同期サーバの起動」24 ページの説明に従ってサーバを再起動してください。

❖ 変更内容を同期するには、次の手順に従います。

1. [File] － [Synchronize] 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。

   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。Apple Street Builders の承認された注文情報がアプリケーションに残っていないことを確認します。

統合データベースでの同期の確認

この項では、Interactive SQL を統合データベースに接続し、変更が同期されていることを確認します。

❖ 統合データベースで同期を確認するには、次の手順に従います。

1. Interactive SQL から統合データベースに接続します。
   - [スタート] － [プログラム] － [SQL Anywhere 9] － [Adaptive Server Anywhere] － [Interactive SQL] を選択します。

     Interactive SQL の [接続] ダイアログが表示されます。

- [ODBC データ・ソース名] を選択し、ドロップダウン・リストから [UltraLite 9.0 Sample] を選択します。

2　承認や拒否を行った注文情報のステータスが変更されていることを確認します。

承認や拒否が同期されたことを確認するため、次の文を発行します。

```
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がなくなっていることを確認します。

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 6：統合データベースのブラウズ

Sybase Central を使用して、Mobile Link 同期を管理できます。同期論理は統合データベースに格納されます。

この項では、Sybase Central を使用して CustDB 統合データベースのスクリプトをブラウズする方法について説明します。

Sybase Central から CustDB データベースへの接続

1. CustDB データベースを起動します。
   - [プログラム] － [SQL Anywhere 9] － [Ultra Light] － [Ultra Light のパーソナル・サーバのサンプル] を選択します。
2. Sybase Central を起動します。
   - [スタート] メニューから、[プログラム] － [SQL Anywhere 9] － [Sybase Central] を選択します。
3. サンプル・データベースに接続します。
   - Sybase Central で、[ツール] － [接続] を選択します。接続タイプが複数ある場合は Mobile Link を選択します。

     [Mobile Link 接続] ダイアログが表示されます。
   - [ODBC] を選択し、[データ・ソース] ボックスに UltraLite 9.0 Sample と入力します。[OK] をクリックして接続します。

同期スクリプトのブラウズ

Sybase Central では、統合データベースに保管されているテーブル、ユーザ、同期されたテーブル、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。

❖ 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　[接続スクリプト] フォルダを開きます。

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 同期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。

2　[同期テーブル] フォルダを開き、[ULCustomer] テーブル・フォルダを開きます。

右ウィンドウ枠には、このテーブル固有の 1 組のスクリプトと、それに対応するイベントがリストされています。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。

同期スクリプトの内容については、『Mobile Link 管理ガイド』>「同期スクリプトの作成」と『Mobile Link チュートリアル』>「CustDB サンプル・アプリケーション」を参照してください。

次の手順

CustDB アプリケーションに加えて、サポートされているインタフェースごとにチュートリアルが用意されています。詳細については、次の項を参照してください。

- **Native Ultra Light for Java**『Native Ultra Light for Java ユーザーズ・ガイド』>「チュートリアル：CustDB サンプル・アプリケーション」
- **Ultra Light ActiveX**『Ultra Light ActiveX ユーザーズ・ガイド』>「チュートリアル：Ultra Light ActiveX アプリケーションのサンプル」または『Ultra Light ActiveX ユーザーズ・ガイド』>「チュートリアル：Pocket IE 用の Ultra Light ActiveX アプリケーション」

- **Ultra Light C++ コンポーネント**『Ultra Light C/C++ ユーザーズ・ガイド』>「チュートリアル：C++ コンポーネントを使用したアプリケーションの構築」
- **Ultra Light Embedded SQL**『Ultra Light C/C++ ユーザーズ・ガイド』>「チュートリアル：Embedded SQL を使用したアプリケーションの構築」
- **Ultra Light for MobileVB**『Ultra Light for MobileVB ユーザーズ・ガイド』>「チュートリアル：Ultra Light for MobileVB アプリケーションのサンプル」
- **Ultra Light.NET**『Ultra Light.NET ユーザーズ・ガイド』>「チュートリアル：Ultra Light.NET アプリケーションの構築」
- **Ultra Light for M-Business Anywhere**『UltraLite for M-Business Anywhere User's Guide』>「UltraLite for M-Business Anywhere Quick Start」

第 3 章

Ultra Light データベース

この章の内容

この章では、Ultra Light データベースのデータ記憶領域、ユーザ認証、文字セットの問題に関する基本情報について説明します。

Ultra Light データベースとスキーマの作成

Ultra Light データベースは、単一ファイル (Palm OS の場合は Palm 継続保管) に格納されます。データベース・ファイルには、テーブルやインデックスのほか、同期に必要な追加情報も含まれます。.

すべてのデータベースには、「**スキーマ**」があります。これは、データベースを構成するテーブルやインデックスなどに関する情報です。このメタデータには、カラム名、データ型、プライマリおよび外部キー定義などが含まれます。ほとんどのリレーショナル・データベースでは、「**システム・テーブル**」または「**カタログ**」と呼ばれる特別なテーブルのセットにスキーマが格納されます。Ultra Light では、スキーマはよりコンパクトな形で格納されます。

Ultra Light スキーマの作成

Ultra Light データベースを作成するには、データベース自体とは個別に Ultra Light スキーマを外部的に作成します。

Ultra Light コンポーネントを使用してアプリケーションを開発している場合、スキーマ・ファイルにスキーマを作成します。次に、スキーマを Ultra Light データベース・ファイルに適用します。

静的インタフェースを使用してアプリケーションを開発している場合、スキーマをリファレンス・データベースに作成します。リファレンス・データベースからアプリケーション・コードを生成すると、スキーマはアプリケーション自体に追加されます。

詳細については、「Ultra Light スキーマ・ファイルの作成」37 ページを参照してください。

データベースへのスキーマの適用

一般的には、Ultra Light データベース・ファイルとともにアプリケーションを配備することはなく、Ultra Light がデータベース・ファイルを作成し、同期を使用してデータベースに適したデータを入力するようにします。

Ultra Light コンポーネントのアプリケーションは、データベース・スキーマ・ファイルを配備することを必要とします。初回の接続試行でデータベース・ファイルを作成し、スキーマ・ファイルをこのデータベース・ファイルに適用するようにアプリケーションを作成できます。プログラミング・インタフェースごとに、サンプル・コードがチュートリアルに用意されています。

Ultra Light 静的インタフェース・アプリケーションは、生成されたコードにデータベース・スキーマ定義を含みます。これらはデータベースを自動的に作成し、初回の接続試行時にスキーマを適用します。

詳細については、「Ultra Light データベースの作成」39 ページを参照してください。

データベース・スキーマの変更

Ultra Light データベースのスキーマは、新しいスキーマをデータベース・ファイルに適用して変更できます。元のスキーマと同様に、新しいスキーマはスキーマ・ファイル (Ultra Light コンポーネント) か、アプリケーションの新しいバージョン (静的インタフェース) に格納されることがあります。

Ultra Light コンポーネントを使用している場合、テーブルとインデックス (データ定義文) を変更する SQL 文を実行して、Ultra Light データベースのスキーマを変更することもできます。

詳細については、「Ultra Light データベース・スキーマのアップグレード」71 ページを参照してください。

参照

Ultra Light データベースの概要については、「Ultra Light データベース機能」7 ページを参照してください。

Ultra Light スキーマ・ファイルの作成

Ultra Light スキーマ・ファイルは、Ultra Light コンポーネントとともに使用されます。次の方法で、Ultra Light スキーマ・ファイルを作成できます。

- **Ultra Light スキーマ・ペインタ** スキーマ・ファイルを作成する最も簡単な方法は、Ultra Light スキーマ・ペインタを使用する方法です。

  スキーマ・ペインタを起動するには、[スタート] ー [プログラム] ー [SQL Anywhere 9] ー [Ultra Light] ー [Ultra Light Schema Painter] を選択するか、Windows エクスプローラでスキーマ・ファイル (拡張子 *.usm*) をダブルクリックします。詳細については、「Ultra Light スキーマ・ペインタ」158 ページを参照してください。

- **Adaptive Server Anywhere データベースからのスキーマ生成** Adaptive Server Anywhere データベースとともに Ultra Light データベースに必要なスキーマ、またはこのスキーマのスーパーセットがある場合は、ulinit コマンド・ライン・ユーティリティを使用して Ultra Light スキーマ・ファイルを生成できます。

  詳細については、「ulinit ユーティリティ」143 ページを参照してください。

  Adaptive Server Anywhere 移行ウィザードを使用して、他のデータベースから Adaptive Server Anywhere データベースにテーブル定義、インデックス、データを移行できます。このウィザードを使用して、Adaptive Server Enterprise、Oracle、SQL Server、または DB2 データベースから Ultra Light スキーマ定義を作成できます。

  詳細については、『ASA SQL ユーザーズ・ガイド』>「Adaptive Server Anywhere へのデータベースの移行」を参照してください。

- **スキーマ・ファイルへの XML ファイルの変換** ulxml コマンド・ライン・ユーティリティを使用して、XML ファイルをスキーマ・ファイルに変換できます。また、スキーマ・ファイルを XML ファイルに変換することもできます。

  たとえば、ソース制御下でデータベース定義をする場合、このユーティリティが役に立ちます。また、このユーティリティは、自動的に構築されたプロセスとも効率的に統合できます。

  詳細については、「ulxml ユーティリティ」161 ページを参照してください。

- **既存の Ultra Light データベースからのスキーマのアンロード**

  ulconv コマンド・ライン・ユーティリティを使用して、既存のデータベースからのスキーマのアンロードなど、Ultra Light データベースで多くの処理を行うことができます。

  詳細については、「ulconv ユーティリティ」130 ページを参照してください。

Ultra Light スキーマ・ファイルを作成する場合、いくつかのデータベース全体の特性を設定します。これには次のようなものがあります。

- **大文字と小文字の区別** データベースの大文字と小文字の区別は、すべての文字列の比較に影響します。インデックスはソート順で保管されるため、大文字と小文字の区別はデータベースの作成時に指定します。ソート順は、データベースの大文字と小文字が区別されるかどうかによって異なります。
- **文字セット** 各データベースには、照合順 (文字セットとソート順) がはっきりと定義されています。照合順は、データベースの作成時に定義されます。

  詳細については、「Ultra Light の文字セット」57 ページを参照してください。

Ultra Light データベースの作成

Ultra Light の作成方法は、使用する開発モデルによって異なります。

❖ Ultra Light データベース (Ultra Light コンポーネント) を作成するには、次の手順に従います。

1　Ultra Light スキーマ・ファイルを作成します。

詳細については、「Ultra Light スキーマ・ファイルの作成」37 ページを参照してください。

2　データベースが見つからない場合にスキーマ・ファイルを使用するアプリケーションを作成します。

データベースに接続するコードでは次の手順を実行します。

```
Open Connection( database identification parameters
)
 If ( database not found ) Then
     Create Database( database schema parameters )
 End If
```

データベース識別パラメータのリストについては、「データベース識別パラメータ」89 ページを参照してください。データベース・スキーマ・パラメータのリストについては、「データベース・スキーマ・パラメータ」102 ページを参照してください。

仕様は、使用しているコンポーネントによって異なります。詳細については、次の項目を参照してください。

- Ultra Light for MobileVB：『Ultra Light for MobileVB ユーザーズ・ガイド』>「Ultra Light データベースへの接続」
- Ultra Light ActiveX：『Ultra Light ActiveX ユーザーズ・ガイド』>「Ultra Light データベースへの接続」
- Native Ultra Light for Java：『Native Ultra Light for Java ユーザーズ・ガイド』>「データベースへの接続」
- Ultra Light.NET：『Ultra Light.NET ユーザーズ・ガイド』>「データベースへの接続」
- Ultra Light C++ コンポーネント：『Ultra Light C/C++ ユーザーズ・ガイド』>「データベースへの接続」
- Ultra Light for M-Business Anywhere：『UltraLite for M-Business Anywhere User's Guide』>「Connecting to an UltraLite database」

3 アプリケーションから、または同期によって Ultra Light データベースにデータを追加します。

❖ Ultra Light データベース (静的インタフェース) を作成するには、次の手順に従います。

1 Adaptive Server Anywhere の「**リファレンス・データベース**」を作成します。リファレンス・データベースには、Ultra Light データベースと同じテーブルとインデックスが格納されます。

詳細については、「リファレンス・データベースの準備」251 ページを参照してください。

2 Ultra Light アプリケーションをリファレンス・データベースから生成します。

詳細については、「Ultra Light データ・アクセス・コードの生成」263 ページを参照してください。

3　アプリケーションが最初に実行されるときに、リファレンス・データベースの情報を使用して Ultra Light データベース・ファイルが作成されます。

4　同期によって、アプリケーションから Ultra Light データベースにデータを追加します。

Ultra Light データベース・ファイルの作成

Ultra Light アプリケーションがデータベースを作成するにはスキーマ定義が必要です。Ultra Light コンポーネントを使用している場合、データベース・スキーマをそれぞれスキーマ・ファイルに配備します。静的開発モデルを使用している場合、アプリケーションにはスキーマ定義が含まれます。

Ultra Light データベースの物理的な保管方法は、ターゲット・プラットフォームに依存します。

- **Palm Computing Platform** データベースは、Data Manager API を使用する Palm 永続的 (静的) メモリに保管されます。Palm OS バージョン 4.0 で動作しているデバイスでは、拡張カードのファイル・ベース保管領域に Ultra Light データベースを保管できます。

  詳細については、「Palm OS のデータベース保管の選択」240 ページと「DatabaseOnPalm 接続パラメータ」93 ページを参照してください。

- **Windows と Windows CE** データベースはファイル・システムに保管されます。Windows CE のデフォルト・ファイルは *¥UltraLiteDB¥ul.udb* です。その他のバージョンの Windows のデフォルト・ファイルは、アプリケーションの作業ディレクトリにある *ul_<project>.udb* です。ここで、*<project>* は開発プロセス中に使用される Ultra Light プロジェクト名です。

  データベース・ファイル名を明示的に指定することも、デフォルトのファイル名を使用することもできます。

詳細については、「DatabaseOnCE 接続パラメータ」91 ページを参照してください。

- **静的型 Java** データベースは一時的、またはファイル・システムのファイルとして保管されます。デフォルトでは、一時的です。

静的 API の場合、初回接続試行時 (データベースの作成時) にデータベースの暗号化などの機能を制御するパラメータを指定できます。Ultra Light コンポーネントの場合は、データーベース作成のメソッドのスキーマ・パラメータを設定します。

関連項目

- **Ultra Light for MobileVB**『Ultra Light ActiveX ユーザーズ・ガイド』>「CreateDatabaseWithParms メソッド」を参照してください。
- **Ultra Light ActiveX**『Ultra Light ActiveX ユーザーズ・ガイド』>「CreateDatabaseWithParms メソッド」を参照してください。
- **Native Ultra Light for Java**『Native UltraLite for Java API リファレンス』の「**ianywhere.native_ultralite.DatabaseManager**」を参照してください。
- **Ultra Light.NET**『Ultra Light.NET ユーザーズ・ガイド』>「ULDatabaseManager クラス」(iAnywhere.Data.UltraLite ネームスペース) または『Ultra Light.NET ユーザーズ・ガイド』>「DatabaseManager クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。
- **Ultra Light C++ コンポーネント**『Ultra Light C/C++ ユーザーズ・ガイド』>「クラス UltraLite_DatabaseManager」を参照してください。
- **Ultra Light for embedded SQL**『Ultra Light C/C++ ユーザーズ・ガイド』>「Ultra Light C/C++ アプリケーションのマクロとコンパイラ指令」を参照してください。
- **Ultra Light 静的型 C++**『Ultra Light C/C++ ユーザーズ・ガイド』>「Ultra Light C/C++ アプリケーションのマクロとコンパイラ指令」を参照してください。

- **Ultra Light 静的型 Java**『Ultra Light 静的型 Java ユーザーズ・ガイド』> 「Ultra Light API リファレンス」を参照してください。
- **Ultra Light for M-Business Anywhere**『UltraLite for M-Business Anywhere User's Guide』> 「Method createDatabaseWithParms」を参照してください。

Ultra Light データベース・プロパティの設定

データベースを最初に作成するときに、Ultra Light データベースのグローバル特性を設定できます。これは、スキーマに作成するか、最初の接続時に作成します。

- 文字セット

  「Ultra Light スキーマ・ファイルの作成」37 ページを参照してください。

- 大文字と小文字の区別

  「Ultra Light スキーマ・ファイルの作成」37 ページを参照してください。

- データの暗号化

  「Ultra Light データベースの暗号化」49 ページと「Obfuscate 接続パラメータ」108 ページを参照してください。

- データベース・ページ・サイズ

  「Ultra Light データベース・ファイル」62 ページと「PageSize 接続パラメータ」109 ページを参照してください。

- 日付と時刻のフォーマット

  これらは、データベース・オプションとしてスキーマ・ファイルに設定します。「Ultra Light データベース・オプション」45 ページを参照してください。

- 算術処理の精度と位取り

  これらは、データベース・オプションとしてスキーマ・ファイルに設定する必要があります。「Ultra Light データベース・オプション」45 ページを参照してください。

- Ultra Light ランタイムがキャッシュとして使用するメモリ容量

  「CacheSize 接続パラメータ」96 ページを参照してください。

- ファイル・システム領域の事前割り付け

「Reserve Size 接続パラメータ」111 ページを参照してください。

Ultra Light データベース・オプション

Ultra Light データベースは、次のデータベース・オプションのセットをサポートしています。これらはスキーマ・ファイルに設定します。1 つのオプションのセットは、日付と時間の処理を制御します。2 番目のセットは、算術演算のデフォルト処理を制御します。

データベース・オプションの設定

Ultra Light コンポーネントの場合、データベース・オプションはスキーマ・ファイルに設定します。

- スキーマ・ペインタを使用している場合、オプションはデータベース・プロパティ・シートに設定します。
- ulinit ユーティリティを使用している場合、オプションは Adaptive Server Anywhere リファレンス・データベースに設定します。
- ulxml ユーティリティを使用している場合、オプションはデータベース・スキーマが記載された XML ドキュメントに設定します。

Ultra Light 静的インタフェースの場合、オプションは Adaptive Server Anywhere リファレンス・データベースに設定します。

Adaptive Server Anywhere データベース・オプションの設定については、『ASA データベース管理ガイド』>「オプションの設定」を参照してください。

日付と時刻のオプション

次のデータベース・オプションは、日付と時間のデフォルト処理を制御します。これらの設定は、DATEFORMAT などの関数を使用して、SQL オペレーション内で変更できます。

- **DateFormat** 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。

指定可能な値は、このリストに続くテーブルにリストされているシンボルを使用して生成されます。デフォルト値は YYYY-MM-DD です。

該当する Adaptive Server Anywhere データベース・オプションは、DATE_FORMAT です。『ASA データベース管理ガイド』>「DATE_FORMAT オプション [互換性]」を参照してください。

- **DateOrder** データベースに送信されるときの日付のデフォルトの解釈を設定します。

  指定可能な値は、MDY、YMD、DMY です。デフォルト値は YMD です。

  該当する Adaptive Server Anywhere データベース・オプションは、DATE_ORDER です。『ASA データベース管理ガイド』>「DATE_ORDER オプション [互換性]」を参照してください。

- **NearestCentury** データベースに送信されるときの 2 桁の整数から成る年度の値の解釈を設定します。

  指定可能な値は、0 ～ 100 の整数です。デフォルト値は 50 です。この値より小さい 2 桁の年度の値 *YY* は 20*YY* に変換され、この値以上の年度の値は 19*YY* に変換されます。

  該当する Adaptive Server Anywhere データベース・オプションは、NEAREST_CENTURY です。『ASA データベース管理ガイド』>「NEAREST_CENTURY オプション [互換性]」を参照してください。

- **TimeFormat** データベースから取り出される時間のデフォルト・フォーマットを設定します。

  指定可能な値は、このリストに続くテーブルにリストされているシンボルを使用して生成されます。デフォルト値は HH:NN:SS.SSS です。

該当する Adaptive Server Anywhere データベース・オプションは、TIME_FORMAT です。『ASA データベース管理ガイド』>「TIME_FORMAT オプション [互換性]」を参照してください。

- **TimestampFormat** データベースから取り出されるタイムスタンプ値のデフォルト・フォーマットを設定します。

  指定可能な値は、このリストに続くテーブルにリストされているシンボルを使用して生成されます。デフォルト値は YYYY-MM-DD HH:NN:ss.SSS です。

  該当する Adaptive Server Anywhere データベース・オプションは、TIMESTAMP_FORMAT です。『ASA データベース管理ガイド』>「TIMESTAMP_FORMAT オプション [互換性]」を参照してください。

- **TimestampIncrement** 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。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、この値が役に立ちます。特に、同期時にタイムスタンプが一致することが重要になります。また、サポートされている統合データベースが異なる場合、タイムスタンプの解決方法が異なります。TimestampIncrement が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。

  指定可能な値は、1 以上の整数です。デフォルト値は 1 です。

  該当する Adaptive Server Anywhere データベース・オプションは、TRUNCATE_TIMESTAMP_VALUES です。『ASA データベース管理ガイド』>「TRUNCATE_TIMESTAMP_VALUES オプション [データベース]」を参照してください。

DateFormat、TimeFormat、TimestampFormat 値には、次の表のシンボルが使用されます。

シンボル	説明
yy	2 桁の年度

シンボル	説明
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 -"m" の数と同数の文字。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。先頭のゼロは必要ありません。
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。先頭のゼロは必要ありません。
nn	2 桁の分。先頭のゼロは必要ありません。
ss[.*ss*..]	秒と秒の一部
aa	AM または PM (12 時間時計)
pp	PM 表記による午後の時間 (12 時間時計)
jjj	指定年の日 (1 ～ 366)

算術処理

Precision と Scale オプションは、算術処理を制御します。

- **Precision** 10 進数算術計算の結果の最大桁数を設定します。

  指定可能な値は、0 ～ 127 の整数です。デフォルト値は 30 です。

  精度は、小数点の左右の合計桁数です。Scale オプションは、算術計算結果が Precision の最大値にトランケートされるときの小数点以下の最小桁数を指定します。

- **Scale** 算術計算結果が Precision の最大値にトランケートされるときの小数点以下の最小桁数を指定します。

  指定可能な値は、0 ～ 127 の整数です。デフォルト値は 6 です。

たとえば、DECIMAL(8,2) 値に DECIMAL(9,2) 値を掛ける場合、結果には DECIMAL(17,4) が必要になることがあります。Precision が 15 の場合、結果は 15 桁のみが保持されます。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 データベースの暗号化

デフォルトでは、Ultra Light データベースは、ディスク上でも永続的なメモリ内でも暗号化されません。データベース・ファイル内のテキスト・カラムとバイナリ・カラムは、16 進エディタなどの表示ツールを使用すると、きちんと読むことができます。セキュリティを強化するために、次の 2 つのオプションが用意されています。

- **難読化**　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、高度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。
- **高度な暗号化**　Ultra Light データベース・ファイルは、AES 128 ビット・アルゴリズムを使用して高度に暗号化できます。このアルゴリズムは、Adaptive Server 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。高度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。

警告

高度に暗号化されているデータベースの暗号化キーをなくしたり忘れてしまったりした場合は、データベースにアクセスできません。このような状況では、テクニカル・サポートでもデータベースに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。

Ultra Light データベースの暗号化

Ultra Light データベースを暗号化するには、データベース・ファイルの作成時に暗号化キーを指定します。指定したキーを使用することでデータベースが暗号化されます。以降の接続では、指定したキーと暗号化キーが照合され、一致しない場合は接続が失敗します。

暗号化キーの変更

各インタフェースには、データベースの暗号化キーを変更する関数が用意されています。既存のキー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から、変更を行ってください。

警告

キーを変更すると、データベースのすべてのローは古いキーを使用して復号化され、新しいキーを使用して再度暗号化されます。この操作は回復不能です。操作の途中でアプリケーションが中断されると、データベースは無効になり、アクセスできなくなります。新しいデータベースを作成してください。

関連項目

- **Ultra Light for MobileVB**『Ultra Light for MobileVB ユーザーズ・ガイド』>「暗号化と難読化」を参照してください。
- **Ultra Light ActiveX**『Ultra Light ActiveX ユーザーズ・ガイド』>「暗号化と難読化」を参照してください。
- **Native Ultra Light for Java**『API リファレンス』の「**ianywhere.native_ultralite.ConnectionParms**」を参照してください。
- **Ultra Light.NET**『Ultra Light.NET ユーザーズ・ガイド』>「ULConnectionParms クラス」(iAnywhere.Data.UltraLite ネームスペース) または 『Ultra Light.NET ユーザーズ・ガイド』>「ConnectionParms クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。
- **Ultra Light C++ コンポーネント**『Ultra Light C/C++ ユーザーズ・ガイド』>「クラス UltraLite_Connection_iface」を参照してください。
- **Embedded SQL と静的型 C++ API**『Ultra Light C/C++ ユーザーズ・ガイド』>「データの暗号化」を参照してください。

- **Ultra Light 静的型 Java**『Ultra Light 静的型 Java ユーザーズ・ガイド』>「Ultra Light データベースの暗号化」を参照してください。
- **Ultra Light for M-Business Anywhere**『UltraLite for M-Business Anywhere User's Guide』>「Encryption and obfuscation」を参照してください。

Palm OS に関する検討事項

Palm Computing Platform 上で Ultra Light データベースを暗号化する場合は、アプリケーションを起動するたびにキーの再入力を要求するプロンプトが表示されます。この項では、キーの再入力を回避するコードを追加する方法について説明します。

暗号化キーは、Palm の「**機能**」として動的メモリに保存できます。すると、アプリケーションは、起動時に、ユーザにキー入力を要求するのではなく、メモリからキーを取り出します。機能には、作成者と機能番号のインデックスが付けられます。ユーザは、各自の作成者 ID または NULL を、機能番号または NULL と一緒に渡して、暗号化キーを保存したり取り出したりできます。

暗号化キーはバックアップされず、デバイスのリセット時にクリアされます。リセット後はキーの取り出しはできなくなるので、ユーザに対してキーの再入力を求めるプロンプトが表示されます。

次のサンプル・コード (Embedded SQL) は、暗号化キーの保存と取り出し方法を示しています。

```
#define UL_STORE_PARMS StoreParms
 static ul_char StoreParms[STORE_PARMS_MAX];
 ...
 startupRoutine() {
    ul_char buffer[MAX_PWD];

    if( !ULRetrieveEncryptionKey(
           buffer, MAX_PWD, NULL, NULL  ) ){
       // prompt user for key
       userPrompt( buffer, MAX_PWD );
      if( !ULSaveEncryptionKey( buffer, NULL, NULL ) )
{
          // inform user save failed
       }
```

```
        }
        // build store parms
        StrCopy( StoreParms, "key=" );
        StrCat( StoreParms, buffer );
        ULPalmLaunch( &sqlca, UL_NULL );
    }
```

次のサンプル・コードは、暗号化キーをクリアすることでデバイスのセキュリティを高めるメニュー項目の使用方法を示しています。

```
case MenuItemClear
     ULClearEncryptionKey( NULL, NULL );
     break;
```

Ultra Light のユーザ認証

Ultra Light には、ユーザ認証用のデータベース・ユーザ ID とパスワードがオプションで用意されています。Adaptive Server Anywhere やその他のマルチユーザ・データベース・システムとは異なり、Ultra Light ユーザ ID は認証のみに使用されます。パーミッションのチェックやデータベース内のオブジェクトの所有権には使用されません。デフォルトでは、Ultra Light データベースではユーザ認証は行われません。いったんデータベースに接続すると、各ユーザはデータベースに完全にアクセスできます。

Ultra Light データベースが作成されると、初期ユーザ ID として DBA とパスワード SQL が割り当てられます。これらは接続時のデフォルト値でもあるため、ユーザ ID またはパスワード接続パラメータを指定しない場合、ユーザ認証を回避できます。

Ultra Light では、16 文字より短いユーザ ID とパスワードを使用して、一度に最大 4 人のユーザ ID を定義できます。

データベースの大文字と小文字を区別しない場合は (デフォルト)、ユーザ ID とパスワードも大文字と小文字を区別しません。データベースの大文字と小文字を区別する場合は、パスワードの大文字と小文字を区別します。

Ultra Light ユーザは既存の接続から追加します。つまり、ユーザ ID またはパスワードを変更してアプリケーションにユーザ認証を追加する場合、デフォルトのユーザ ID とパスワードを使用してデータベースに接続した後でこの作業を行います。また、ユーザ ID の変更はできません。この場合、ユーザを 1 人追加して、既存のユーザを削除します。パスワードの変更に使用する関数は、ユーザ ID の変更に使用するものと同じです。

ユーザ認証を追加するための一般的スキーマは、次のとおりです。

❖ アプリケーションにユーザ認証を追加するには、次の手順に従います。

1. デフォルトの **uid** パラメータと **pwd** パラメータを使用してデータベースに接続します。

新しいユーザは既存の接続から追加する必要があります。このため、データベースへの初回の接続は、DBA と SQL のデフォルトのユーザ ID とパスワードを使用して行います。

2 ユーザ ID とパスワード入力のプロンプトを表示します。

ユーザに対してプロンプトを表示する方法は、アプリケーションによって異なります。

3 このユーザ ID とパスワードの組み合わせにアクセスを許可します。

アクセスの許可方法は、使用しているインタフェースによって異なります。

4 元のユーザ ID のアクセス権を取り消します。

運用環境では、デフォルトのユーザ ID とパスワードにデータベースへのアクセス権を残しておくと、セキュリティ上の問題が発生します。

Ultra Light ユーザ ID は、Mobile Link ユーザ名や、開発中や配備後に使用するリファレンス・データベースまたは統合データベースで使用されるユーザ ID とは別のものです。多くの場合、これら 2 つに同じ値を指定するかもしれませんが、それぞれの概念はまったく異なっています。たとえば、CustDB サンプル・アプリケーションでは、アプリケーションの起動時に従業員番号を入力するためのプロンプトが表示されます。この従業員番号は Mobile Link 同期を行うためのデータベースを識別するものであり、接続またはデータ・アクセスを行うための Ultra Light ユーザ ID ではありません。

関連項目

- **Ultra Light for MobileVB**『Ultra Light ActiveX ユーザーズ・ガイド』>「ユーザの認証」を参照してください。
- **Ultra Light.NET**『Ultra Light.NET ユーザーズ・ガイド』>「ユーザ認証」を参照してください。
- **Ultra Light C++ コンポーネント**『Ultra Light C/C++ ユーザーズ・ガイド』>「ユーザの認証」を参照してください。
- **Ultra Light ActiveX**『Ultra Light ActiveX ユーザーズ・ガイド』>「ユーザの認証」を参照してください。

- **Native Ultra Light for Java**『Native Ultra Light for Java ユーザーズ・ガイド』>「ユーザ認証」を参照してください。
- **Embedded SQL**『Ultra Light C/C++ ユーザーズ・ガイド』>「ユーザの認証」を参照してください。
- **静的型 C++ API**『Ultra Light C/C++ ユーザーズ・ガイド』>「ユーザの認証」を参照してください。
- **Ultra Light 静的型 Java**『Ultra Light 静的型 Java ユーザーズ・ガイド』>「アプリケーションへのユーザ認証の追加」を参照してください。
- **Ultra Light for M-Business Anywhere**『UltraLite for M-Business Anywhere User's Guide』>「Authenticating users」を参照してください。

Mobile Link と Ultra Light のユーザ ID の共有

Ultra Light と Mobile Link のユーザ認証のメカニズムはそれぞれ違いますが、1 つのユーザ ID とパスワードで Mobile Link と Ultra Light の両方のユーザ認証を提供できます。ユーザ ID とパスワードを共有するには、これらを変数内に格納し、同じ変数を Ultra Light のユーザ認証の呼び出しと同期の呼び出しで使用します。

Mobile Link 統合サイトでパスワードがリセットされた場合に新しいパスワードのためのプロンプトが表示されるように、アプリケーションを設計できます。

❖ Mobile Link または Ultra Light の新しいパスワードのためのプロンプトを表示するには、次の手順に従います。

1. ユーザ ID とパスワードを変数内に保存します。
2. 同期を実行します。
3. ユーザの認証が行われなかったために同期が失敗した場合は、新しいパスワードを入力するように要求されます。

4 適切な関数またはメソッドを使用して Ultra Light ユーザのパスワードを更新します。

5 同期情報を更新し、同期を再実行します。

Mobile Link ユーザの認証については、『Mobile Link クライアント』>「Mobile Link ユーザの認証」を参照してください。

Ultra Light の文字セット

Ultra Light アプリケーションには、文字セットに関する問題が発生する場面がいくつかあります。

- **Ultra Light スキーマ** データベース自体には、データベースの作成時に指定する単一の照合順 (文字セットとソート順) があります。照合順によって、インデックス内の文字データの順序や文字比較の結果などが決まります。

  詳細については、「Ultra Light データベース文字セット」58 ページを参照してください。

- **Ultra Light ランタイム・ライブラリまたはコンポーネント** データベースにアクセスする Ultra Light コンポーネントまたはランタイム・ライブラリは、メッセージや環境との対話用の文字セットを使用します。

  ランタイム文字セットは、ユニコードでも ANSI でもかまいません。使用される文字セットによって、データがデータベース・ファイルに保管される方法が決まります。データベースを管理するには、作成時に使用したのと同じ文字セットのランタイムを使用します。

  詳細については、「Ultra Light ランタイム文字セット」59 ページを参照してください。

- **同期** Ultra Light データベースのデータと Mobile Link 同期サーバを同期する場合、Ultra Light データベースで使用される文字セットと統合データベースで使用される文字セットの一貫性が保たれる必要があります。

  詳細については、「同期と文字セット」61 ページを参照してください。

Ultra Light データベース文字セット

Ultra Light コンポーネントを使用している場合、スキーマ・ペインタを使用して Ultra Light データベース・スキーマを作成するには、データベース・スキーマを作成するときに文字セットと照合順を指定します。

Ultra Light アプリケーションは、処理の効率化を図るため、ターゲット・プラットフォームのネイティブのマルチバイト文字コードを使用します。リファレンス・データベースが別の文字コードを使用する場合、Ultra Light アプリケーションはターゲット・デバイスのデフォルトの照合を使用します。

次のいずれかの条件に該当する場合、Ultra Light アプリケーションはリファレンス・データベースの照合順を採用します。

- リファレンス・データベースがシングルバイト文字セットを使用する場合。
- ターゲット・デバイスのネイティブの文字コードがマルチバイト、リファレンス・データベースの文字コードもマルチバイトであり、Ultra Light アナライザがリファレンス・データベースで使用する照合順を発見した場合。

たとえば、932JPN リファレンス・データベースを使用して Palm Computing Platform 日本語版用のアプリケーションを構築する場合、Ultra Light アプリケーションは照合情報を継承できます。これは、ネイティブの文字コードが、リファレンス・データベースと同じ文字コードを使用しているためです。ただし、932JPN リファレンス・データベースを使用して Windows CE プラットフォーム用の Ultra Light アプリケーションを構築する場合、アプリケーションはユニコードとデフォルトのユニコード照合情報を使用します。

ソート順

文字セットがシングルバイトであるか、ターゲット・デバイスのネイティブの文字セットがリファレンス・データベースの文字セットと同じである場合、CHAR(*n*) または VARCHAR(*n*) のカラムについて、リファレンス・データベースの照合順に従って比較とソートが行われます。

データベースの作成については、「Ultra Light データベースとスキーマの作成」36 ページを参照してください。

データ記憶領域

文字セットの保管方法は、スキーマの作成時の照合順だけでなく、データベースを管理する Ultra Light ランタイム・ライブラリの文字セット (ANSI またはユニコード) によって決まります。

詳細については、「Ultra Light ランタイム文字セット」59 ページを参照してください。

Ultra Light ランタイム文字セット

Ultra Light ランタイム・ライブラリの文字セットは、ターゲット・オペレーティング・システムによって異なります。文字セットは、データをアプリケーションと交換する方法を決定し、データ記憶領域にも影響します。この項では、Ultra Light がサポートしているプラットフォームで使用される文字セットに関する補足情報について説明します。

Palm Computing Platform

シングルバイト Palm Computing Platform デバイスは、Code Page 1252 (Windows US コード・ページ) に基づいた文字セットを使用します。1252Latin1 コード・ページは、Palm Computing Platform 用のアプリケーション開発に適しています。1252Latin1 コード・ページは、デフォルトの Adaptive Server Anywhere 照合順です。Palm Computing Platform 日本語版デバイスでは 932JPN 文字セットを使用します。

Windows CE

Windows CE オペレーティング・システムは、ユニコードを使用します。Windows CE で稼働する Ultra Light も、ユニコードで CHAR(*n*) と VARCHAR(*n*) カラムを保管します。Adaptive Server Anywhere 照合順は、8 ビットの ASCII 文字セットの動作を定義します。

Windows CE 用の Ultra Light アプリケーションは、Adaptive Server Anywhere 照合順に対応する 8 ビット ASCII 文字を持つユニコード文字を比較するときに、Adaptive Server Anywhere 照合順を使用します。そのとき、アクセント付きの文字とアクセントなしの文字は区別しないでソートします。対応する 8 ビット ASCII 文字を持たないユニコード文字である場合は、2 つのユニコード文字を比較します。

Windows デスクトップ・オペレーティング・システム

Ultra Light コンポーネントはすべてユニコード・ベースです。

Windows NT/2000/XP と Windows 98 上の Ultra Light Embedded SQL および静的型 C++ アプリケーションによって使用されるランタイム・ライブラリは、ANSI バージョンと UNICODE バージョンの両方で提供されます。バージョン 9 より前の Ultra Light バージョンには、ANSI バージョンのランタイム・ライブラリのみが含まれていました。

データベース・ファイルの互換性

データベースを作成するために使用されるランタイムまたはコンポーネントによって、文字がデータベースに保管される方法が決まります。ANSI コンポーネントまたはランタイム・ライブラリを使用して Ultra Light データベースを作成し、このデータベース・ファイルをユニコード・コンポーネントまたはランタイム・ライブラリで使用することはできません。

次の表は、Ultra Light コンポーネントまたはランタイム・ライブラリによって使用されている文字セットをリストします。これらの文字セットによって、あるバージョンの Ultra Light によって作成されたデータベース・ファイルを別のバージョンで使用できるかどうかが決まります。特に、Windows オペレーティング・システム (Windows CE を除く) 上でバージョン 8.0.2 の Ultra Light for MobileVB または Ultra Light ActiveX によって作成されたデータベースは、バージョン 9.0 以降のソフトウェアでは開くことができないので注意してください。

環境	8.0.2	9.0 以降
Windows コンポーネント	ANSI	ユニコード
Windows CE コンポーネント	ユニコード	ユニコード
Windows (Native Ultra Light for Java)	ユニコード	ユニコード
Windows 8.0.2 DLL	ANSI	なし
Windows (*ulrt9.dll*)	なし	ANSI
Windows (*ulrtw9.dll*)	なし	ユニコード
Windows (*ulrtcw9.dll* (エンジン))	なし	ユニコード
Windows CE	ユニコード	ユニコード

静的型 Java

エラー処理オブジェクトの **SQLException** と **SQLWarning** を使用すると、Java アプリケーションでエラー・メッセージや警告メッセージを表示できます。デフォルトでは、これらのメッセージは英語で表示されます。

Java Locale を適切な言語に設定すると、ローカライズされたエラー・メッセージや警告メッセージを英語以外の言語で表示できます。たとえば、フランス語でメッセージを表示するには、次のようなコード・フラグメントを使用します。

```
java.util.Locale locale = new java.util.Locale( "fr",
"" );
 java.util.Locale.setDefault( locale );
```

デフォルトの Locale は、プログラムの起動時に設定します。メッセージをエラー処理オブジェクトにすると、プログラムの実行時には、そのメッセージの表示に使用する言語が使用されることになります。詳細については、Java のマニュアルを参照してください。

同期と文字セット

同期を行うとき、Mobile Link 同期サーバは、必ずアプリケーション・データベースからアップロードされた文字をユニコードに変換し、Unicode ODBC API を使用して統合データベース・サーバに渡します。統合データベース・サーバ、または ODBC ドライバは、受け取った文字を統合データベースの文字コードに変換するために必要な処理を実行します。統合データベースがユニコードを使用している場合を除いて、この 2 度目の変換は必ず実行されます。

情報をダウンロードすると、統合データベース・サーバは文字をユニコードに変換します。次に、Mobile Link 同期サーバが、Ultra Light アプリケーションの要件に応じて自動的に文字を変換します。

Ultra Light アプリケーションと統合データベースが同じ文字コードを使用する場合、変換は行われません。変換が必要となる場合、Ultra Light アプリケーションの複数の文字コードが 1 つのユニコードにマップされたりする問題が生じることがあります。また、逆の現象も考えられます。このような場合、Mobile Link 同期サーバは一貫性を保ちながら変換処理を実行しますが、その動作は統合データベース・サーバの変換メカニズムよって異なります。

Ultra Light データベースの内部論理

Ultra Light は、各テーブルのデータのローを格納するほか、各ローのステータス情報や、ローに効率的にアクセスするためのインデックスも格納します。

Ultra Light データベース・ファイル

Ultra Light データベースは、データベースのテーブル、インデックス、および他のすべての情報が保持された単一ファイルに格納されます。

テーブルとインデックスは、固定サイズのページに格納されます。データベース I/O 操作は一度に 1 ページずつ実行されます。デフォルトのページ・サイズは 4 K です。詳細については、「PageSize 接続パラメータ」109 ページを参照してください。

テンポラリ・ファイル

Ultra Light は、テンポラリ・ファイルを管理して、処理中の情報を保持したり、アプリケーションの状態に関する情報を保持します。このファイルは、Ultra Light が動作していないときにデータを失うことなく削除できます。

Windows と Windows CE では、Ultra Light データベース・ファイルの名前は一般に *dbname.udb* の形式です。この場合、テンポラリ・ファイルは *dbname.~db* (パスは同じ) です。データベース名が異なる形式の場合は、テンポラリ・ファイルの名前の最後に '~' が追加されます。Palm OS では、テンポラリ・ファイルは *ul_tmp_<creator-id>* です。

Palm OS

Palm OS 上の Ultra Light データベースは、拡張カードの Palm 仮想ファイル・システムか、Palm レコードベースのデータ・ストアに格納できます。詳細については、「Palm OS のデータベース保管の選択」240 ページを参照してください。

Ultra Light がローのステータスを記録する方法

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。ローのステータスは、トランザクション処理、リカバリ、同期の制御に使用されます。

削除を実行すると、影響を受けるローのステータスは、削除されたことを反映するように変更されます。削除のロールバックは、ローを元のステータスにリストアすることと同様、簡単に実行できます。

削除をコミットしても、必ずしも、影響を受けるローがメモリから削除されるとはかぎりません。一度も同期が行われていないローは削除されます。同期されたローの場合、次の同期によって統合データベースでの削除が確認されるまで、そのローは削除されません。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。

同様に、Ultra Light データベースのローを更新すると、その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。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。

ローの古いバージョンは、同期の後に削除されます。同期してから次の同期までの間に複数回ローが更新された場合は、ローの最も古いバージョンと最も新しいバージョンが保持されます。

Ultra Light テーブルのプライマリ・キー

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。

静的 API の場合、Ultra Light ジェネレータは、リファレンス・データベースのプライマリ・キー・カラムを使用して、Ultra Light データベースのプライマリ・キー・カラムを生成します。Ultra Light データベースで使用するデータにテーブルのプライマリ・キー・カラムが含まれていない場合、Ultra Light ジェネレータはテーブルの一意性制約

を検索し、一意性制約となるカラムを Ultra Light データベースのプライマリ・キーとします。ユニークなカラムがない場合、ジェネレータはエラーをレポートします。

Ultra Light データベースのインデックス

Ultra Light インデックスは、インデックス・エントリが非常に少ない B+ ツリーに似ています。

pure Java データベースを除いて、各インデックス・エントリは 2 バイトで、各インデックス・ページには 256 のエントリが含まれます。また、各インデックスには固定されたオーバヘッドがあるため、Ultra Light インデックスが使用するメモリは、テーブル内の 1 つのローにつき 2 バイト超になります。各インデックスのオーバヘッドは、1 つのインデックスにつき 1 KB 強です。通常、サイズの大きなテーブルでは、Ultra Light インデックス・ページの 85% 以上が使用されます。

Ultra Light Java データベースのメモリ要件には、一貫したルールはありません。

トランザクション処理、リカバリ、バックアップ

Ultra Light では、トランザクション処理を使用できます。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どちらかです。トランザクションが「**コミット**」されると、すべての処理がデータベースに格納されます。また、トランザクションが「**ロールバック**」されると、処理が 1 つもデータベースに格納されません。

一部のトランザクションは、単一の処理から成ります。多くのプログラミング・インタフェースは、「**オートコミット**」設定を使用して処理の後にトランザクションをコミットします。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。

システム障害からのリカバリ

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を保つ状態にリカバリされます。障害が発生する前にコミットされたすべてのトランザクションは、Ultra Light データベースに保持されます。障害発生時にコミット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・バイトを使用して、リカバリ時のローの処理を決定します。Ultra Light データベースのローを挿入、更新、または削除すると、その操作内容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、リカバリするときにトランザクション全体がロールバックされます。

ステータス・バイトの詳細については、「Ultra Light がローのステータスを記録する方法」63 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 データベース・ストアそのものが破損した場合を考慮した唯一の保護機能は、同期です。

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 データベースの制限事項

次の表は、ソフトウェアのデータ構造によって制限される Ultra Light データベース・オブジェクトのサイズと数に関する絶対制限値を示します。実際には、コンピュータのメモリ、CPU、記憶装置から受ける制限の方が厳しいのが普通です。

項目	制限
接続数／データベース	14
カラム数／テーブル	65535 であるが、ロー・サイズによって制限される。 ロー・サイズは約 4 KB に制限されるので、テーブルあたりのカラム数の実際的な制限はこれよりも大幅に小さくなり、ほとんどの状況で 4000 をかなり下回ります。
インデックス数	約 1000。 ページ・サイズを 2 KB に設定した場合、テーブルあたりのインデックスの最大数は約 500 に減少します。
ロー数／データベース	継続保管によって制限される。
ロー数／テーブル	65534
テーブル数／データベース	約 1000。 ページ・サイズを 2 KB に設定した場合、テーブルの最大数は約 500 に減少します。
参照されるテーブル数／トランザクション	制限なし。

項目	制限
ロー・サイズ	約 4 KB (圧縮)。LONG VARCHAR と LONG BINARY 値は個別に保管されるため、制限値 4 KB には含まれない。
ファイルベースの継続保管	2 GB ファイル、または OS 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)

その他の制限事項については、「Ultra Light の SQL サポートの概要」 180 ページを参照してください。

Ultra Light ではサポートされていない Adaptive Server Anywhere 機能

次の Adaptive Server Anywhere 機能は、Ultra Light データベースではサポートされていません。

- **カスケード更新とカスケード削除**　アプリケーションの中には、ビジネス・ルールを実装するために宣言制約や参照整合性に依存するものがあります。更新と削除が自動的にカスケードされる場合、Ultra Light では、同期ダウンロード時を除いてこれらの機能は使用できません。

  外部キーに該当する値があるプライマリ・キーを削除しようとすると、エラーが発生します。また、外部キーが元の値を参照しているときにプライマリ・キーの値を更新しようとする場合も、エラーが発生します。

- **検査制約**　テーブルやカラムの検査制約を Ultra Light データベースに含めることはできません。

- **計算カラム**　計算カラムを Ultra Light データベースに含めることはできません。

- **グローバル・テンポラリ・テーブル** Ultra Light は、グローバル・テンポラリ・テーブルを一時的な状態として認識するわけではありません。永続的なベース・テーブルとして認識します。
- **宣言されたテンポラリ・テーブル** Ultra Light アプリケーションでは、テンポラリ・テーブルを宣言できません。
- **ストアド・プロシージャ** Ultra Light アプリケーションでは、ストアド・プロシージャやユーザ定義関数を呼び出すことはできません。
- **関数** 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。

  サポートされている関数のリストについては、「Ultra Light SQL 関数」188 ページを参照してください。
- **トリガ** トリガは、Ultra Light データベースではサポートされていません。
- **システム・テーブルへのアクセス** Ultra Light データベースには、システム・テーブルは存在しません。
- **システム関数** Ultra Light アプリケーションでは、プロパティ関数を含む Adaptive Server Anywhere システム関数を使用できません。
- **データベースにおける Java** データベースでは、クエリに Java メソッドを含めたり、別の方法で Java を使用したりすることはできません。
- **timestamp カラム** Ultra Light データベースでは、Transact-SQL の timestamp カラムは使用できません。Transact-SQL の timestamp カラムは、次のデフォルトで作成されます。

  ```
  DEFAULT TIMESTAMP
  ```

  作成したカラムを次のように使用できます。

  ```
  DEFAULT CURRENT TIMESTAMP
  ```

この 2 つの間には、動作に違いがあります。ローが更新されたとき、DEFAULT CURRENT TIMESTAMP カラムは自動的には更新されませんが、DEFAULT TIMESTAMP カラムは自動的に更新されます。DEFAULT CURRENT TIMESTAMP カラムに最後の更新時間を反映させるには、そのカラムを明示的に更新してください。

SQL の制限事項

Ultra Light アプリケーションで動的 SQL を使用している場合、使用可能な SQL の範囲は Adaptive Server Anywhere よりも狭くなります。

詳細については、「動的 SQL」201 ページを参照してください。

静的インタフェースを使用している Ultra Light アプリケーションはより広範な SQL を使用できますが、次の SQL 機能はサポートされていません。

- **動的 SQL**　動的 SQL は、静的インタフェースを使用しているアプリケーションでは使用できません。
- **SAVEPOINT 文**　Ultra Light データベースは、トランザクションのサポートは行いますが、トランザクションのセーブポイントはサポートしません。
- **SET OPTION 文**　Ultra Light データベースはオプションのセットをサポートしていますが、Ultra Light アプリケーションの SET OPTION 文を使用してオプション設定を変更することはできません。

  Ultra Light コンポーネントの場合、データベース・スキーマ・ファイルのオプションを設定するには、これらをリファレンス・データベースに設定して ulinit を使用するか、Ultra Light スキーマ・ペインタを使用するか、スキーマを保持する XML ドキュメントに設定します。

  Ultra Light データベース・オプションの詳細については、「Ultra Light データベース・オプション」45 ページを参照してください。
- **スキーマの変更**　静的インタフェース・アプリケーションから Ultra Light データベースのスキーマを変更するには、アプリケーションの新規バージョンを構築します。

Ultra Light コンポーネントから Ultra Light データベースのスキーマを変更する場合、スキーマ変更文を使用するか、更新したスキーマ・ファイルを配備できます。

詳細については、「Ultra Light データベース・スキーマのアップグレード」71 ページを参照してください。

Ultra Light データベース・スキーマのアップグレード

Ultra Light アプリケーションの新しいバージョンを開発する場合、データベース・スキーマを変更できます。アップグレードした Ultra Light スキーマを配備し、何らかの制約を受ける既存のエンド・ユーザ・データベース内のデータを管理できます。この機能は、静的型 Java API を使用して構築された Ultra Light アプリケーションでは使用できません。

警告
スキーマのアップグレードは取り消すことができません。スキーマのアップグレード中にエラーが発生すると、使用不能なデータベースとともにエラーが残ってしまう可能性があります。重要なデータが含まれるデータベースのスキーマをアップグレードする場合は、データベース・ファイルを同期またはコピーし、アプリケーションの変更内容をすべてバックアップしてください。

アップグレードしたスキーマを配備するためのメカニズムは、Ultra Light コンポーネント (新しいスキーマ・ファイルを必要とします) を使用しているか、または静的インターフェース (スキーマ定義は、生成されたアプリケーション・コードに格納されています) を使用しているかによって異なります。

❖ 新しいスキーマを配備するには、次の手順に従います (Ultra Light コンポーネント)。

1　新しいスキーマ・ファイルを作成します。

スキーマ・ペインタまたは ulinit を使用して新しいスキーマを作成できます。

スキーマ・ペインタを使用する場合は、古い名前と新しい名前の間のマッピングを定義することによって、配備するスキーマを準備します。このマッピングは、移行中のデータの消失を防ぐために使用されます。

a. スキーマ・ペインタで、[ファイル] ー [スキーマの展開の準備] を選択します。

b. 古いテーブル名またはカラム名と、新しいデータベース・スキーマ内での対応する名前の間に、マッピングを指定します。

ulinit を使用して新しいスキーマを作成する場合、名前が変更されたカラム内のデータが失われます。スキーマは、古いカラムが削除されて新しいカラムが作成されたものとして解釈されます。

2 スキーマ・ファイルを既存のデータベースに適用します。

コンポーネントは、データベース・スキーマを DatabaseSchema または ULDatabaseSchema オブジェクトとして公開します。スキーマ・オブジェクトは、Connection または ULConnection オブジェクトの DatabaseSchema プロパティを使用して取得します。

ULDatabaseSchema オブジェクトの ApplyFile メソッドを使用して新しいスキーマを適用します。

❖ 新しいスキーマを配備するには、次の手順に従います (Embedded SQL と静的型 C++ API)。

1 リファレンス・データベースのスキーマを修正します。

2 アプリケーションの新しいバージョンを作成します。

3 アプリケーションが ULEnableGenericSchema() を呼び出すことを確認します。

静的インタフェースが組み込まれた新しい Ultra Light アプリケーションがデバイスに配備されると、Ultra Light はデフォルトで空のデータベースを再作成するため、新しいアプリケーションが配備される前にデータベース内にあったデータは失われます。ULRegisterSchemaUpgradeObserver を呼び出すと、既存のデータベースが新しいアプリケーションのスキー

マにアップグレードされます。データベース・スキーマをアップグレードするアプリケーションはこの関数を呼び出します。

『Ultra Light C/C++ ユーザーズ・ガイド』>「ULRegisterSchemaUpgradeObserver 関数」を参照してください。

4　新しいアプリケーションが適用されると、スキーマは自動的にアップグレードされます。

スキーマのアップグレード方法については、「スキーマのアップグレード方法」74 ページを参照してください。

スキーマのアップグレードの監視

データベース・スキーマのアップグレードには時間がかかることがあります。Upgrade observer によって、ユーザに進捗情報が表示され、ユーザはこのプロセスの最初のステップでアップグレードをキャンセルできます。

このメカニズムは、使用しているインタフェースによって異なります。

- **Ultra Light for MobileVB** OnSchemaUpgradeStateChange および OnSchemaUpgradeProgress イベントのハンドラを実装します。

  『Ultra Light for MobileVB ユーザーズ・ガイド』>「OnSchemaUpgradeProgress イベント」と『Ultra Light for MobileVB ユーザーズ・ガイド』>「OnSchemaUpgradeStateChange イベント」を参照してください。

- **Ultra Light.NET** SchemaUpgradeListener インタフェースを実装し、DatabaseSchema.ApplyFile() にオブジェクトを指定します。スキーマのアップグレード時には、SchemaUpgradeListener.SchemaUpgrading メソッドが呼び出されます。

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

- **Native Ultra Light for Java** SchemaUpgradeListener インタフェースを実装し、DatabaseSchema.applyFile() にオブジェクトを指定します。スキーマのアップグレード時には、SchemaUpgradeListener.schemaUpgrading メソッドが呼び出されます。

  詳細については、『API リファレンス』の「**ianywhere.native_ultralite.SchemaUpgradeListener**」を参照してください。

- **Ultra Light C/C++ インタフェース** スキーマのアップグレード・イベントを処理するコールバック関数を実装および登録します。

  『Ultra Light C/C++ ユーザーズ・ガイド』>「ULRegisterSchemaUpgradeObserver 関数」と『Ultra Light C/C++ ユーザーズ・ガイド』>「ULRegisterSchemaUpgradeObserver のコールバック関数」を参照してください。

- **Ultra Light ActiveX** この機能は Ultra Light ActiveX には使用できません。

- **Ultra Light for M-Business Anywhere** この機能は Ultra Light for M-Business Anywhere には使用できません。

スキーマのアップグレード方法

アップグレード前のバックアップ
スキーマのアップグレードを試行する前に、データベース・ファイルをコピーするか同期をとって、データをバックアップすることを強くおすすめします。

スキーマのアップグレード・プロセスでは、古いスキーマと新しいスキーマで一致する名前を使用します。データベース内のローに新しいスキーマとの互換性がない場合、このローはデータベースから削除されます。一般に、データを持つテーブルに制約を追加したり、予測できないカラム変換を実行したりすると、ローが失われることがあります。

スキーマは次のようにアップグレードされます。

1. 新しいスキーマにはない古いテーブルは、削除されます。
2. 新しいテーブルが作成されます。
3. 古いスキーマと新しいスキーマの両方に存在するテーブルについては、必要に応じてカラムが追加、削除されます。新しいカラムが null 入力不可能で、デフォルト値を持たない場合は、0 (数値データ型)、空の文字列 (文字データ型)、空のバイナリ値が入力されます。
4. プロパティが変更されたカラムは修正されます。

警告

任意のローの変換中にエラーが発生した場合は、そのローが削除され、SQL 警告 SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE が生成されます。

5. インデックスと制約が再構築されます。この手順によって、たとえばインデックスが UNIQUE として再定義され、重複する値を持つ場合、ローが削除されることもあります。

Ultra Light ソフトウェアのアップグレード

Windows CE 以外の Windows オペレーティング・システムでは、Ultra Light for MobileVB または Ultra Light ActiveX を使用している場合、8.0.2 ソフトウェアを使用して作成された Ultra Light データベース・ファイルを開くことはできません。詳細については、『SQL Anywhere Studio 新機能ガイド』> 「Ultra Light の動作の変更」を参照してください。

Ultra Light データベース・ファイルの以前のリリースとの互換性については、「Ultra Light ランタイム文字セット」59 ページを参照してください。

Ultra Light ランタイム

「**Ultra Light ランタイム**」は、Ultra Light データベースおよび同期を管理します。静的型 Java API を除いて、各ターゲット・プラットフォームの Ultra Light ランタイムは、単一のコードベースに基づいています。Ultra Light ランタイムにはさまざまな形式があります。

- **コンポーネント** Ultra Light ランタイムは、各コンポーネントにリンクされます。
- **静的ライブラリ** 静的インタフェースの場合、Ultra Light ランタイムは、アプリケーションにリンクする静的ライブラリとして提供されています。配備する個別のファイルはありません。
- **動的ライブラリ** C/C++ Windows および Windows CE アプリケーションの場合、Ultra Light ランタイムは、動的リンク・ライブラリ (DLL) として提供されます。
- **個別の実行プログラム** Windows と Windows CE の場合、「**Ultra Light エンジン**」と呼ばれる個別の実行プログラムが提供されています。Ultra Light エンジンは、複数のアプリケーションからの接続をサポートするランタイムの唯一のバージョンです。このようなアプリケーションは、Ultra Light エンジンを使用するときにそれぞれクライアント・ライブラリにリンクする必要があります。

  詳細については、「Ultra Light エンジンの使用」80 ページを参照してください。

Ultra Light での同時実行性について

Ultra Light データベースは、複数の要求を同時に受け取ることがあります。同時要求を正確に処理するアプリケーションを設計するには、Ultra Light がデータベースにおける同時実行性を管理する方法について理解する必要があります。

概念

データベースへの同時アクセスについて考える場合、さまざまな概念を分けて考えると分かりやすくなります。これらの概念は、高レベルから低レベルに分けられます。

- **アプリケーション** Ultra Light エンジンは、複数の個別アプリケーションからの要求に応えることができます。Ultra Light ランタイムの他のバージョンでは、データベースに同時に接続できるアプリケーションは 1 つのみです。

  詳細については、「Ultra Light ランタイム」76 ページを参照してください。

- **スレッド** Ultra Light ランタイムは、マルチスレッド・アプリケーションをサポートしています。1 つのアプリケーションを作成して、それぞれがデータベースに接続できる複数のスレッドを使用できます。

  詳細については、「Ultra Light アプリケーションのスレッド」79 ページを参照してください。

- **接続** 単一スレッド・アプリケーションがデータベースに対して複数の接続を開くこともできます。この場合、個々の接続が使用できるのは単一スレッドのみです。

- **トランザクション** 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から成ります。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継続保管されません。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。

- **要求** トランザクションは、1 つまたは複数の要求から成ります。要求には、データを読み込むクエリ、データを変更する挿入、更新、削除、または同期などがあります。

- **現在のロー** アプリケーションによってクエリの結果セットが処理されている場合、ポインタは結果セット内の「**現在のロー**」に配置されています。一部のインタフェースでは、この現在のローはカーソル (結果セット内の位置を示すポインタ) を使用して明示的に示されます。他のインタフェースで

は、結果セット・オブジェクトまたはテーブル・オブジェクトでメソッドを使用して現在のローを識別したり、変更します。このようなメソッドは、カーソルを背後で使用します。

複数のデータベース

Ultra Light ランタイムは、一度に最大 4 台のデータベースを管理できます。1 つの Ultra Light アプリケーションが個々のデータベースに対して複数の接続を開くことができます。各データベースのデータは独立しているため、このようなアプリケーションでは同時実行性に関する問題は発生しません。

ロッキング

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。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。アプリケーションは、データ変更の試行後に SQLCODE の値を確認します。

ローの再読み込み

ロッキングと同時実行性の管理方法を理解するには、それぞれ独自のトランザクションを持つ接続 A と接続 B を考えてみると分かりやすくなります。

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。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。このとき、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。データベースの変更内容は、接続 A がトランザクションをコミットするまで継続保管されません。

ローを読み込んだりフェッチしても、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しない場合、接続 B はこのローを変更できます。

接続 B が現在のローを変更する場合、このローはロックされます。これによって、接続 A は現在のローを変更できなくなります。接続 A が現在のローを再フェッチしたときにこのローが削除されている場合、接続 A は結果セット内の次のローを取得します。このローも削

除されている場合、接続 A はローの最新のコピーを取得します。接続 A が使用しているインデックスのカラムが変更されている場合、接続 A は、この変更が削除後に行われた挿入だと判断し、結果セット内の次のローを取得します。

同期

同期は、個々の接続として動作します。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して、同期時にデータへのアクセスを無効にすることができます。

詳細については、『Mobile Link クライアント』> 「Disable Concurrency 同期パラメータ」を参照してください。

Ultra Light アプリケーションのスレッド

Ultra Light ランタイム・ライブラリはスレッド対応であるため、開発ツールとターゲット・プラットフォームの両方がマルチスレッド・アプリケーションをサポートしていれば、マルチスレッド・アプリケーションを開発できます。例外は次のとおりです。

- 基礎となる開発ツールの制限事項のため、Ultra Light for MobileVB を使用してマルチスレッド・アプリケーションを開発することはできません。
- eMbedded Visual Basic、JScript、JavaScript の制限事項のため、Ultra Light ActiveX または Ultra Light for M-Business Anywhere を使用してマルチスレッド・アプリケーションを開発することはできません。
- オペレーティング・システムの制限事項のため、Palm OS に対してマルチスレッド・アプリケーションを開発することはできません。
- 静的インタフェースからは、同じクエリを一度に複数回実行することはできません。このため、特定のクエリの結果セットの複数のインスタンスに一度にアクセスすることはできません。

各スレッドは、Database Manager オブジェクトまたは静的インタフェースにおけるその同等物 (ULData、SQLCA) を含む、オブジェクトの独自セットを管理します。Database Manager オブジェクトは、スレッド上で初期化しなくてもスレッドに渡すことができますが、データの管理はすべて個々のスレッド上で行います。

静的型 Java API

静的型 Java API によって使用される Ultra Light ランタイムでは、単一アプリケーションが単一データベースにアクセスできます。

Ultra Light ランタイムはスレッド対応です。Sun Java VM のユーザは、マルチスレッド Ultra Light アプリケーションを実行するにはバージョン 1.2 以降を使用してください。Pocket PC 上の Jeode VM と IBM Java VM のユーザは、これらの VM が JDK 1.1.8 に基づいている場合でも、マルチスレッド Ultra Light アプリケーションを実行できます。

Ultra Light ランタイム全体が単一のクリティカル・セクションとして扱われ、一度に 1 つのスレッドだけがこのセクションに入ることができます。

詳細については、『Ultra Light 静的型 Java ユーザーズ・ガイド』>「Ultra Light JdbcDatabase.connect メソッドの使用」を参照してください。

Ultra Light エンジンの使用

Ultra Light ランタイムをライブラリとしてまたはコンポーネント形式で配備する代わりに、Ultra Light エンジン形式で配備できます。

Ultra Light エンジンは、Ultra Light データベースのデータベース・サーバとして動作します。Ultra Light エンジンは、個別の実行プログラムとして Windows プラットフォームと Windows CE プラットフォームに対して提供されています。Ultra Light エンジンには、複数のアプリケーションからデータベースに対して同じマシンからアクセスできるという利点があります。一方、Ultra Light エンジンには、Ultra Light ランタイムの他のバージョンよりも多くのシステム・リソースが必要だという欠点があり、パフォーマンスが低下する可能性があります。

❖ Ultra Light エンジンを使用するには、次の手順に従います。

1　コンポーネントのクライアント・バージョンを配備します。

クライアント・バージョンは次のとおりです。パスは SQL Anywhere のインストールと同じです。

- **Ultra Light.NET (Windows CE)**
  UltraLite¥UltraLite.NET¥ce¥arm¥ulnetclient9.dll
- **Ultra Light.NET (Windows XP)** *win32¥ulnetclient9.dll*
- **Native Ultra Light for Java (Windows CE)**
  UltraLite¥NativeUltraLiteForJava¥ce¥arm¥julclient9.dll
- **Native Ultra Light for Java (Windows XP)**
  UltraLite¥NativeUltraLiteForJava¥win32¥julclient9.dll
- **C++ コンポーネント (Windows CE)**
  UltraLite¥ce¥arm¥lib¥ulrtc.lib (静的ライブラリ)
- **C++ コンポーネント (Windows XP)**
  UltraLite¥win32¥386¥lib¥ulimpcw.lib (インポート・ライブラリ) と *UltraLite¥win32¥386¥ulrtcw9.dll* (動的ライブラリ)
- **Ultra Light for M-Business Anywhere (Windows CE)**
  UltraLite¥UltraLiteForMBusinessAnywhere¥ce¥arm¥ulpodclient9.dll。クライアント・ライブラリは、*AvantGo¥Pods* ディレクトリの下に配置します。
- **Ultra Light for M-Business Anywhere (Windows XP)**
  UltraLite¥UltraLiteForMBusinessAnywhere¥win32¥386¥ulpodclient9.dll。クライアント・ライブラリは、*AvantGo¥Pods* ディレクトリの下に配置します。
- **Ultra Light ActiveX** 使用できません。
- **Ultra Light for MobileVB** 使用できません。

2　DatabaseManager 構成体のランタイム・タイプを指定します。

この手順が必要なのは、Native Ultra Light for Java と Ultra Light.NET のみです。

エンジンの起動

次の方法で、Ultra Light エンジンを起動できます。

- **Ultra Light エンジンの手動起動** コマンド・プロンプトで次のコマンドを入力します。

  ```
  dbuleng9
  ```

- **アプリケーションによる Ultra Light エンジンの起動** Ultra Light コンポーネントのクライアント・バージョンを使用して Ultra Light アプリケーションを配備すると、初回の接続時に Ultra Light エンジンが起動します。エンジンがすでに起動している場合、エンジンを使用してデータベースに接続します。

  dbuleng9.exe の場所を指定するには、最初にデータベースに接続するときに StartLine 接続パラメータを使用します。StartLine パラメータが指定されていない場合、Windows CE では、クライアントは *dbuleng9.exe* の場所を求めて *¥Windows*、root (¥)、*¥UltraLiteDB* ディレクトリの順に探し、他の Windows オペレーティング・システムでは、SQL Anywhere インストールの *win32* サブディレクトリを探します。

Ultra Light エンジンを使用してデータベースに接続するには、ユーザ ID (uid) とパスワード (pwd) 接続パラメータを指定します。

Ultra Light エンジンは、*dbulstop* ユーティリティを使用して手動で停止できます。また、アプリケーションにエンジンを停止させることもできます。データベースから最後に切断されるアプリケーションによって、Ultra Light エンジンは自動的に停止されます。

第 4 章

接続パラメータ

この章の内容

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

概要

Ultra Light データベースに接続するには、アプリケーションに接続パラメータを指定します。通常、接続パラメータには、データベースのファイル名とパスを入力することによって、接続を確立するデータベースを指定します。

1 つのアプリケーションを複数のプラットフォームに対してコンパイルすることがあるため、ターゲット・プラットフォームごとに個別のパラメータを使用してデータベースを識別できます。ユーザ認証が有効である場合、接続パラメータにはユーザ名とパスワードも指定します。

詳細については、「データベース識別パラメータ」89 ページと「OpenConnection パラメータ」95 ページを参照してください。

Ultra Light データベースは、アプリケーション自体によって作成されることが多く、データベースの特性は接続パラメータによって定義されます。接続パラメータには、スキーマ・ファイル (Ultra Light コンポーネント用) と、データベース機能を調整するためのオプションのパラメータを含めます。Embedded SQL、静的型 C++ API、または静的型 Java API を使用している場合、アプリケーションにすでに格納されている情報からデータベースが作成され、スキーマ・ファイルは必要ありません。

詳細については、「データベース・スキーマ・パラメータ」102 ページと「追加接続パラメータ」107 ページを参照してください。

すべての接続パラメータは大文字と小文字を区別しません。

次の表は、使用可能な接続パラメータのリストを示します。データベースの作成時のみ使用されるデータベース・スキーマ・パラメータには、アスタリスク (*) が付いています。

パラメータ	説明
Additional Parms	追加接続パラメータのプレースホルダ。 「AdditionalParms 接続パラメータ」90 ページを参照してください。

パラメータ	説明
Cache Size	キャッシュのサイズを定義します。「CacheSize 接続パラメータ」96 ページを参照してください。
Connection Name	接続名を指定します。「ConnectionName 接続パラメータ」97 ページを参照してください。
Database On CE	Windows CE で接続する Ultra Light データベース・ファイルのパスとファイル名。「DatabaseOnCE 接続パラメータ」91 ページを参照してください。
Database On Desktop	接続先の Ultra Light データベース・ファイルのパスとファイル名。「DatabaseOnDesktop 接続パラメータ」92 ページを参照してください。
Database On Palm	Palm OS 上の Ultra Light データベースの識別子。「DatabaseOnPalm 接続パラメータ」93 ページを参照してください。
Encryption Key	データベースの暗号化キー。「EncryptionKey 接続パラメータ」98 ページを参照してください。
Obfuscate*	簡単な暗号化スキーマをデータベースに適用します。「Obfuscate 接続パラメータ」108 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。「PageSize 接続パラメータ」109 ページを参照してください。
Palm Allow Backup	Palm OS デバイス上の HotSync バックアップの動作を制御します。「PalmAllowBackup パラメータ」110 ページを参照してください。
Password	ユーザのパスワード。「Password 接続パラメータ」99 ページを参照してください。
Reserve Size	予約サイズを定義します。「Reserve Size 接続パラメータ」111 ページを参照してください。
Schema On CE*	Windows CE 上の Ultra Light スキーマのパスとファイル名。「SchemaOnCE 接続パラメータ」103 ページを参照してください。

パラメータ	説明
Schema On Desktop*	Ultra Light スキーマのパスとファイル名。「SchemaOnDesktop 接続パラメータ」104 ページを参照してください。
Schema On Palm*	Palm OS の Ultra Light スキーマ。「SchemaOnPalm 接続パラメータ」105 ページを参照してください。
User ID	データベースへの接続に使用するユーザ ID。「User ID 接続パラメータ」100 ページを参照してください。
VFS On Palm*	仮想ファイル・システムを使用して Palm カードを識別します。「VFSOnPalm パラメータ」105 ページを参照してください。

ファイル・パスの指定

接続パラメータで指定するファイル名とパスは、使用する Ultra Light コンポーネントに応じて、次の稼働条件を満たす必要があります。

ターゲット・プラットフォーム	稼働条件
Java	すべてのバックスラッシュ (円記号) をエスケープする。例：`"file_name=¥¥UltraLite¥¥MyFile.udb"`
Windows CE	絶対パス
Windows	絶対パスまたは相対パス

接続パラメータの指定

各接続パラメータは、次の方法で指定できます。

- **Connection Parameters オブジェクトのプロパティとして指定**
  次のインタフェースは、Connection Parameters オブジェクトを提供します。このオブジェクトには、個々の接続パラメータであるプロパティがあります。

- Ultra Light for MobileVB Ultra Light コントロールをフォームに追加し、そのプロパティを [プロパティ] ウィンドウで指定して、接続パラメータを指定できます。
- Ultra Light ActiveX
- Native Ultra Light for Java
- Ultra Light.NET。Ultra Light.NET コントロールをフォームに追加し、そのプロパティを [プロパティ] ウィンドウで指定して、接続パラメータを指定できます。

- **AdditionalParms プロパティに指定** Connection Parameters オブジェクトを提供するインタフェースには、Additional Parms プロパティがあります。このプロパティは、接続文字列を値として取ります。
- **接続文字列のキーワードとして指定** Ultra Light インタフェースは、接続時に接続文字列を提供できます。接続文字列内のキーワードは、個々の接続パラメータです。
- **UL_STORE_PARMS マクロに指定** Embedded SQL と静的型 C++ API は両方とも UL_STORE_PARMS マクロを使用して、データベース・ファイルに影響する接続パラメータを保持します。これらのパラメータは、データベースが作成されるときの初期の接続試行時にのみ使用されます。
  UL_STORE_PARMS マクロは、接続文字列を取ります。

  たとえば、次の定義は接続パラメータを設定します。

```
#define UL_STORE_PARMS     UL_TEXT(
"reserve_size=2m" )
```

可能であれば、Connection Parameters オブジェクトの使用をおすすめします。このオブジェクトを使用すると、接続文字列を使用するよりも検査が簡単になり、インタフェースがより体系的になります。

接続文字列と接続パラメータ

あまり一般的に使用されない一部のパラメータは、接続文字列でのみ指定できます。接続文字列を AdditionalParms プロパティ、UL_STORE_PARMS マクロ、または引数として接続文字列を取る Open メソッドのいずれに指定できるかは、インタフェースによって異なります。

パラメータがプロパティと接続文字列の両方に指定される場合、プロパティの値が優先されます。

トラブルシューティングのヒント

Ultra Light では、認識されていない接続文字列パラメータは無視されます。その結果、スペリングの誤りがすぐに明らかにならない場合があります。

データベース識別パラメータ

この項の接続パラメータを使用して Ultra Light データベースを識別します。これらのパラメータのうち少なくとも 1 つを接続試行ごとに指定してください。

データベース識別パラメータは、データベースを削除するときにも使用されます。

関連項目

- **Ultra Light for MobileVB**『Ultra Light for MobileVB ユーザーズ・ガイド』>「OpenConnection メソッド」と『Ultra Light for MobileVB ユーザーズ・ガイド』>「OpenConnectionWithParms メソッド」を参照してください。
- **Ultra Light ActiveX**『Ultra Light ActiveX ユーザーズ・ガイド』>「OpenConnection メソッド」と『Ultra Light ActiveX ユーザーズ・ガイド』>「OpenConnectionWithParms メソッド」を参照してください。
- **Native Ultra Light for Java**『Native Ultra Light for Java API リファレンス』の「**ianywhere.native_ultralite.DatabaseManager**」を参照してください。
- **Ultra Light.NET**『Ultra Light.NET ユーザーズ・ガイド』>「ULDatabaseManager クラス」(iAnywhere.Data.UltraLite ネームスペース) または『Ultra Light.NET ユーザーズ・ガイド』>「DatabaseManager クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。
- **Ultra Light C++ コンポーネント**『Ultra Light C/C++ ユーザーズ・ガイド』>「クラス UltraLite_DatabaseManager」を参照してください。
- **Ultra Light for M-Business Anywhere**『UltraLite for M-Business Anywhere User's Guide』>「Method openConnection」と『UltraLite for M-Business Anywhere User's Guide』>「Method openConnectionWithParms」を参照してください。

- **Ultra Light for Embedded SQL**『Ultra Light C/C++ ユーザーズ・ガイド』>「SQLCA (SQL Communication Area) の初期化」を参照してください。
- **Ultra Light 静的型 C++ API**『Ultra Light C/C++ ユーザーズ・ガイド』>「データベースへの接続」を参照してください。
- **Ultra Light 静的型 Java**『Ultra Light 静的型 Java ユーザーズ・ガイド』>「JDBC を使用したデータベースへの接続」を参照してください。

AdditionalParms 接続パラメータ

機能

追加接続パラメータの指定を許可します。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	AdditionalParms
Ultra Light ActiveX	AdditionalParms
Ultra Light.NET	AdditionalParms
Native Ultra Light for Java	additionalParms
Ultra Light for M-Business Anywhere	additionalParms

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページと「追加接続パラメータ」107 ページを参照してください。

使用法

一般的にあまり使用されない一部の接続パラメータには、関連付けられているプロパティが Ultra Light コンポーネントにありません。これらのパラメータは、AdditionalParms に接続文字列として指定できます。

値の範囲

接続文字列

デフォルト

なし

参照

「接続パラメータの指定」86 ページ

DatabaseOnCE 接続パラメータ

機能

Windows CE で接続する Ultra Light データベース・ファイルのパスとファイル名です。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	DatabaseOnCE
Ultra Light ActiveX	DatabaseOnCE
Ultra Light.NET	DatabaseOnCE
Native Ultra Light for Java	databaseOnCE
Ultra Light for M-Business Anywhere	databaseOnCE
接続文字列	**ce_file**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲

文字列

デフォルト

Database On CE が指定されていない場合は、Database On Desktop の値が使用されます。

どちらも指定されていない場合は、デフォルト値の *¥UltraLiteDB¥ulstore.udb* が使用されます。デフォルトの動作に頼らずに、パラメータを明示的に指定することをおすすめします。

説明

データベースの作成時に、このパラメータによって新しいデータベース・ファイルの名前が付けられます。

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。

- ファイル名に拡張子がない場合は、拡張子が *.udb* のファイルと見なされます。
- ファイルのフル・パスを指定してください。この値を使用したファイル名の置換は行われません。

- *.udb* ファイルがすでに存在する場合、スキーマ・ファイルは必要ありません。
- データベースにデフォルト以外の名前を使用する場合、Database On CE が必要です。
- 接続パラメータを使用する場合、このディレクトリが存在することを確認してください。ディレクトリは自動的には作成されません。

例

サンプル・データベース *udemo.udb* を作成して接続するには、次のようにします。

```
"schema_file=MyOrders.usm;CE_FILE=udemo.udb"
```

参照

「ファイル・パスの指定」86 ページ

DatabaseOnDesktop 接続パラメータ

機能

デスクトップ開発環境における接続先のデータベース・ファイルです。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	DatabaseOnDesktop
Ultra Light ActiveX	DatabaseOnDesktop
Ultra Light.NET	DatabaseOnDesktop
Native Ultra Light for Java	databaseOnDesktop
Ultra Light for M-Business Anywhere	databaseOnDesktop
接続文字列	{ **file_name** | **DBF** }

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲

文字列

デフォルト

デフォルト値は *ulstore.udb* です。

デフォルトの動作に頼らずに、パラメータを明示的に指定することをおすすめします。

説明

データベースの作成時に、このパラメータによって新しいデータベース・ファイルの名前が付けられます。

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。

ファイル名に拡張子がない場合は、拡張子が *.udb* のファイルと見なされます。

例

- ディレクトリ *c:¥mydb* にインストールされるサンプル・データベース *udemo.udb* を作成して接続するには、次の接続文字列を使用します。

```
"schema_file=MyOrders.usm;DBF=c:¥mydb¥udemo.udb"
```

参照

「ファイル・パスの指定」86 ページ

DatabaseOnPalm 接続パラメータ

機能

接続するデータベースの Palm 作成者 ID です。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	DatabaseOnPalm
Ultra Light for M-Business Anywhere	databaseOnPalm
接続文字列	**palm_db**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲

文字列

デフォルト

C++ の場合は、アプリケーションの作成者 ID。MobileVB アプリケーションの場合、デフォルト値はありません。

説明

多くのアプリケーションでは、DatabaseOnPalm 接続パラメータを指定する必要はありません。Ultra Light for MobileVB から、アプリケーションの作成者 ID と一致する DatabaseOnPalm 接続パラメータを指定します。

Ultra Light は、最初にデータベースに接続するときに、ul_udb_*creator-id* という名前で *creator-id* という作成者 ID を持つデータベースを作成します。この場合、*creator-id* は、Database On Palm 接続パラメータの値か、(このパラメータが指定されていない場合は) アプリケーションの作成者 ID です。

参照

- 「ファイル・パスの指定」86 ページ
- 「Palm 作成者 ID の概要」242 ページ

OpenConnection パラメータ

OpenConnection パラメータは、OpenConnection メソッドと OpenConnectionWithParms メソッドによってデータベース識別パラメータとともに使用されます。

関連項目

- **Ultra Light for MobileVB**『Ultra Light for MobileVB ユーザーズ・ガイド』>「OpenConnection メソッド」と『Ultra Light for MobileVB ユーザーズ・ガイド』>「OpenConnectionWithParms メソッド」を参照してください。
- **Ultra Light ActiveX**『Ultra Light ActiveX ユーザーズ・ガイド』>「OpenConnection メソッド」と『Ultra Light ActiveX ユーザーズ・ガイド』>「OpenConnectionWithParms メソッド」を参照してください。
- **Native Ultra Light for Java**『Native Ultra Light for Java API リファレンス』の「**ianywhere.native_ultralite.DatabaseManager**」を参照してください。
- **Ultra Light.NET**『Ultra Light.NET ユーザーズ・ガイド』>「ULConnectionParms クラス」(iAnywhere.Data.UltraLite ネームスペース) または『Ultra Light.NET ユーザーズ・ガイド』>「ConnectionParms クラス」(iAnywhere.UltraLite ネームスペース) を参照してください。
- **Ultra Light C++ コンポーネント**『Ultra Light C/C++ ユーザーズ・ガイド』>「OpenConnection 関数」を参照してください。
- **Ultra Light for M-Business Anywhere**『UltraLite for M-Business Anywhere User's Guide』>「Method openConnection」と『UltraLite for M-Business Anywhere User's Guide』>「Method openConnectionWithParms」を参照してください。
- **Ultra Light for Embedded SQL**『Ultra Light C/C++ ユーザーズ・ガイド』>「ユーザの認証」を参照してください。
- **Ultra Light 静的型 C++ API**『Ultra Light C/C++ ユーザーズ・ガイド』>「ユーザの認証」を参照してください。

- **Ultra Light 静的型 Java**『Ultra Light 静的型 Java ユーザーズ・ガイド』>「アプリケーションへのユーザ認証の追加」を参照してください。

CacheSize 接続パラメータ

機能

データベース・キャッシュのサイズを定義します。

構文

インタフェース	接続パラメータ
接続文字列	**cache_size**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

使用法

データベースを設定するときに使用します。k か K、m か M を使用して、キロバイトまたはメガバイトを示します。

値の範囲

最小キャッシュ・サイズは 4 K です。

デフォルト

デフォルトは 16 * page_size です。指定した値は、page_size の倍数に最も近い値に切り捨てられて適用されます。

説明

キャッシュのサイズを定義します。サイズはバイト単位で指定します。キロバイトの単位を示すにはサフィックス k または K を使用し、メガバイトの単位を示すにはサフィックス M または m を使用します。

デフォルトのキャッシュ・サイズは 16 ページです。デフォルトのページ・サイズは 4 K なので、デフォルトのキャッシュ・サイズは 64 K です。最小キャッシュ・サイズは、プラットフォームによって異なります。

デフォルトのキャッシュ・サイズは小さめの値です。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。

キャッシュ・サイズをデータベース自体のサイズより大きくしても、パフォーマンスは向上しません。また、キャッシュ・サイズを大きくすると、使用できる他のアプリケーションの数に影響する場合があります。

Palm Computing Platform では、このパラメータは仮想ファイル・システム (VFS) データベースだけに適用されます。キャッシュ自体は、VFS 記憶装置ではなくレコード記憶装置に常駐します。

例

たとえば、次の文字列では、128 K のキャッシュ・サイズが設定されます。

```
"cache_size=128k"
```

ConnectionName 接続パラメータ

機能

接続の名前を指定します。接続名が必要となるのは、データベースとの接続を複数作成する場合だけです。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	ConnectionName
Ultra Light ActiveX	ConnectionName
Ultra Light.NET	ConnectionName
Native Ultra Light for Java	connectionName
Ultra Light for M-Business Anywhere	connectionName
接続文字列	**con**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

使用法

ConnectionName 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。

EncryptionKey 接続パラメータ

機能

データベースの暗号化キー。CreateDatabase を呼び出すときに、Ultra Light データベースの暗号化キーを定義できます。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	EncryptionKey
Ultra Light ActiveX	EncryptionKey
Ultra Light.NET	EncryptionKey
Native Ultra Light for Java	encryptionKey
Ultra Light for M-Business Anywhere	encryptionKey
接続文字列	{ **key** | **dbkey** }

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲

文字列

デフォルト

キーは指定されません。

説明

データベースの作成時のみ使用します。

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 128 ビット・アルゴリズムを使用して高度に暗号化できます。このアルゴリズムは、Adaptive Server Anywhere データベースの暗号化に使用するアルゴリズムと同じです。高度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る可能性があります。

例

```
"schema_file=MyOrders.usm;KEY=MyKey"
```

参照

「Ultra Light データベースの暗号化」49 ページ

Password 接続パラメータ

機能

ユーザのパスワード。データベースの大文字と小文字を区別しない場合は、パスワードの大文字と小文字を区別せず、データベースの大文字と小文字を区別する場合は、パスワードの大文字と小文字も区別します。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	Password
Ultra Light ActiveX	Password
Ultra Light.NET	Password
Native Ultra Light for Java	password
Ultra Light for M-Business Anywhere	password
接続文字列	{ **password** | **PWD** }

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

使用法

特に制限なし

値の範囲

文字列

デフォルト

SQL

説明

すべてのデータベース・ユーザにはパスワードがあります。パスワードをユーザに提供し、データベースへの接続が許可されるようにしてください。

Password (PWD) 接続パラメータは暗号化されていません。

例

- 次の接続文字列フラグメントは、ユーザ ID DBA とパスワード SQL を指定します。

```
"UID=DBA;PWD=SQL;schema_file=MyOrders.usm"
```

User ID 接続パラメータ

機能

データベースにログオンするユーザ ID です。値には、データベースで認証されたユーザを指定します。データベースの大文字と小文字を区別しない場合は、ユーザ ID の大文字と小文字を区別せず、データベースの大文字と小文字を区別する場合は、ユーザ ID の大文字と小文字も区別します。

データベースは、1 人の認証済みユーザ DBA で作成されます。このユーザの最初のパスワードは SQL です。デフォルトでは、UID=DBA と PWD=SQL で接続が確立します。デフォルト・ユーザを無効にするには、次のように入力します。

```
connection.revokeConnectionFrom.
```

ユーザを追加したりユーザのパスワードを変更するには、次のように入力します。

```
connection.grantConnectTo.
```

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	UserID
Ultra Light ActiveX	UserID
Ultra Light.NET	UserID
Native Ultra Light for Java	userID
Ultra Light for M-Business Anywhere	userID
接続文字列	{ **userid** | **UID** }

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

使用法

特に制限なし

値の範囲

文字列

デフォルト

DBA

説明

データベースに接続するとき、デフォルトのユーザ ID DBA とパスワード SQL をデータベースに残している場合を除いて、必ずユーザ ID を指定します。

例

- 次の接続文字列フラグメントは、ユーザ ID DBA とパスワード SQL を指定します。

```
"schema_file=MyOrders.usm;uid=DBA;pwd=SQL"
```

データベース・スキーマ・パラメータ

この項のキーワードを使用して、Ultra Light データベースのスキーマを指定します。

関連項目

- **Ultra Light for MobileVB**『Ultra Light for MobileVB ユーザーズ・ガイド』>「CreateDatabase メソッド」と『Ultra Light for MobileVB ユーザーズ・ガイド』>「CreateDatabaseWithParms メソッド」を参照してください。
- **Ultra Light ActiveX**『Ultra Light ActiveX ユーザーズ・ガイド』>「CreateDatabase メソッド」と『Ultra Light ActiveX ユーザーズ・ガイド』>「CreateDatabaseWithParms メソッド」を参照してください。
- **Native Ultra Light for Java**『Native Ultra Light for Java API リファレンス』の「**ianywhere.native_ultralite.DatabaseManager**」を参照してください。
- **Ultra Light.NET**『Ultra Light.NET ユーザーズ・ガイド』>「OpenWithCreate メソッド」(iAnywhere.Data.UltraLite ネームスペース) または『Ultra Light.NET ユーザーズ・ガイド』>「CreateDatabase メソッド」(iAnywhere.UltraLite ネームスペース) を参照してください。
- **Ultra Light C++ コンポーネント**『Ultra Light C/C++ ユーザーズ・ガイド』>「CreateAndOpenDatabase 関数」を参照してください。
- **Ultra Light for M-Business Anywhere**『UltraLite for M-Business Anywhere User's Guide』>「Method createDatabase」と『UltraLite for M-Business Anywhere User's Guide』>「Method createDatabaseWithParms」を参照してください。
- **Ultra Light for embedded SQL**『Ultra Light C/C++ ユーザーズ・ガイド』>「Ultra Light C/C++ アプリケーションのマクロとコンパイラ指令」を参照してください。

- **Ultra Light 静的型 C++**『Ultra Light C/C++ ユーザーズ・ガイド』> 「Ultra Light C/C++ アプリケーションのマクロとコンパイラ指令」を参照してください。

SchemaOnCE 接続パラメータ

機能

Windows CE 上に配備されているスキーマ・ファイル名を識別します。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	SchemaOnCE
Ultra Light ActiveX	SchemaOnCE
Ultra Light.NET	SchemaOnCE
Native Ultra Light for Java	schemaOnCE
Ultra Light for M-Business Anywhere	schemaOnCE
接続文字列	**ce_schema**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲

文字列

デフォルト

推奨するファイル拡張子は *.usm* です。

説明

データベースの作成時のみ使用します。

WindowsCE 上の Ultra Light スキーマ・ファイルのパスとファイル名。Ultra Light スキーマ・ファイルのデフォルトの拡張子は .usm です。これは、CE に対して CreateDatabase を使用する場合の必須パラメータです。

例

- 次の接続文字列フラグメントは、ce_schema パラメータと schema_file パラメータを指定します。デスクトップで実行する場合は、*MyOrders.usm* が使用されます。CE デバイス (エミュレータを含む) で実行する場合は、*orders.usm* が使用されます。

```
"CE_SCHEMA=orders.usm;SCHEMA_FILE=MyOrders.usm"
```

SchemaOnDesktop 接続パラメータ

機能

デスクトップ開発環境のスキーマ・ファイルを識別します。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	SchemaOnDesktop
Ultra Light ActiveX	SchemaOnDesktop
Ultra Light.NET	SchemaOnDesktop
Native Ultra Light for Java	schemaOnDesktop
Ultra Light for M-Business Anywhere	schemaOnDesktop
接続文字列	**schema_file**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲

文字列

デフォルト

推奨するファイル拡張子は *.usm* です。

説明

データベースの作成時のみ使用します。

開発環境の Ultra Light スキーマのパスとファイル名。

例

- 次の接続文字列フラグメントは、ce_schema パラメータと schema_file パラメータを指定します。デスクトップで実行する場合は、*MyOrders.usm* が使用されます。CE デバイス (エミュレータを含む) で実行する場合は、*orders.usm* が使用されます。

```
"CE_SCHEMA=orders.usm;SCHEMA_FILE=MyOrders.usm"
```

SchemaOnPalm 接続パラメータ

機能

Palm OS デバイス上に配備されているスキーマ・ファイルを識別します。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	SchemaOnPalm
Ultra Light for M-Business Anywhere	schemaOnPalm
接続文字列	**palm_schema**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲

文字列

デフォルト

デフォルト値はありません。

説明

このパラメータでは、Palm OS デバイスの Ultra Light スキーマを指定します。データベースを作成するときのみ必要です。

.pdb はデスクトップ上の拡張子ですが、接続パラメータ文字列で *.pdb* を指定しないでください。

例

- 次の接続文字列フラグメントは、palm_schema パラメータと schema_file パラメータを指定します。

```
"PALM_SCHEMA=orders;SCHEMA_FILE=MyOrders.usm"
```

VFSOnPalm パラメータ

機能

仮想ファイル・システムを使用して Palm カードを識別します。

このパラメータは Ultra Light for MobileVB でのみ使用できます。Embedded SQL または静的型 C++ API アプリケーションの仮想ファイル・システムを使用するには、EnablePalmFileDB 関数を使用します。

構文

インタフェース	接続パラメータ
Ultra Light for MobileVB	VFSOnPalm
接続文字列 (Ultra Light for MobileVB)	PALM_FS=VFS

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲

パラメータとしての VFSOnPalm はブール値です。

接続文字列では、このパラメータを次のように指定します。

```
PALM_FS=VFS
```

説明

Palm デバイス用の VFS カードを使用し、データベースをそのカードに格納する場合は、CreateDatabase と OpenConnection の両方で `palm_fs=vfs` パラメータを指定する必要があります。

デフォルトのデータベース・ファイル名は *ul_udb_YYYY.udb* です。この場合、*YYYY* はアプリケーションの作成者 ID です。DatabaseOnPalm パラメータに異なる作成者 ID を指定することによって、ファイル名を制御できます。たとえば、次の接続文字列は、ファイル名が *ul_udb_XXXX.udb* のカード上のデータベースを参照します。

```
palm_db=XXXX;palm_fs=vfs
```

VFSOnPalm パラメータを指定する場合でも、palm_db パラメータ (Database on Palm) を有効な作成者 ID に設定してください。

VFS On Palm パラメータを指定しないと、データベースがカード上ではなくデバイス上に作成 (または削除、接続が確立) されます。

追加接続パラメータ

これらは、データベース作成時にデータベースを設定するためのオプション・パラメータです。これらのパラメータの一部はパフォーマンスに影響するので、これらのパラメータをテストして、使用しているアプリケーションに最適なパフォーマンスを判別することをおすすめします。

関連項目

- **Ultra Light for MobileVB**『Ultra Light for MobileVB ユーザーズ・ガイド』>「CreateDatabase メソッド」と『Ultra Light for MobileVB ユーザーズ・ガイド』>「CreateDatabaseWithParms メソッド」を参照してください。
- **Ultra Light ActiveX**『Ultra Light ActiveX ユーザーズ・ガイド』>「CreateDatabase メソッド」と『Ultra Light ActiveX ユーザーズ・ガイド』>「CreateDatabaseWithParms メソッド」を参照してください。
- **Native Ultra Light for Java**『Native Ultra Light for Java API リファレンス』の「**ianywhere.native_ultralite.DatabaseManager**」を参照してください。
- **Ultra Light.NET**『Ultra Light.NET ユーザーズ・ガイド』>「OpenWithCreate メソッド」(iAnywhere.Data.UltraLite ネームスペース) または『Ultra Light.NET ユーザーズ・ガイド』>「CreateDatabase メソッド」(iAnywhere.UltraLite ネームスペース) を参照してください。
- **Ultra Light C++ コンポーネント**『Ultra Light C/C++ ユーザーズ・ガイド』>「CreateAndOpenDatabase 関数」を参照してください。
- **Ultra Light for M-Business Anywhere**『UltraLite for M-Business Anywhere User's Guide』>「Method createDatabase」と『UltraLite for M-Business Anywhere User's Guide』>「Method createDatabaseWithParms」を参照してください。
- **Ultra Light for embedded SQL**『Ultra Light C/C++ ユーザーズ・ガイド』>「Ultra Light C/C++ アプリケーションのマクロとコンパイラ指令」を参照してください。

- **Ultra Light 静的型 C++**『Ultra Light C/C++ ユーザーズ・ガイド』>「Ultra Light C/C++ アプリケーションのマクロとコンパイラ指令」を参照してください。

DatabaseName 接続パラメータ

機能

複数のデータベースに接続するアプリケーションの場合、このパラメータを使用してデータベースを区別できます。

構文

インタフェース	接続パラメータ
接続文字列	**dbn**

使用法

このパラメータを接続文字列または AdditionalParms 文字列のいずれかに指定します。

データベースが起動されると、データベース名が設定されます。これによって、データベース ファイルを指定しなくてもデータベース名パラメータを使用して新しい接続を確立できるようになります。

デフォルト

デフォルト値は、パスと拡張子を削除してデータベース ファイル名から生成されます。Palm OS の場合、デフォルト値は作成者 ID です。

Obfuscate 接続パラメータ

機能

データベースを読みにくくします。難読化は簡単な暗号化です。

構文

インタフェース	接続パラメータ
接続文字列	**obfuscate**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

値の範囲　**0** または **1**。値が 1 の場合は、データベースを読みにくくすることを示します。

使用法　データベースの作成時のみ使用します。

Embedded SQL と静的型 C++ API の開発者は、UL_ENABLE_OBFUSCATION マクロを使用してデータベースを読みにくくすることもできます。『Ultra Light C/C++ ユーザーズ・ガイド』 > 「UL_ENABLE_OBFUSCATION マクロ」を参照してください。

デフォルト　デフォルトでは、データベースは読みにくくされていません。

参照　「Ultra Light データベースの暗号化」49 ページ

PageSize 接続パラメータ

機能　データベース・ページ・サイズを定義します。

構文

インタフェース	接続パラメータ
接続文字列	**page_size**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

使用法　データベースの作成時のみ使用します。

データベースの作成時に使用します。k または K を使用してキロバイトを示します。

デフォルト　Ultra Light データベースのデフォルト・ページ・サイズは 4 K です。サイズの範囲は 2 K ～ 4 K です。

説明　Ultra Light データベースはページ単位で格納されます。I/O 操作は一度に 1 ページずつ実行されます。すべてのターゲット・プラットフォームで使用できます。ページ・サイズを 2 KB に設定すると、最大テーブル数が約 500 に減少します。

このパラメータは、既存のデータベースの開始時には無視されます。

例

次の保管パラメータ文字列を使用すると 2 KB のページを指定できます。

```
"schema_file=MyOrders.usm;PAGE_SIZE=2K"
```

PalmAllowBackup パラメータ

機能

Palm デバイス上の HotSync のバックアップ動作を制御します。

構文

インタフェース	接続パラメータ
接続文字列	**palm_allow_backup**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

使用法

データベースを設定するときに使用します。

値の範囲

yes または **no**。

説明

バックアップ・ビットが Ultra Light データベースに設定されている場合、また、このパラメータが **yes** に設定されている場合は、デバイスが HotSync を使用して同期されるたびに Palm データベース全体がバックアップされます。このパラメータが設定されていない場合、Ultra Light はバックアップ・ビットがクリアされていることを確認します。ほとんどの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、このパラメータを設定する必要はありません。

バックアップ・ビットは、データベース・ファイルが HotSync によって配備されるときに設定されます。また、ULUtil ユーティリティでも設定できます。詳細については、「ULUtil ユーティリティ」156 ページを参照してください。

例

次の文字列は、このパラメータを設定します。

```
#define UL_STORE_PARMS UL_TEXT("palm_allow_backup=yes")
```

Reserve Size 接続パラメータ

機能

Ultra Light 継続データの保管に使用する、ファイル・システム領域を予約します。

構文

インタフェース	接続パラメータ
接続文字列	**reserve_size**

接続パラメータの使用の詳細については、「接続パラメータの指定」86 ページを参照してください。

使用法

k か K、m か M を使用して、キロバイトまたはメガバイトを示します。

値の範囲

値は KB または MB で表します。

説明

reserve_size パラメータを使用して、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ることができます。ファイル・システムの領域を予約すると、パフォーマンスが多少向上し、メモリが不足するという障害を防ぐことができます。デフォルトでは、永続ストレージ・ファイルのサイズは、アプリケーションがデータベースを更新して、サイズを大きくする必要が生じた場合にだけ大きくなります。

reserve_size で予約されるファイル・システム領域には、ロー・データだけでなく、継続保管ファイルのメタデータも含まれます。データベースのデータ量から、必要なファイル・システム領域を計算する場合は、メタデータのオーバヘッドとデータの圧縮を考慮してください。テスト・データを入れてデータベースを実行し、継続保管ファイルのサイズを確認することをおすすめします。

reserve_size パラメータは、起動時に継続保管ファイルを設定された予約サイズまで大きくすることにより、領域を予約します。これは、そのファイルが以前に存在していたかどうかに関係なく行われます。このファイルはトランケートされません。

このパラメータは、アプリケーションが Virtual File System (VFS) を使用していないかぎり、Palm Computing Platform には適用されません。

例

reserve_size パラメータを使用して、次のように領域を事前に割り付けます。

```
"CE_SCHEMA=orders;RESERVE_SIZE=128K"
```

この例では、起動時に継続保管ファイルのサイズが最低 128 KB 確保されます。

第 5 章

Ultra Light ユーティリティ・リファレンス

この章の内容

この章では、Ultra Light ユーティリティ・プログラムのリファレンス情報について説明します。

Ultra Light エンジン

機能

Ultra Light データベースを管理します。単一アプリケーションのホストとなる Ultra Light ランタイムとは対照的に、Ultra Light エンジンでは、複数のアプリケーションが Ultra Light データベースに同時にアクセスできます。

構文

dbuleng9

適用対象

Windows XP と Windows CE

説明

Ultra Light エンジンのコマンド・ライン・オプションはありません。

参照

- 「Ultra Light エンジンの使用」80 ページ
- 「dbulstop ユーティリティ」129 ページ

Ultra Light ジェネレータ

適用対象　静的インタフェースのみ

機能　Ultra Light ジェネレータは、アプリケーション・データベースを実装し、追加の C/C++ または Java ソース・ファイルを生成します。この追加のソース・ファイルは、コンパイルしてアプリケーションにリンクさせます。

構文　**ulgen** [*options*] [*project* [*output-filename*]]

オプション	説明
-a	SQL 文字列名を大文字にする (Java の場合)
-c "*keyword*=*value*;…"	リファレンス・データベースのデータベース接続パラメータを指定する
-e	SQL 文字列を生成された定数と置換する (Java の場合)
-f *filename*	出力ファイル名を指定する
-g	警告を表示しない
-i	内部クラスを生成する (Java の場合)
-j *project-name*	プロジェクト名
-l *type*	各文の実行プランのログをファイルに取る。次のいずれかの型を指定します。 • xml • short • long
-m *version*	生成される同期スクリプトのバージョン名を指定する
-o *table-name*,…	同期中にテーブルがアップロードされる順序を指定する
-p *package-name*	生成されたクラスのパッケージ名 (Java の場合)

オプション	説明
-q	バナーを表示しない
-r *filename*	信用されたルート証明書を含むファイル
-s *filename*	インタフェース定義内の SQL 文字列のリストを生成する (Java の場合)
-t *target*	ターゲットの言語。次のいずれかであることが必要です。 • c • c++ • java
-u *pub-name*	使用するパブリケーション (C++ API のみ)
-v *pub-name*	同期に使用するパブリケーション
-x	小さい C/C++ ファイルをたくさん生成する

説明

Ultra Light ジェネレータが作成したコードは、ユーザがコンパイルして Ultra Light アプリケーションの一部にします。Ultra Light ジェネレータの出力は、Adaptive Server Anywhere リファレンス・データベースのスキーマと、Embedded SQL ソース・ファイルで使用する特定の SQL 文またはテーブルに基づいています。

すべての文とテーブルが dbo.ul_statement テーブルで定義されていることを確認してから、ジェネレータを実行してください。これは、次のように行います。

- Embedded SQL では、各ファイルで SQL プリプロセッサを実行します。
- C/C++ API と Java では、ul_add_statement を使用してデータベースに文を追加するか、データベースにパブリケーションを定義します。または、この両方を行います。

このテーブルでは、文はプロジェクトと対応しています。ジェネレータのコマンド・ラインでプロジェクト名を指定すると、生成されたデータベースに含まれる文を決定できます。

ジェネレータのコマンド・ラインでは、複数のプロジェクトを指定できます。また、プロジェクトとパブリケーションを混在させることもできます。ジェネレータの実行は、生成されたデータベースのそれぞれに対し 1 回のみにしてください。

出力ファイル名を指定しないと、生成されたコードは *project* という名前のファイルに書き込まれます。-f コマンド・ライン・オプションを使用して出力ファイル名を指定することをおすすめします。

Ultra Light ジェネレータのオペレーションのカスタマイズ　Ultra Light アナライザには、コード生成処理のカスタマイズに使用できるようにフックが用意されています。これらのフックは、ストアド・プロシージャ名です。次の名前を持つストアド・プロシージャを指定すると、Ultra Light アナライザは分析処理の前後にそのストアド・プロシージャを呼び出します。

- **sp_hook_ulgen_begin( )**
- **sp_hook_ulgen_end( )**

これらのフックはリファレンス・データベース内で定義され、アナライザの分析フェーズ中にだけ使用されます。フックを次のように作成できます。

```
CREATE PROCEDURE sp_hook_ulgen_begin ()
 BEGIN
 // actions here
 END
 CREATE PROCEDURE sp_hook_ulgen_end ()
 BEGIN
 // actions here
 END
```

オプション

project　生成されたデータベースに含める一連の文を決定するプロジェクト名です。ファイル名をより正確に指定するには、-j オプションを使用します。

output-filename　生成されたファイルの名前です。拡張子は付けません。ファイル名をより正確に指定するには、-f オプションを使用します。

Java では、この名前はデータベースの名前でもあり、接続のときに指定します。

-a Java アプリケーションを開発している場合は、プロジェクト内の SQL 文の名前が定数としてアプリケーションで使用されます。規則により、定数は大文字です。単語と単語の間にはアンダースコア文字が使用されます。-a オプションは、SQL 文の名前をこの規則に合ったものにします。文字を大文字にし、元の名前の大文字の前にアンダースコアまたは大文字がなければ、アンダースコアを挿入します。たとえば、`MyStatement` 文は `MY_STATEMENT` になります。また、`AStatement` 文は `ASTATEMENT` になります。

生成された名前では、アルファベット以外の文字とスペースは、-a オプションを使用しなくてもアンダースコアに置換されます。

-c connection-string ここで接続文字列を指定することによって、ジェネレータにリファレンス・データベースの読み込みと修正のためのパーミッションが与えられます。このパラメータは必須です。

-e 生成されたデータベースの SQL 文字列は、より小さい、生成された文字列に置換されます。このオプションは、大量の文を含むデータベースの容量削減に役立ちます。

-f filename 出力ファイルの指定にはこの方法をおすすめします。拡張子は指定しません。

-g 警告メッセージを表示しません。エラー・メッセージは引き続き表示されます。

Ultra Light ジェネレータは、状況によっては、生成されたコードがエラーになる場合があることを示す警告を提供します。たとえば、SQL 文にテンポラリ・テーブルが含まれていると、警告が生成されます。

-I デフォルトでは、生成されたクラスは、メイン・データベース・クラスでの場合を除き、トップレベルの非パブリック・クラスとして書き込まれます。`-I` を使用すると、生成されたクラスは内部クラスとして書き込まれます。このオプションを使用する場合は、内部クラスを正しくコンパイルできる Java コンパイラを使用してください。

-j project-name プロジェクトの指定にはこの方法をおすすめします。このオプションを使用して、次のように複数のプロジェクトを指定できます。

```
ulgen -j project1 -j project2 ...
```

-l type　アプリケーション内でクエリの実行プランのログを取ります。これらのプランは Interactive SQL で表示できます。使用可能な型は、次のとおりです。

- **xml**　XML フォーマットで記述します。Interactive SQL の [ファイル] ー [開く] を選択し、プランを表示します。
- **short**　プランの簡潔な説明を *<statement>.txt* という名前のファイルに記述します。内容は、EXPLANATION 関数によって生成されます。
- **long**　プランの詳細な説明を *<statement>.txt* という名前のファイルに記述します。内容は、PLAN 関数によって生成されます。

-m version　生成される同期スクリプトのバージョン名を指定します。生成された同期スクリプトを Mobile Link 統合データベースに使用すると簡単に同期できます。デフォルト値は **ul_default** です。

-o table-name,…　同期中にテーブルがアップロードされる順序を指定します。このオプションを使用すると、アップロード中の参照整合性エラーを避けることができます。アップロードされる各テーブルを一度だけ指定してください。

-p package-name　Java 出力の生成時に生成されるファイルに付けるパッケージ名を指定します。

-q　出力メッセージを表示しません。

-r filename　信用されたルート証明書を含むファイルです。これは、Certicom セキュリティ・ソフトウェアを使用したセキュリティ機能のある同期の確保に使用されます。

ジェネレータは、これらの信用されたルートを Ultra Light アプリケーションに埋め込みます。アプリケーションは、証明書のチェーンを Mobile Link 同期サーバから受け取ると、そのルートが信用されたルートの中にあるかどうかを確認し、ある場合のみ接続を受け入れます。

ジェネレータは、信用されたルート証明書ファイル内のすべての証明書の有効期限を確認し、有効期限の残りが 6 か月 (180 日) 未満の証明書に次の警告を発行します。

警告：証明書はあと %1 日で失効します。

すでに有効期限が切れている証明書には、「証明書が失効しました」というエラーを発行します。

Palm OS 以外のプラットフォームでは -r を使用する代わりに、証明書を別に指定して trusted_certificates セキュリティ・パラメータによってそれに対応します。詳細については、『Mobile Link クライアント』>「trusted_certificates」を参照してください。

詳細については、『Mobile Link クライアント』>「Ultra Light 同期パラメータ」と『Mobile Link 管理ガイド』>「Mobile Link トランスポート・レイヤ・セキュリティ」を参照してください。

-s filename　SQL 文を定数として持つインタフェースを生成します。このオプションは、Java の場合にのみ使用されます。インタフェース・ファイルのフォーマットは、次の例と同じようなフォーマットです。

```
package com.sybase.test;
 public interface EmpTestSQL {
     String EMPLOYEE = "select emp_fname, emp_lname
         from employee where emp_id = ?";
     String UPDATE_EMPLOYEE = "update employee
         set emp_fname = ?, emp_lname = ?
         where emp_id = ?";
 }
```

filename には拡張子 *.java* を指定しません。-a オプションは、文の名前の大文字と小文字を制御します。

-t target　生成されるファイルの種類と拡張子を指定します。

- Java を使用している場合は、*target* に **java** を指定します。Embedded SQL または C++ API を使用している場合は、*target* に **c** または **c++** を指定します。ファイル名の拡張子は、C++ API または Embedded SQL のどちらを使用しているかにかかわらず、target によって決まります。
- **c++** を指定した場合は、次のファイルが生成されます。
    - **filename.cpp**　生成された API のコード。

- **filename.h**　ヘッダ・ファイル。このファイルは見る必要はありません。
- **filename.hpp**　使用しているアプリケーションの C++ API 定義。

- *target* に **c** を指定した場合は、*filename.c* が生成されます。

-u pub-name　パブリケーションの C++ API を生成している場合は、-u オプションを使用してパブリケーション名を指定します。

-v pub-name　同期させるパブリケーションを指定します。パブリケーションを使用して同期対象の変更を定義しない場合は、すべての変更が同期します。

パブリケーションで指定したカラムまたはテーブルがアプリケーション内の SQL 文で参照されない場合は、Ultra Light データベースには組み込まれません。

複数のパブリケーションを指定するには、繰り返し -v オプションを使用します。次に例を示します。

```
ulgen -v pub1 -v pub2 ...
```

パブリケーションの最大数は 32 です。

詳細については、『Mobile Link クライアント』> 「Ultra Light クライアント」を参照してください。

-x　このオプションは、生成されたコードを含むファイルが大きすぎて C/C++ コンパイラでコンパイルできない場合に使用することを意図しています。

このオプションを使用すると、Ultra Light ジェネレータは小さいファイルをたくさん生成します。-x が使用されている場合、Ultra Light ジェネレータはデータベース用に 1 つと各 SQL 文用に 1 つの C/C++ ファイルを書き出します。

このオプションは、Java コードの生成時には影響しません。

SQL プリプロセッサ

適用対象

Embedded SQL 静的開発モデルのみ

機能

SQL プリプロセッサは、コンパイラが実行される前に、Embedded SQL を含む C または C++ プログラムを処理します。

構文

sqlpp [*options*] *sql-filename* [*output-filename*]

オプション	説明
-c "*keyword=value*;…."	リファレンス・データベースのデータベース接続パラメータを指定する
-d	小さいデータ・サイズを優先するコードを生成する
-e *level*	SQL 構文に準拠しないものをエラーとして通知する
-g	Ultra Light の警告を表示しない
-h *line-width*	出力する行の長さの最大値を制限する
-k	SQLCODE のユーザ宣言をインクルードする
-m *version*	生成される同期スクリプトのバージョン名を指定する
-n	行番号
-o *operating-sys*	ターゲット・オペレーティング・システム。WIN32、WINNT、NETWARE、UNIX のいずれか。
-p *project-name*	Ultra Light プロジェクト名
-q	クワイエット・モード (バナーを表示しない)
-s *string-len*	コンパイラに与える最大文字列の長さ
-w *level*	SQL 構文に準拠しないものを警告として通知する
-x	マルチバイト SQL 文字列をエスケープ・シーケンスに変更する
-z *sequence*	照合順を指定する

参照　『ASA プログラミング・ガイド』> 「概要」

説明　SQL プリプロセッサは、コンパイラが実行される前に、Embedded SQL を含む C または C++ ソース・ファイルを処理します。このプリプロセッサは、*input-file* の SQL 文を C/C++ に変換します。結果は *output-file* 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。デフォルトの出力ファイル名は拡張子 *c* が付いた *SQL-filename* ベースの名前です。ただし、*SQL-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。

SQL プリプロセッサは、Ultra Light アプリケーションの一部であるファイルを前処理しているときに、Adaptive Server Anywhere リファレンス・データベースにアクセスする必要があります。**–c** オプションを使用してリファレンス・データベースの接続パラメータを指定してください。

プロジェクト名を指定しない場合は、SQL プリプロセッサは Ultra Light ジェネレータも実行し、追加のコードを生成された C/C++ ソース・ファイルに追加します。このコードには、ユーザのデータベース・スキーマの C/C++ 言語記述とアプリケーションへの SQL 文の実装が含まれます。

Ultra Light ジェネレータのオペレーションのカスタマイズ　Ultra Light アナライザには、コード生成処理のカスタマイズに使用できるようにフックが用意されています。これらのフックは、ストアド・プロシージャ名です。次の名前を持つストアド・プロシージャを指定すると、Ultra Light アナライザは分析処理の前後にそのストアド・プロシージャを呼び出します。

- **sp_hook_ulgen_begin( )**
- **sp_hook_ulgen_end( )**

これらのフックはリファレンス・データベース内で定義され、アナライザの分析フェーズ中にだけ使用されます。フックを次のように作成できます。

```
CREATE PROCEDURE sp_hook_ulgen_begin ()
 BEGIN
 // actions here
 END
 CREATE PROCEDURE sp_hook_ulgen_end ()
 BEGIN
 // actions here
 END
```

オプション

-c Ultra Light アプリケーションの一部であるファイルの前処理を実行するときに必要です。ここで接続文字列を指定することにより、SQL プリプロセッサは読み込みと修正のためにリファレンス・データベースにアクセスできるようになります。

-d データ領域サイズを減らす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。これはコード・サイズを増加させます。

-e 指定した SQL/92 のセットに含まれない Embedded SQL をエラーとして通知します。このオプションは、Ultra Light には適用されません。

次に、設定できる *level* の値とその意味を示します。

- **e** 初級レベルの SQL/92 構文ではない構文を通知します。
- **i** 中級レベルの SQL/92 構文ではない構文を通知します。
- **f** 上級レベルの SQL/92 構文では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 がサポートしない機能を通知します。
- **w** サポートされているすべての構文を許容します。

-g Ultra Light のコード生成に固有の警告を表示しません。

-h num *sqlpp* が出力する行の最大長を NUM 文字に制限します。行の内容が次の行に続くことを表す文字は円記号 (¥) です。また、NUM に指定できる最小値は 10 です。

-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。

-m version　生成される同期スクリプトのバージョン名を指定します。生成された同期スクリプトを Mobile Link 統合データベースに使用すると簡単に同期できます。デフォルト値は **ul_default** です。

-n　C ファイルに行番号情報を生成します。これは、生成された C コード内の適切な場所にある *#line* 指令で構成されます。使用中のコンパイラが *#line* 指令をサポートしている場合、このオプションを指定すると、コンパイラは **SQL-filename** の行番号を使用してエラーをレポートし、C/C++ 出力ファイルの行番号は使用しません。また、*#line* 指令はソースレベルのデバッガで間接的に使用されるので、**SQL-filename** を表示しながらデバッグできます。

-o　ターゲット・オペレーティング・システムを指定します。このオプションが、プログラムを実行するオペレーティング・システムと一致するように注意してください。プログラム内に、特殊記号への参照が生成されます。この記号はインタフェース・ライブラリで定義されます。適切でないオペレーティング・システムを指定したり、適切でないライブラリを使用したりすると、リンカがエラーを検知します。サポートされているオペレーティング・システムは次のとおりです。

- **WIN32**　Microsoft Windows 95/98/Me と Windows CE
- **WINNT**　Microsoft Windows NT/2000/XP

-p project-name　Embedded SQL ファイルが属する Ultra Light プロジェクトを識別します。Ultra Light アプリケーションの一部であるファイルを処理する場合にのみ適用します。

-q　クワイエット・モードで作動します。バナーを表示しません。

-s string-len　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。

-w level　指定した SQL/92 のセットに含まれない Embedded SQL を、警告として通知します。このオプションは、Ultra Light には適用されません。

次に、設定できる *level* の値とその意味を示します。

- **e**　初級レベルの SQL/92 構文ではない構文を通知します。
- **i**　中級レベルの SQL/92 構文ではない構文を通知します。
- **f**　上級レベルの SQL/92 構文では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 がサポートしない機能を通知します。
- **w**　サポートされているすべての構文を許容します。

-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 sequence　照合順またはファイル名を指定します。推奨する照合順のリストを表示するには、コマンド・プロンプトで **dbinit −l** と入力してください。

HotSync コンジット・インストーラ

機能

このユーティリティは、使用しているマシンでの HotSync コンジットのインストールまたは削除に使用します。

構文

dbcond9 [*options*] *id*

オプション	説明
id	コンジットを使用するアプリケーションの作成者 ID
-n *name*	HotSync マネージャが表示する名前
-k *key*	デフォルトの暗号化キー
-p *parms*	デフォルトのクライアント同期パラメータ
-x	指定した作成者 ID のコンジットを削除する

説明

このユーティリティは、使用しているマシンでの HotSync コンジットのインストールに使用します。HotSync コンジット・インストーラを実行するには、HotSync マネージャが必要です。

オプション

id　コンジットを使用するアプリケーションのユーザ ID。指定した *creatorID* 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。このオプションは必須です。

-k key　デフォルトの暗号化キー。空の文字列 (`-k ""`) を指定し、既存の暗号化キーをクリアします。

-n name　HotSync マネージャが表示する名前。これは、コンジットがデータを格納するサブディレクトリの名前でもあります。このオプションを `-x` と一緒に使用しないでください。デフォルト値は、**MobiLink conduit** です。

-p parms　デフォルトのクライアント同期パラメータ。空の文字列 (`-p ""`) を指定し、既存のパラメータをクリアします。

-x　指定した *creatorID* のコンジットを削除します。`-x` を指定しなければ、コンジットがインストールされます。

例

次のコマンド・ラインは、作成者 ID が Syb2 である CustDB サンプル・アプリケーション用のコンジットをインストールします。

```
dbcond9 -n CustDB Syb2
```

dbulstop ユーティリティ

機能　Ultra Light エンジンを停止します。

構文　**dbulstop**

適用対象　Windows XP と Windows CE

説明　Ultra Light エンジンの stop ユーティリティのコマンド・ライン・オプションはありません。

参照

- 「Ultra Light エンジンの使用」80 ページ
- 「Ultra Light エンジン」114 ページ

ulconv ユーティリティ

機能

このユーティリティは、フォーマット間で Ultra Light データベースとスキーマ・ファイルを変換します。

構文

ulconv create [*options*]

ulconv { **load** | **unload** } [*options*] *xml-file*

ulconv sync [*options*] [*sync-parms*]

これらのオプションは、使用している構文によって異なります。このユーティリティのコマンド・ライン・オプションでは、大文字と小文字が区別されます。

- **create** 指定されたスキーマ・ファイルから新しい空の Ultra Light データベースを作成します。次のオプションがサポートされています。

オプション	説明
-c "*keyword=value;…*"	Ultra Light データベース接続パラメータ。必須。
-m	マルチバイトの Ultra Light データベースを作成する
-p *creator-id*	指定された作成者 ID を使用して Palm OS データベースを作成する。データベース・ファイルには、ul_udb_*creator-id* の Palm OS PDB 名が付けられます。 作成者 ID の詳細については、「Palm 作成者 ID の概要」242 ページを参照してください。
-q	メッセージまたはバナーを表示しない
-S *schema-file*	指定されたスキーマ・ファイルを使用して、データベース・スキーマを定義する
-u	ユニコードの Ultra Light データベースを作成する。デフォルト動作。
-v	冗長メッセージを表示する

オプション	説明
-y	データベース・ファイルが存在する場合、それを上書きする

- **load** XML ファイルを読み込み、データを新規または既存のデータベースにロードします。次のオプションがサポートされています。

オプション	説明
xml-file	ロードするデータが格納された XML ファイル
-c "*keyword=value;…*"	Ultra Light データベース接続パラメータ。必須。
-d	既存のデータベースを開けない場合のみ、新しいデータベースを作成する。既存のデータベースを開けない場合、XML ファイルのスキーマ定義は無視します。 **-d** が指定されていない場合、新しいデータベースが作成されます。

-D *directory*	外部データが格納されたファイルのディレクトリを指定する。
	
	xml-file には、次のように、外部ファイルの long binary と long character カラムのカラム・データが含まれることがあります。
	
	`<uldata>`
	`  <table name="table-name">`
	`  <row`
	`  longData.File="column.dat"`
	`.../>`
	`  ...`
	`  </table>`
	`  ...`
	`  </uldata>`
	
	-D が指定されている場合、column.dat のパスは、指定されたディレクトリに関連付けられていると解釈されます。-D が指定されていない場合、パスは *xml-file* に関連付けられています。

オプション	説明
-i	挿入されたローを次の同期時にアップロードする。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。
-m	マルチバイトの Ultra Light データベースを作成する
-n	スキーマのみをロードする。データは無視されます。
-p *creator-id*	指定された作成者 ID を使用して Palm OS データベースを作成する
-q	メッセージまたはバナーを表示しない
-S *schema-file*	指定されたスキーマ・ファイルにスキーマを保存する
-u	ユニコードの Ultra Light データベースを作成する。デフォルト動作。
-v	冗長メッセージを表示する
-y	データベース・ファイルが存在する場合、それを上書きする

- **sync** 指定された同期パラメータを使用して、Ultra Light データベースを開いて同期する。次のオプションがサポートされています。

オプション	説明
sync-parms	同期を管理する、セミコロンで区切られた *keyword=value* ペアのリスト。 キーワードは、大文字と小文字の区別のない次のリストから選択します。 ◆ downloadOnly (ブール値) ◆ newpasswd (文字列) ◆ passwd (文字列) ◆ publicationMask (整数) ◆ sendColumnNames (ブール値) ◆ uploadOnly (ブール値) ◆ username (文字列) ◆ version (文字列) 同期パラメータの詳細については、『Mobile Link クライアント』> 「Ultra Light 同期パラメータ」を参照してください。
-a *auth-parm*	Mobile Link 認証パラメータ。複数の **-a** *auth-parm* オプションを使用して複数の認証パラメータを指定できます。 これらのパラメータは、コマンド・ラインに表示される順序で送信されます。 Mobile Link 認証パラメータについては、『Mobile Link 管理ガイド』> 「authenticate_parameters 接続イベント」を参照してください。
-c "*keyword=value*;…"	Ultra Light データベース接続パラメータ。必須。

オプション	説明
-k *stream-type*	使用する同期ストリーム。 *stream-type* は、次のいずれかの値である必要があります。これらは大文字と小文字が区別されます。 ◆ tcpip (デフォルト) ◆ http
-q	メッセージまたはバナーを表示しない
-s streamParms=*keyword=value;...*	同期時に使用するストリーム・パラメータ。 ストリーム・パラメータは *stream-type* 固有です。デフォルトの streamParms は、-k tcpip stream-type に該当する "host=localhost" です。 使用可能なストリーム・パラメータのリストについては、『Mobile Link クライアント』> 「Ultra Light 同期クライアントのネットワーク・プロトコルのオプション」を参照してください。 ブール値の false は、**0**、**no**、**off**、または **false** によって指定できます。また、true は、**1**、**yes**、**on**、または **true** によって指定できます。 整数値は、8 進数、16 進数、または 10 進数で指定できます。 文字列値は引用符で囲うことができます (引用符は削除されます)。
-v	冗長メッセージを表示する

- **unload** Ultra Light データベース・ファイルを XML ドキュメントに保存します。次のオプションがサポートされています。

オプション	説明
xml-file	データのアンロード先の XML ファイル
-B *max-blob-size*	XML ファイルに書き込む long binary または long character データの最大サイズ。デフォルト値は 10 KB です。 値を -1 にすると、最大サイズが指定されません (すべてのデータが XML ファイルに格納されます)。
-c "*keyword=value;…*"	Ultra Light データベース接続パラメータ。必須。
-d	データのみをアンロードする。ストリームはアンロードしません。
-D *blob-directory*	**-B** *max-blob-size* 制限を越える long binary データを格納するディレクトリ。 値は、*tableName-columnName-rowNumber.bin* という名前のファイルとして格納されます。 デフォルトのディレクトリは、XML ファイルと同じディレクトリです。
-e *table,...*	指定されたテーブルのデータを除外する
-n	スキーマのみをアンロードする。データはアンロードしません。
-q	クワイエット・モード (メッセージを表示しない)
-S *schema-file*	指定された Ultra Light スキーマ・ファイルにスキーマを保存する
-t *table,...*	指定されたテーブルのデータのみを出力する
-v	冗長メッセージ
-y	XML ファイルが存在する場合、それを上書きする

説明

データベースのアンロードおよびロードの基本フォーマットである XML ファイルは、SQL Anywhere インストール環境のファイル *win32¥usm.xsd* です。このファイルは、Ultra Light XML ユーティリティのファイルと同じです。

アプリケーションが Embedded SQL または静的型 C++ API を使用して構築される場合、アプリケーションを Palm OS に配備する場合、および Ultra Light データベース・コンバータによって作成されたデータベースを使用する場合、アプリケーションで特別な呼び出しを行います。`ULEnableGenericSchemaEx ( &sqlca, false )` を呼び出してから、データベース API を初期化します。

接続文字列には、データベース・ファイル (DBF) パラメータが含まれています。

アンロード用として使用される場合、Ultra Light データベース・ファイルのテンポラリ・コピーが作成されてから、データがアンロードされます。

ロードまたは作成用として使用される場合、ユーザ ID とパスワードの組み合わせを接続文字列に指定できます。このユーザ ID とパスワードの組み合わせは、認証されたユーザのリストに追加されるほか、DBA と SQL のデフォルトのユーザ ID とパスワードの組み合わせにも追加されます。

例

次の例は、Ultra Light データベース・コンバータの使用例を示します。

- アプリケーションがある Palm OS 上で分散させる Ultra Light データベースを作成します。

  1. 統合データベースに対して Mobile Link を起動します。

     ```
     start dbmlsrv9 -c dsn=...
     ```

  2. Ultra Light スキーマ・ファイルを作成します。

     ```
     ulinit -f app.usm -c dsn=...
     ```

  3. 作成者 ID **SYB2** を使用して Palm OS フォーマットでデータベースを作成します。

     ```
     ulconv create -p SYB2 -c
     DBF=app.udb;schema_file=app.usm
     ```

このコマンドは、app.udb という名前のデスクトップ・ファイルを作成します。このファイルは、HotSync を使用して Palm OS デバイスに同期されるときに PDF ul_udb_SYB2 として表示されます。このファイルをデスクトップにエクスポートし戻すと、ul_udb_SYB2 として表示されます。

4. アプリケーション・データをダウンロードして、Mobile Link と同期します。次のコマンドをすべての行に入力します。

```
ulconv sync -c "DBF=app.pdb"
 username=appuid;version=app;downloadOnly=true
```

username、version、downloadOnly の各値は、同期を制御する同期パラメータです。

5. Ultra Light アプリケーションがある Palm デバイスに *palmos.pdb* をコピーします。

- すでにデスクトップにコピーされている既存の Windows CE Ultra Light データベースの断片化を解除します。

1. ストリームとデータを XML ファイル *cedatabase.xml* にアンロードします。

```
ulconv unload -c DBF=cedatabase.udb
cedatabase.xml
```

2. 新しいデータベースを XML ファイルにロードします (-u を使用すると、新しいユニコード・データベースになります)。

```
ulconv load -u -c DBF=newdb.udb cedatabase.xml
```

3. 新しいデータベース・ファイルを Windows CE デバイスにコピーします。

- 既存の Ultra Light データベースをデスクトップから Palm OS フォーマットに変換します。

1. データを *database.xml* にアンロードします。

```
ulconv unload -c DBF=database.udb database.xml
```

2. 作成者 ID **CREA** を使用して Palm OS フォーマットで新しい Ultra Light データベースを作成、ロードします。

```
ulconv load -p CREA -c dbf=palm.pdb
database.xml
```

3. *palm.pdb* を Palm OS デバイスにコピーします。

- UNICODE Ultra Light データベースからマルチバイト文字セットのデータベースに変換します。

```
ulconv unload -c DBF=CEDatabase.udb newdb.xml
 ulconv load -m -c DBF=MBDatabase.udb newdb.xml
```

- 指定されたテーブルをデータのみとして保存します。

```
ulconv unload -c dbf=existingDB.udb -d -t
table1,table2
```

ulcreate ユーティリティ

機能　Ultra Light データベースを作成します。

構文　**ulcreate** *options*

このユーティリティのコマンド・ライン・オプションでは、大文字と小文字が区別されます。

オプション	説明
-c "*keyword=value;…*"	Ultra Light データベース接続文字列。必須。 DBF パラメータを使用して、データベース・ファイル名を設定します。Ultra Light 接続文字列の詳細については、「接続パラメータ」83 ページを参照してください。
-C	すべての文字列比較で大文字と小文字を区別するものとしてデータベースを作成する
-I *database-id*	グローバル・オートインクリメント・カラムの初期データベース ID を設定する
-l	使用可能な照合順をリストして終了する。 照合順のリストについては、『ASA データベース管理ガイド』>「提供されている照合と推奨する照合」を参照してください。
-m	マルチバイトの Ultra Light データベースを作成する
-p *creator-id*	指定された作成者 ID を使用して Palm OS データベースを作成する。データベース・ファイルには、ul_udb_*creator-id* の Palm OS PDB 名が付けられます。 作成者 ID の詳細については、「Palm 作成者 ID の概要」242 ページを参照してください。
-q	クワイエット・モード (メッセージを表示しない)
-S *schema-file*	指定されたスキーマ・ファイルを使用して、データベース・スキーマを定義する

オプション	説明
-u	Unicode データベースを作成する (デフォルト)
-v	冗長メッセージを表示する
-y	データベース・ファイルが存在する場合、それを上書きする
-z *collation-sequence*	照合順を指定する

説明

データベース・ファイルが存在しない場合、データベースは、大文字と小文字が区別されず、現在のロケールに依存した照合順を持つ非 Unicode データベースとして作成されます。

例

すべてデフォルト設定を使用して Ultra Light データベースをファイルに作成します。

```
ulcreate -c "dbf=test.udb"
```

大文字と小文字を区別する Unicode データベースを作成して、データベース・ファイルが存在する場合はそれを上書きします。

```
ulcreate -c "dbf=test.udb" -C -u -y
```

データベースを作成し、ファイル *myschema.usm* で指定されているスキーマを適用します。

```
ulcreate -c "dbf=test.udb" -S myschema.usm
```

暗号化キー **afvc_1835** を使用して、暗号化されたデータベースを作成します。

```
ulcreate -c "dbf=test.udb;key=afvc_1835"
```

uldbsgen ユーティリティ

機能

テーブル定義と Mobile Link 同期スクリプトを含め、Adaptive Server Anywhere 統合データベースのスキーマを定義する SQL 文をファイルに書き込みます。

このユーティリティは、同期スクリプトの生成時に Ultra Light スキーマ・ペインタから呼び出されます。

構文

uldbsgen *options sql-file*

このユーティリティのコマンド・ライン・オプションでは、大文字と小文字が区別されます。

オプション	説明
-a *sync-info-file*	Ultra Light スキーマ・ペインタによって保存された、同期設定を含むファイルを指定する
-c "*keyword=value;...*"	Ultra Light データベース接続文字列。必須。 Ultra Light 接続文字列の詳細については、「接続パラメータ」83 ページを参照してください。
-d *var=value*	スクリプトの初期値を設定する
-e *table,...*	リストされたテーブルを除外する
-gm	Mobile Link 同期スクリプトのみ書き込む
-gt	テーブル定義のみ書き込む
-m *ml-version*	Mobile Link 同期バージョンを設定する
-oa	データベースをアップグレードする場合にキャンセルする (デフォルト)
-or	読み込み専用。データベースをアップグレードしない。
-ou	古い Ultra Light リリースのデータベースである場合にアップグレードする
-p *publication,...*	リストされているパブリケーションに含まれるテーブルに対してのみ文を書き込む

オプション	説明
-q	クワイエット・モード (メッセージを表示しない)
-s *event,...*	指定された同期イベントに対してのみ Mobile Link 同期スクリプトを書き込む
-t *table,...*	リストされているテーブルに対してのみ文を書き込む
-v	冗長メッセージ
-y	確認メッセージを表示しないで *sql-file* を上書きする

説明

uldbsgen ユーティリティは、主に Ultra Light スキーマ・ペインタによって使用されます。このユーティリティは、オプションのバッチモードを使用するためのコマンド・ライン・ユーティリティとして提供されています。

参照

- 「Ultra Light スキーマ・ペインタ」158 ページ
- 『SQL Anywhere Studio ヘルプ』> 「[統合データベースと Mobile Link スクリプトの生成] ダイアログ」

ulinit ユーティリティ

適用対象　Ultra Light コンポーネント

機能　*ulinit* ユーティリティを使用して、任意の Ultra Light コンポーネントで使用する *.usm* ファイルを作成できます。このユーティリティは Adaptive Server Anywhere データベースに接続します。したがって、このユーティリティを使用するには SQL Anywhere Studio (バージョン 8.0.2 以降) が必要です。

構文　**ulinit -f** *schema_file* **-n** *pub_name* [*options*]

オプション	説明
-c *"connection_string"*	*keyword=value* の形式で、データベース接続パラメータをセミコロンで区切って指定する。このパラメータを指定して、Adaptive Server Anywhere データベースに接続します。これらのパラメータを指定することによって、Adaptive Server Anywhere データベースに接続できるようになります。
-f *schema_file*	出力ファイルの名前を指定する。このオプションは必須です。
-m *version*	生成される Mobile Link スクリプトのバージョン文字列を指定する

オプション	説明
-n *pubname*	テーブルを Ultra Light データベース・スキーマに追加する。 *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・スキーマに追加されます。複数のパブリケーションを Ultra Light データベース・スキーマに追加するには、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 スキーマに追加するには、**-n*** を指定します。 このオプションは必須です。
-o *"keyword=value;…"*	スキーマ作成オプションを指定する
-palm *id*	Palm OS と互換性のあるスキーマ・ファイルを作成する。id は、データベースを識別する 4 桁の Palm 作成者 ID です。
-q	クワイエット・オペレーション - エラーと警告だけをレポートする
-s *pubname*	同期のパブリケーションを指定する。*pubname* は、指定したパブリケーションとして Ultra Light データベースに追加されるリファレンス・データベース内のパブリケーションを指定します。 -s を指定しないと、Ultra Light スキーマにはパブリケーションが指定されません。 このオプションは複数回使用できます。
-t *file*	信用されたルート証明書を含むファイルを指定する
-w	警告を表示しない

オプション	説明
-z *ordering*	同期中にテーブルがアップロードされる順序を指定する (たとえば、*-z table1,table2* と指定)

説明

-n オプションと -s オプションは両方とも、リファレンス・データベース内のパブリケーション名を引数として使用しますが、目的は異なります。

- -n オプションは、Ultra Light データベース・スキーマに入れるテーブルを定義します。このオプションは、指定したパブリケーションを Ultra Light データベースに作成せず、同期には使用されません。
- -s オプションは、Ultra Light データベース・スキーマに指定したパブリケーションを定義します。これら指定したパブリケーションは同期に使用されます。-s オプションは、Ultra Light データベース・スキーマに入れられるテーブルは定義しません。

例

テーブルを含む *customer.usm* というファイルを TestPublication に作成します。

```
ulinit -c "uid=dba;pwd=sql" -f customer.usm -n
TestPublication
```

2 つの異なるパブリケーションを持つスキーマを作成します。

```
ulinit -c "dsn=dsn-name" -f schema.usm -n Pub1 -n Pub2 -
s Pub1 -s Pub2
```

たとえば、1 つのパブリケーションには優先度の高い同期用の小さいデータのサブセットが含まれ、もう 1 つにはバルク・データが含まれます。

パブリケーションの同期は、Ultra Light スキーマでビット・マスクを使用して管理されます。詳細については、『Mobile Link クライアント』> 「別々に同期するためのデータ・セットの設計」を参照してください。

ulinit で Palm 用の Ultra Light スキーマを作成するときに、-palm オプションを使用して *.pdb* ファイルを生成します。次に例を示します。

```
ulinit -c "uid=dba;pwd=sql;dsn=ASA 9.0 Sample"
 -f tutcustomer.usm -n TutCustomersPub -palm Syb3
```

注意
Syb3 は、Palm 作成者 ID の例です。アプリケーションの作成者 ID と一致する 4 桁の登録済み Palm 作成者 ID を使用します。MobileVB 開発者の場合、この ID を MobileVB プロジェクト設定で設定してください。

ulinit で生成される PDB ファイルを Palm デバイスにロードします。アプリケーションによって使用される作成者 ID は、PDF ファイル名と一致させてください。Ultra Light アプリケーションがスキーマ・ファイルからデータベースを作成するときは、*.pdb* ファイル拡張子を付けない作成者 ID を Open を呼び出すときのパラメータに含めてください。次に例を示します。

```
DatabaseManager.CreateDatabase( "palm_schema=Syb3" )
```

Ultra Light Interactive SQL ユーティリティ

機能　Ultra Light データベースに対して SQL 文を実行します。

構文　**ulisql** [*options*] [*command-file*]

オプション	説明
-c "*keyword=value;...*"	Ultra Light データベース接続文字列。接続文字列を指定しない場合は、接続ダイアログが表示されます。 接続文字列には、次のキーワードが含まれます。 ◆ dbf (データベース・ファイル) ◆ uid (ユーザ ID) ◆ pwd (パスワード) 接続文字列の詳細については、「接続パラメータ」 83 ページを参照してください。
-oa	古いファイル・フォーマットのデータベースを開かない
-or	古いファイル・フォーマットのデータベースを読み込み専用として開く
-ou	古いファイル・フォーマットのデータベースを最新バージョンのフォーマットにアップグレードする

オプション	説明
command-file	実行する 1 つ以上の SQL 文を含むファイル。*command-file* が指定されている場合、ユーザ・インタフェースは表示されません。 ファイルは、次の規則に従います。 ◆ 各文は、セミコロンで終わる。空白以外の文字をセミコロンと行の終わりの間に含めない。 ◆ 各文は改行で始まる。文の前には空白のみ含めることができる。 ◆ コメント文字はサポートされない。 ◆ 結果セット (クエリ) を返す文は無視される。

説明

Ultra Light Interactive SQL は、Ultra Light データベースに対して SQL 文を実行するためのグラフィカル・ユーティリティです。このユーティリティは、アプリケーション開発中に、SQL 文の開発とテスト、データベース内のデータのブラウズに役立ちます。

Ultra Light Interactive SQL には、クエリのアクセス・プランを表示するウィンドウがあります。パフォーマンスが低いクエリがある場合は、このクエリ・プランが問題の診断に役立つ可能性があります。たとえば、適切なインデックスを作成すると、クエリの高速化に役立つ場合があります。

古いデータベース・ファイルをソフトウェアの最新バージョンのネイティブ・ファイル・フォーマットにアップグレードすることを選択した場合、Ultra Light によってこの処理が行われます。このアップグレードによって、データベースは、古いバージョンのソフトウェアで開発したアプリケーションを含め、古いバージョンの Ultra Light からは使用できなくなります。古いデータベースがアップグレードされないようにするには、-or または -oa オプションを使用します。-or オプションを選択した場合、Ultra Light はデータベースを読み込み専用モードで開きます。行った変更は、コミットした場合でも、Ultra Light Interactive SQL の終了時に破棄されます。

いずれの **-o** パラメータも指定しない場合、データベースをアップグレードする必要があると、[Upgrade] ダイアログ・ボックスが表示されます。このダイアログを使用して、アップグレードするか、読み込み専用モードで開くか、または操作をキャンセルするかを選択できます。

参照

- 「動的 SQL 文」217 ページ
- 「クエリの最適化」234 ページ

ulload ユーティリティ

機能

XML ファイルから Ultra Light データベースにデータをロードします。

構文

ulload *options xml-file*

このユーティリティのコマンド・ライン・オプションでは、大文字と小文字が区別されます。

オプション	説明
-c "*keyword=value;...*"	Ultra Light データベース接続文字列。必須。 DBF パラメータを使用して、データベース・ファイル名を設定します。Ultra Light 接続文字列の詳細については、「接続パラメータ」83 ページを参照してください。
-d	データのみをロードする
-D *directory*	外部ファイルに格納されたデータのディレクトリを指定する。デフォルトは、XML ファイルと同じディレクトリです。 このオプションは、主に ulunload で作成された XML ファイルに使用されます。
-i	挿入されたローを次の同期時にアップロードする。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。
-m	マルチバイト・データベースを作成する
-n	スキーマのみ。データは無視されます。
-p *creator-id*	指定された *creator-id* を使用して Palm OS データベースを作成する
-q	クワイエット・モード (メッセージを表示しない)
-S *schema-file*	指定されたスキーマ・ファイルに Ultra Light スキーマを保存する
-u	Unicode データベースを作成する (デフォルト)。

オプション	説明
-v	冗長メッセージ
-y	確認メッセージを表示しないでデータベースを置き換える。 このオプションは、**-d** が使用されている場合は機能しません。

説明

ulload ユーティリティは、XML ファイルから Ultra Light データベースにデータをロードします。Ultra Light データベース・ファイルが存在しない場合は、これを作成します。オプションによって、データベース・スキーマ、データ、またはその両方をロードするかの指定と、Ultra Light データベースの文字セットの指定ができます。

例

sample.xml 内のスキーマとデータから、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ます。

```
ulunload -d -c dbf=sample.udb sample.xml
```

ulsync ユーティリティ

機能

Ultra Light データベースを同期します。

構文

ulsync *options sync-parms*

このユーティリティのコマンド・ライン・オプションでは、大文字と小文字が区別されます。

オプション	説明
-a *authentication-parameter*	Mobile Link 認証パラメータ
-c "*keyword=value;...*"	Ultra Light データベース接続文字列。必須。 DBF パラメータを使用して、データベース・ファイル名を設定します。Ultra Light 接続文字列の詳細については、「接続パラメータ」83 ページを参照してください。
-k *stream-type*	同期ストリームを指定する。*stream-type* には、**tcpip** または **http** を指定します。デフォルトのストリームは **tcpip** です。
-q	クワイエット・モード (メッセージを表示しない)
-s *stream-parm=value;...*	セミコロンで区切られた同期ストリーム・パラメータのリスト。デフォルト値は **host=localhost** です。 詳細については、『Mobile Link クライアント』>「Ultra Light 同期クライアントのネットワーク・プロトコルのオプション」を参照してください。
-v	冗長メッセージ
-y	確認メッセージを表示しないでデータベースを置き換える

オプション	説明
sync-parms	セミコロンで区切られたキーワードと値の組み合わせのリスト。キーワードでは、大文字と小文字を区別しません。キーワードのリストは次のとおりです。 ◆ downloadOnly (ブール値) ◆ newpasswd (文字列) ◆ passwd (文字列) ◆ publicationMask (整数) ◆ sendColumnNames (ブール値) ◆ uploadOnly (ブール値) ◆ username (文字列) ◆ version (文字列)

説明

ulsync ユーティリティは、Ultra Light データベースを Mobile Link 同期サーバと同期させます。これは、アプリケーション開発中に同期をテストするためのツールです。

sync-parms オプション (同期パラメータを設定) を **-s** *stream-parms* オプション (同期ストリーム・パラメータを設定) と混同しないでください。

同期パラメータの詳細については、『Mobile Link クライアント』> 「同期パラメータ」を参照してください。ストリーム・パラメータの詳細については、『Mobile Link クライアント』> 「Ultra Light 同期クライアントのネットワーク・プロトコルのオプション」を参照してください。

例

次のコマンドは、データベース・ファイル *ul.udb* を、マシン **server** で稼働していてポート 8181 で受信している Mobile Link 同期サーバと HTTP によって同期します。Mobile Link ユーザ名は **ml-user** であり、スクリプト・バージョンは **script-version** です。

このコマンドは、1 行に入力してください。

```
ulsync -c "dbf=ul.udb"
        -k http
        -s "host=server;port=8181"
        "username=ml-user;version=script-version"
```

ulunload ユーティリティ

機能

Ultra Light データベースから XML ファイルにデータをアンロードします。

構文

ulunload *options xml-file*

このユーティリティのコマンド・ライン・オプションでは、大文字と小文字が区別されます。

オプション	説明
-B *max-size*	XML ファイルに格納するバイナリまたは文字データの最大サイズ。デフォルトは 10K。**-B -1** を使用すると、最大サイズが指定されず、すべてのデータが XML ファイルに格納されます。 最大サイズを超えるデータは、*tablename-colum-name-rownumber.bin* という名前のファイルとして、XML ファイルと同じディレクトリまたは -D で指定されたディレクトリに保存されます。
-c "*keyword=value;...*"	Ultra Light データベース接続文字列。必須。 DBF パラメータを使用して、データベース・ファイル名を設定します。Ultra Light 接続文字列の詳細については、「接続パラメータ」83 ページを参照してください。
-d	データのみをアンロードする。スキーマ定義はアンロードしません。
-D *directory*	**-B** で指定され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指定する。デフォルトは、XML ファイルと同じディレクトリです。
-e *table,...*	リストされたテーブルを除外する
-n	スキーマのみ。データは無視されます。
-q	クワイエット・モード (メッセージを表示しない)

オプション	説明
-S *schema-file*	指定されたスキーマ・ファイルに Ultra Light スキーマを保存する
-t *table,...*	リストされたテーブルだけをアンロードする
-v	冗長メッセージ
-y	確認メッセージを表示しないで *xml-file* を上書きする

説明

ulunload ユーティリティは、Ultra Light データベースから XML ファイルにデータをアンロードします。**-d** オプションを使用しないかぎり、アンロードされたファイルは、ulload ユーティリティを使用して新しいデータベースにロードできます。

例

Ultra Light データベース *sample.udb* を XML ファイル *sample.xml* にアンロードします。

```
ulunload -c "dbf=sample.udb" sample.xml
```

Ultra Light データベース *sample.udb* から XML ファイル *sample.xml* にデータをアンロードして、ファイルが存在する場合にそれを上書きします。

```
ulunload -c "dbf=sample.udb" -d -y sample.xml
```

ULUtil ユーティリティ

機能

Ultra Light Palm ユーティリティは、Ultra Light アプリケーションのリモート・データベースに格納されているデータをすべて削除する Palm Computing Platform アプリケーションです。

説明

Ultra Light Palm 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。

```
%ASANY9%¥UltraLite¥Palm¥68k¥ULUtil.prc
```

ULUtil は、デバイスが異なるユーザ間で共有されるような配備で役立ちます。デバイスを取得したときに、前のユーザのデータをクリアして記憶領域を節約できます。また、前のユーザも機密性のあるデータをクリアできます。**ULUtil** を使用しない場合、アプリケーションのデータをクリアするには、アプリケーションを削除して再インストールするしか方法はありません。

ULUtil を設定すると、以降の同期で Palm ストアを PC にバックアップできます。この機能を使用すると、最初の同期を実行してから、ストアをバックアップできます。このストアは他のデバイスに配備できるため、最初の同期を実行する必要がなくなります。バックアップ・オプションは、その後のバックアップを防ぐために Ultra Light ランタイムによって自動的にオフにされます。同期のたびにバックアップすることをデータベースに明示的に要求する場合は、UL_STORE_PARMS に palm_allow_backup パラメータを追加してください。

詳細については、『Ultra Light C/C++ ユーザーズ・ガイド』> 「UL_STORE_PARMS マクロ」を参照してください。

ULUtil をデバイスにインストールしてしまえば、Ultra Light アプリケーションのデータは次の手順で削除できます。

1. **ULUtil** に切り替えます。
2. [Ultra Light アプリケーション] リストからアプリケーションを選択します。バージョン 8 以降で構築されたアプリケーションのみが表示されます。
3. [削除] ボタンを押します。

拡張カードを使用しているデバイスでは、ULUtil によりファイルベース・ストアとレコードベース・ストアの両方へのアクセスが可能になります。

Ultra Light スキーマ・ペインタ

機能

Ultra Light スキーマ・ペインタを使用して、新しい Ultra Light スキーマ・ファイルを作成するか、既存の Ultra Light スキーマ・ファイルを編集できます。また、Adaptive Server Anywhere 統合データベースのテーブル定義と同期スクリプトも定義できます。

Ultra Light スキーマ・ペインタの機能の一部を説明するチュートリアルについては、「チュートリアル：Ultra Light データベースの使用」163 ページを参照してください。

Ultra Light スキーマ・ペインタの起動

❖ **Ultra Light スキーマ・ペインタを起動するには、次の手順に従います。**

- [スタート] ー [プログラム] ー [Sybase SQL Anywhere 9] ー [Ultra Light] ー [Ultra Light Schema Painter] を選択します。

  または、コマンド・プロンプトを開き、次のコマンドを入力します。

  ```
  ulview
  ```

スキーマ・ファイルの作成、保存、エクスポート

❖ **新しいスキーマ・ファイルを作成するには、次の手順に従います。**

1. [ファイル] ー [新規作成] フォルダを開き、[Ultra Light Schema] をクリックします。
2. [新しい Ultra Light スキーマ] ダイアログで、ファイル名を入力します。
3. [OK] をクリックし、スキーマを作成します。

❖ **ファイルを保存するには、次の手順に従います。**

1　[ファイル] ー [保存] を選択してファイルを保存します。

2　保存形式として、*.xml* または *.usm* を選択できます。

❖ **Palm スキーマ・ファイルをエクスポートするには、次の手順に従います。**

1　スキーマ・アイコンを右クリックし、ポップアップ・メニューから [Palm 用にスキーマをエクスポート] を選択します。

2　Palm の作成者 ID を入力します。

3　[OK] をクリックします。

スキーマ・ファイルの管理

スキーマ内のテーブルまたはカラムの名前を初めて変更する場合、テーブルまたはカラムの元の名前が保存されます。たとえば、cust という名前のテーブルを作成し、これを後で customer に変更する場合、cust は古い名前として保存されます。このテーブルの名前を 2 回目に customer_info に変更する場合、古い名前は cust として残ります。

スキーマは、スキーマ・ファイルを使用して既存のデータベースのスキーマを変更できるよう設計されています。たとえば、アプリケーションのあるバージョン 1 が cust という名前のテーブルを使用して出荷されたとします。バージョン 2 の変更の一部として、テーブルの名前を customer に変更し、バージョン 1 のスキーマ・ファイルを変更します。これによって、cust が古い名前として自動的に保存されます。ここで、このスキーマ・ファイルをバージョン 1 のデータベース・ファイルに適用すると、古い名前である cust という名前のテーブルが検索され、これが Customer に変更されます。このような規則はテーブル内のカラムにも適用されます。

このため、スキーマ・ファイルを配備したら古い名前をスキーマ・ファイルからクリアすることが、将来の互換性を保つ上で重要になります。

詳細については、「Ultra Light データベース・スキーマのアップグレード」71 ページを参照してください。

❖ 配備後にスキーマ・ファイル内の古い名前をすべてクリアするには、次の手順に従います。

1. Ultra Light スキーマ・ペインタを使用してスキーマ・ファイルを開きます。
2. [ファイル] ー [アップグレード情報のクリア] を選択します。

これによって、テーブルとカラムの古い名前がすべて空の値に設定されます。これで、アプリケーションの次のバージョン用としてスキーマ・ファイルを安全に編集できます。

オブジェクト名の手動変更

場合によっては、テーブルやカラムの名前を手動変更した方がよいときがあります。たとえば、アプリケーションのバージョン 1 と 2 が配備されているときに、このデータベースのバージョン 1 と 2 の両方をバージョン 3 にアップグレードできる単一の Ultra Light スキーマ・ファイルを作成したいとします。

❖ オブジェクト名を手動変更するには、次の手順に従います。

1. Ultra Light スキーマ・ペインタを使用してスキーマを開きます。
2. [ファイル] ー [Palm 用にスキーマをエクスポート] を選択します。

この機能を使用して、スキーマ内の現在の古い名前を確認できます。ulxml を使用する場合、<table> と <column> XML 要素にテーブルとカラムの古い名前を明示的に設定できます。

ulxml ユーティリティ

適用対象　Ultra Light コンポーネント

機能　*ulxml* ユーティリティを使用して、XML、USM、および Palm PDB ファイル・フォーマット間で Ultra Light データベース・スキーマ定義を変換できます。たとえば、XML ファイルから *.usm* ファイルを作成できます。このスキーマ・ファイルは任意の Ultra Light コンポーネントで使用できます。

構文　**ulxml** [*options*] *input-file output-file*

オプション	説明
-y	出力ファイルが存在する場合、それを上書きする
-to*type* where *type=xml*|*usm*|*pdb* 注意：pdb ファイルには CreatorID が必要です。	ファイルを以下の標準フォーマットのいずれかに変換する。 *toxml* を使用して Ultra Light スキーマを XML に変換する。 *tousm* を使用して XML ファイルを Ultra Light スキーマに変換する。 *topdb creator-id* を使用して XML ファイルを Palm 用の Ultra Light スキーマに変換する。

Ultra Light スキーマをエクスポートして XML フォーマットで作業できます。

```
<?xml version="1.0" ?>
- <ul:ulschema xmlns:ul="urn:ultralite">
  - <tables>
    - <table name="ULCustomer">
      - <columns>
          <column name="cust_id" type="integer" null="yes"
            default="global_autoincrement" />
          <column name="cust_name" type="varchar(30)" />
        </columns>
      - <primarykey>
          <primarycolumn name="cust_id" direction="asc" />
        </primarykey>
      - <indexes>
        - <index name="ULCustomerName" unique="yes">
            <indexcolumn name="cust_name" direction="asc" />
          </index>
        </indexes>
      </table>
    + <table name="ULProduct">
    - <table name="ULEmployee">
      - <columns>
          <column name="emp_id" type="integer" null="no" />
```

ドキュメントを定義する XML スキーマは、SQL Anywhere インストール環境の *win32¥usm.xsd* に格納されています。ulxml ユーティリティにはこのスキーマ・ファイルが必要です。

Samples¥NativeUltraLiteForJava¥sample.xml、*Samples¥UltraLiteActiveX¥sample.xml*、*Samples¥UltraLiteForMobileVB¥sample.xml* にある記述サンプルを表示、使用できます。

Ultra Light スキーマ・ペインタ

Ultra Light スキーマ・ペインタは、デフォルトでは、Ultra Light スキーマ・ファイルをネイティブの USM ファイル・フォーマットで作成、オープン、保存します。ただし、ファイル・タイプのドロップダウン・ボックスで Ultra Light XML スキーマ・ファイルを選択することによって、XML ファイルを作成、オープン、保存するオプションを使用できます。

リターン・コード

成功した場合は 0 が返され、失敗した場合は 0 未満が返されます。

第 6 章

チュートリアル：Ultra Light データベースの使用

この章の内容

この章では、Ultra Light データベース・スキーマとデータベース・ファイルに関連するいくつかのタスクを説明します。Ultra Light スキーマ・ペインタ、Ultra Light Interactive SQL ユーティリティ、Ultra Light コマンド・ライン管理ツールについて紹介します。また、Ultra Light データベースから Adaptive Server Anywhere 統合データベースを生成し、同期する方法も示します。

レッスン 1：Ultra Light データベース・スキーマの作成

このレッスンでは、Windows CE、Windows XP、Palm OS デバイスの単一テーブル Ultra Light データベース・スキーマを構築します。

このレッスンは、Ultra Light データベースの使用に関するチュートリアル全体の最初のレッスンです。このレッスンには、複数の Ultra Light コンポーネント・チュートリアルの 1 つから到達する場合もあります。その場合は、スキーマ・ファイルの作成後にメイン・チュートリアルに戻ってください。

Ultra Light スキーマの詳細については、「Ultra Light データベースとスキーマの作成」36 ページを参照してください。

このチュートリアルを開始するには、スキーマとその他のファイルを格納するディレクトリを作成します。このチュートリアルでは、*C:¥tutorial¥* ディレクトリを使用します。別のロケーションにチュートリアルのディレクトリを作成する場合は、*C:¥tutorial¥* の代わりに、そのディレクトリへのパスを指定してください。

❖ スキーマ・ファイルを作成するには、次の手順に従います。

1. Ultra Light スキーマ・ペインタを起動します。

   [スタート] ― [プログラム] ― [Sybase SQL Anywhere 9] ― [Ultra Light] ― [Ultra Light Schema Painter] を選択します。

2. 新しいスキーマ・ファイル *tutCustomer* を作成します。

   - [ファイル] メニューから [新規] ― [Ultra Light スキーマ] を選択します。
   - ファイル名については、**c:¥tutorial¥tutCustomer.usm** と入力するか、[参照] をクリックしてフォルダを選択し、**tutCustomer.usm** と入力します。
   - その他の設定はデフォルト値のままにし、[OK] をクリックしてスキーマを作成します。

3　テーブル customer を作成します。

- Ultra Light スキーマ・ペインタの左ウィンドウ枠で *tutCustomer* 項目を展開し、[テーブル] フォルダを選択します。
- 右ウィンドウ枠で、[テーブルの追加] をダブルクリックします。

  [新しいテーブル] ダイアログが表示されます。
- 名前を Customer と入力します。
- [追加] をクリックして次のカラムを追加します。

カラム名	データ型 (サイズ)	カラムの NULL 値の許可	デフォルト値
ID	integer	いいえ	オートインクリメント
FName	char (15)	いいえ	なし
LName	char (20)	いいえ	なし
City	char (20)	はい	なし
Phone	char (12)	はい	555-1234

チュートリアルでのみ使用

同期する予定のデータベースでは、オートインクリメントをプライマリ・キーとして使用することはおすすめしません。代わりに、グローバル・インクリメントまたは UUID 値を使用します。このチュートリアルの目的では、オートインクリメントで十分です。

同期設定でのユニーク・プライマリ・キーの管理の詳細については、『Mobile Link 管理ガイド』>「ユニークなプライマリ・キーの管理」を参照してください。

- ID をプライマリ・キーとして設定します。[プライマリ・キー] をクリックし、ID をインデックスに追加して昇順のマークを付けます。
- 内容を確認し、[OK] をクリックすると、テーブル定義が完了し、[新規テーブル] ダイアログが終了します。

4 [ファイル] － [保存] をクリックして *tutcustomer.usm* ファイルを保存します。

5 オプションとして、Palm スキーマ・ファイルをエクスポートします。

Palm OS をターゲットのプラットフォームとして使用する場合は、Palm OS のスキーマ定義をエクスポートできます。

- [ファイル] － [Palm 用にスキーマをエクスポート] を選択します。
- Palm 作成者 ID を入力します。チュートリアルの目的では **Syb3** を使用できますが、配備されたアプリケーションではこれを使用しないでください。

Palm 作成者 ID に関する注意

Palm 作成者 ID は Palm によって割り当てられます。サンプル・アプリケーションを作成する場合は、Syb3 を作成者 ID として使用できます。ただし、運用アプリケーションを作成する場合は、独自の作成者 ID を入手して使用してください。

- ファイル名をデフォルト設定のままにして、PDB ファイルをチュートリアル・ディレクトリに保存します。[OK] をクリックします。[OK] をクリックします。

これで Ultra Light データベースのスキーマが定義されました。このデータベースのテーブルは 1 つだけですが、Ultra Light データベースでは多数のテーブルを使用できます。

Ultra Light コンポーネント・チュートリアルの読者
このレッスンを Ultra Light コンポーネント・チュートリアルの一部として実行している場合、またはこの章のチュートリアル全体を順番に実行している場合があります。Ultra Light コンポーネント・チュートリアルからこのレッスンに到達した場合、必要なのはスキーマ・ファイルのみであるため、ここでメイン・チュートリアルに戻ることができます。それ以外の場合は、続行してください。

❖ Ultra Light データベース・ファイルマを作成するには、次の手順に従います。

1　Ultra Light スキーマ・ペインタの [ツール] メニューから、[Ultra Light データベースの作成] を選択します。

[Ultra Light データベースの作成] ダイアログが表示されます。

2　[マルチバイト文字データベース (デスクトップ)] を選択し、その他の設定はデフォルト値のままにします。

3　[OK] をクリックし、データベースを作成します。

レッスン 2：統合データベースの定義と作成

Ultra Light スキーマ・ペインタを使用して、Adaptive Server Anywhere 統合データベースのテーブルと同期スクリプトを定義する SQL コマンド・ファイルを生成できます。この機能は、Ultra Light アプリケーションを拡張して同期を組み込む場合に役立ちます。

このレッスンでは、Ultra Light データベースとのタイムスタンプベースの同期を管理する統合データベースを作成します。作業には、次の手順が含まれます。

1. Customer テーブルを保持するパブリケーションを作成します。
2. テーブルの同期設定を定義します。
3. テーブル定義と同期スクリプトを生成します。
4. Adaptive Server Anywhere 統合データベースを作成します。

このレッスンでは、「レッスン 1：Ultra Light データベース・スキーマの作成」164 ページの最後の状態と同じく、tutCustomer スキーマを開いた状態で Ultra Light スキーマ・ペインタを開いていることを想定します。

❖ Customer テーブルを保持するパブリケーションを作成するには、次の手順に従います。

1 パブリケーションを作成します。

左ウィンドウ枠の [パブリケーション] フォルダを選択します。右ウィンドウ枠にある [パブリケーションの追加] をダブルクリックします。[パブリケーション] ダイアログが表示されます。

2 [パブリケーション名] フィールドに、**CustomerPublication** と入力します。

3 Customer テーブルを選択し、[>>] をクリックしてパブリケーション内のテーブルのリストに追加します。[OK] をクリックします。

次の手順では、同期設定を定義します。

❖ 同期設定を定義するには、次の手順に従います。

1　カスタム設定のテーブルのリストにテーブルを追加します。

- 左ウィンドウ枠で、[Mobile Link 同期] フォルダを開きます。右ウィンドウ枠で、[テーブル固有の設定の追加] をダブルクリックします。

  [Mobile Link テーブル設定] ダイアログが表示されます。

- Customer テーブルを選択し、[>>] をクリックして、Mobile Link 設定のテーブルのリストに追加します。[OK] をクリックします。

2　設定を定義します。

左ウィンドウ枠で、[Mobile Link 同期] フォルダを選択します。

Customer テーブルを右クリックし、[プロパティ] を選択します。[Mobile Link 同期] プロパティ・シートが表示されます。ここでの設定は、このデモンストレーションを簡略化するために選択されています。運用環境では、設定はビジネス・ルールによって決まります。

a. [方向] タブでは、設定を [完全な同期] のままにします。

b. [ローのインクリメント] タブで、[タイムスタンプ] を選択します。[シャドー・テーブルを使用] オプションのチェックボックスの状態をデフォルトのまま残します。

このテーブルのタイムスタンプ値を保持する追加のカラムが、統合データベースに作成されます。

c. [ローの分割] タブでは、設定を [すべてのリモート・データベースで同じデータ] のままにします。

d. [削除されたロー] タブで、[削除されたローをダウンロードして、リモートからはローを削除] を選択し、2 つのオプションのうち最初のオプションを選択します。削除されたローの識別値を保持する Customer_deletes テーブルが、統合データベースに作成されます。

e. [競合] タブと [解決] タブでは、値をデフォルト設定のままにします。

f. [OK] をクリックして設定を保存します。

3 オプションとして、テーブル定義と同期スクリプトをプレビューします。

テーブルを右クリックし、ポップアップ・メニューで [統合テーブルとスクリプトのプレビュー] を選択して、テーブル定義と同期スクリプトをプレビューできます。プレビューされた定義とスクリプトは、この Ultra Light データベースと同期できる Adaptive Server Anywhere 統合データベースに必要なテーブルと同期スクリプトを定義します。コメント行の先頭には -- が付いています。

次の手順では、テーブル定義と同期スクリプトを保持する SQL コマンド・ファイルを生成します。

❖ 統合データベース・テーブルとスクリプトの定義を生成するには、次の手順に従います。

1 [ツール] メニューから、[統合テーブルとスクリプトの生成] を選択します。

[統合データベースと Mobile Link スクリプトの生成] ダイアログが表示されます。

2 [設定] グループで、Mobile Link スクリプト・バージョンを Tutorial に設定します。他の 2 つのチェックボックスは、デフォルト値のままにします。

3 [生成する SQL] グループで、[統合テーブル、トリガ、Mobile Link スクリプト、プロシージャ] チェックボックスがオンになっていることを確認します。

実際のアプリケーションの開発時には、アプリケーションを変更しながら同期スクリプトとテーブル定義を何度か再生成する場合があります。このため、ダイアログには、データベース・オブジェクトの一部のみを生成するオプションが用意されています。

4　[生成する SQL ファイル] はデフォルト設定 (*tutCustomer.sql*) のままにし、[OK] をクリックしてスクリプトを生成します。

最後の手順では、生成された SQL コマンド・ファイルを使用して、Adaptive Server Anywhere 統合データベースを作成します。

❖ 統合データベースを作成するには、次の手順に従います。

1　統合データベース・ファイルを作成します。

コマンド・プロンプトを開き、チュートリアル・ディレクトリに変更します。次のコマンドを入力して、データベース・ファイル *consol.db* を作成します。

```
dbinit consol.db
```

2　データベース用の ODBC データ・ソースを定義します。

a.　ODBC アドミニストレータを開きます。

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

b.　[ユーザ DSN] テーブルで [追加] をクリックします。[データ・ソースの新規作成] ダイアログが表示されます。

c.　リストから [Adaptive Server Anywhere 9.0] を選択し、[完了] をクリックします。[Adaptive Server Anywhere の ODBC の設定] ダイアログ・ボックスが表示されます。

d.　ダイアログに次の設定を入力します。

フィールド	値
データ・ソース名 ([ODBC] タブ)	Consolidated
ユーザ ID ([ログイン] タブ)	DBA
パスワード ([ログイン] タブ)	SQL
サーバ名 ([データベース] タブ)	consol
データベース・ファイル ([データベース] タブ)	c:¥tutorial¥consol.db

e. [ODBC] タブで、[接続のテスト] をクリックして設定をテストします。

f. 接続のテストに成功したら、[OK] をクリックして定義を保存し、ODBC アドミニストレータを閉じます。テストに失敗した場合は、設定を確認します。

3 Interactive SQL を使用して統合データベースに接続します。

a. [スタート] メニューから、[プログラム] － [SQL Anywhere 9] － [Adaptive Server Anywhere] － [Interactive SQL] の順に選択します。

b. [接続] ダイアログに ODBC データ・ソース Consolidated を指定し、[OK] をクリックして接続します。

4 生成された SQL コマンド・ファイルを開きます。

[ファイル] メニューで [開く] を選択します。*c:¥tutorial¥tut-Customer.sql* ファイルを開きます。

5 SQL コマンド・ファイルを実行します。

[SQL] － [実行] を選択して SQL 文を実行し、統合データベースにテーブルと同期スクリプトを作成します。

これで統合データベースが作成されました。当然ながらデータは含まれていません。

レッスン 3：Ultra Light データベースへのデータの入力

このチュートリアルでは、Ultra Light Interactive SQL を使用して、Ultra Light データベースにデータを追加します。このレッスンでは、Ultra Light Interactive SQL ユーティリティを紹介します。

❖ Ultra Light データベースにデータを入力するには、次の手順に従います。

1　Ultra Light Interactive SQL を起動します。

コマンド・プロンプトで次のコマンドを入力します。

```
ulisql
```

[接続] ダイアログが表示されます。

2　[ファイル名] フィールドで、レッスン 1 で作成した *tutCustomer.udb* ファイルをブラウズします。

ユーザ ID の DBA と パスワード の SQL を入力します。[OK] をクリックして、このデータベースに接続します。

3　次の文を入力して、データベースに Customer 名を追加します。

```
INSERT Customer (FName, LName, City )
 VALUES ('Jane', 'Doe', 'Boston')
```

[F5] を押して文を実行し、テーブルにローを追加します。

4　オプションとして、この文を変更してその他の名前を自由に追加します。名前の数は、このチュートリアルでは重要ではありません。

5　次の文を入力して変更をコミットします。

```
COMMIT
```

6　次の文を実行して、テーブル内の値をチェックします。

```
SELECT * FROM Customer
```

7 Ultra Light Interactive SQL を閉じます。

レッスン 4：データベースの同期

このレッスンでは、行った変更を Ultra Light データベースにアップロードします。Ultra Light アプリケーションがないため、このレッスンでは ulsync コマンド・ライン・ユーティリティを使用して同期します。

❖ データベースを同期するには、次の手順に従います。

1. 統合データベースに対して動作する Mobile Link 同期サーバを起動します。

   コマンド・プロンプトで、ディレクトリをチュートリアル・ディレクトリ (*consol.db* 統合データベース・ファイルを保持するディレクトリ) に変更し、次のコマンドを入力します。

   ```
   dbmlsrv9 -c "dsn=Consolidated" -zu+
   ```

   -zu+ コマンド・ライン・オプションは、認識されていないユーザが同期できる便利なオプションです。運用環境では使用しないでください。

2. Ultra Light データベースを同期します。

   この Ultra Light データベースにアクセスできるアプリケーションは一度に 1 つのみであるため、Ultra Light Interactive SQL が閉じられていることを確認します。

   コマンド・プロンプトで、次のコマンドを 1 行で入力します。

   ```
   ulsync -c "dbf=tutCustomer.udb"
   "version=Tutorial;username=test"
   ```

   データベースが同期します。

これで、チュートリアルは完了です。これにより、いずれかのデータベースでデータを調べること、Interactive SQL (統合データベースの場合) または Ultra Light Interactive SQL (Ultra Light データベースの場合) を使用してデータベースに変更を加えること、ここで説明した手法を使用してこれらのデータベースを同期することができます。

第 2 部　Ultra Light SQL

第 2 部では、Ultra Light アプリケーションで使用可能な SQL の範囲について説明します。

Ultra Light コンポーネントは、実行時にクエリやその他の SQL 文を作成できます (動的 SQL)。

静的インタフェースはより広範囲の SQL をサポートしていますが、アプリケーションで使用する文は、コンパイル時に指定します。

第 7 章

SQL 言語の要素

この章の内容

この章では、Ultra Light データベースの SQL 文の構成要素とデータ管理について説明します。これらの SQL 文の構成要素は、すべての Ultra Light データベースで共通です。

Ultra Light の SQL サポートの概要

Ultra Light では、データを表すために使用できるデータ型とデータにアクセスするために使用できる SQL 機能は、採用している開発モデルによって決まります。

静的インタフェース (Embedded SQL、静的型 C++ API、または静的型 Java API) を使用する場合、使用できる SQL の範囲は広くなりますが、アプリケーションによって使用されるすべての文はコンパイル時に指定します。Ultra Light コンポーネントを使用してアプリケーションを開発する場合、動的 SQL で使用できる SQL の範囲は狭くなりますが、SQL 文は実行時に構築できます。

Ultra Light プログラムが、Ultra Light でサポートされていない SQL 文や機能を使用しようとすると、「`Ultra Light` では使用できない機能です。」(SQLCODE -749) という SQL エラーがレポートされます。また、動的 SQL によって構文エラーが返されることもあります。

- **データ型** Ultra Light は、Adaptive Server 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。

  Adaptive Server Anywhere 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場合、広い範囲のデータ型を使用できます。Ultra Light ではサポートされていないこれらの Adaptive Server Anywhere のデータ型は、Ultra Light ジェネレータによってより小さい基本型のセットに変換されます。スキーマ・ペインタを使用して Ultra Light データベースを作成する場合、使用できるのはより小さい基本型のセットに制限されます。

  Ultra Light の基本型のリストについては、「Ultra Light のデータ型」184 ページを参照してください。

  Adaptive Server Anywhere データ型の完全なリストについては、『ASA SQL リファレンス・マニュアル』> 「SQL データ型」を参照してください。

- **識別子** 識別子は、カラムやテーブルなど、データベース・オブジェクトの名前を表します。Ultra Light は、Adaptive Server Anywhere と同じ識別子の規則をサポートしています。

Ultra Light のテーブルには所有者がいません。既存の SQL と、プログラムで生成された SQL の便宜のために、Ultra Light では、SQL 文で構文 *owner.table-name* がサポートされていますが、owner 部分はチェックされません。

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

- **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、Adaptive Server Anywhere と同じ文字列の規則をサポートしています。

  Ultra Light データベースを Adaptive Server Anywhere リファレンス・データベースから作成する場合、文字列の規則は、Ultra Light ジェネレータの実行時にリファレンス・データベースで有効なデータベース・オプションによって決まります。文字列の規則を設定する場合、特に重要なのは QUOTED_IDENTIFIER オプションです。動的 SQL は、常にこのオプションが ON (Adaptive Server Anywhere でのデフォルト設定) であるとして動作します。

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

  文字列の比較結果や文字列のソート順は、データベースと文字セットの両方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。

  詳細については、「Ultra Light データベースの作成」39 ページを参照してください。

- **関数** Ultra Light は、いくつかの小さな例外を除き、Adaptive Server Anywhere と同じ関数の範囲をサポートしています。サポートされている関数は、動的 SQL と同様、Embedded SQL などの静的インタフェースは同じです。

  サポートされている関数のリストについては、「Ultra Light SQL 関数」188 ページを参照してください。

- **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。

  Adaptive Server Anywhere では、式を形成するために広範囲の演算子を使用できます。これらの演算子は、静的インタフェース (Embedded SQL、静的型 C++ API、または静的型 Java API) を使用して Ultra Light アプリケーションを開発するときに使用できます。たとえば、Adaptive Server Anywhere では、SQL 変数を使用して式を形成できます。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。ただし、@@identity グローバル変数は例外であり、Ultra Light アプリケーション内で使用できます。

  Adaptive Server Anywhere の式の詳細については、『ASA SQL リファレンス・マニュアル』> 「式」を参照してください。

  動的 SQL は、サポートしている式の範囲が静的 SQL よりも制限されています。たとえば、動的 SQL は、静的 SQL でサポートされているすべての場所でサブクエリをサポートしているわけではありません。

  動的 SQL で使用できる式については、「動的 SQL 式」205 ページを参照してください。

- **探索条件** 探索条件または述部は、SELECT 文の WHERE 句、HAVING 句、ON 句で使用されます。

  動的 SQL は、サポートしている条件の範囲が静的 SQL よりも制限されています。

  動的 SQL で使用できる探索条件については、「動的 SQL の探索条件」214 ページを参照してください。

  静的インタフェースでは、Adaptive Server Anywhere でサポートされている条件の全範囲を使用できます。

  Adaptive Server Anywhere の探索条件の詳細については、『ASA SQL リファレンス・マニュアル』> 「探索条件」を参照してください。

- **文** SQL 文は、上記の構成要素から構築されます。

動的 SQL で使用できる SQL 文のリストについては、「動的 SQL 文」217 ページを参照してください。

静的 Ultra Light アプリケーションでは、次の SQL 文を使用できます。

- **データ操作言語**　SELECT 文、INSERT 文、UPDATE 文、DELETE 文を含めることができます。このような、実行時に値が設定される文では、プレースホルダを使用できます。

  詳細については、「Ultra Light SQL 文の記述」261 ページを参照してください。

- **TRUNCATE TABLE 文**　この文を使用して、テーブル全体をすばやく削除できます。

- **トランザクションの制御**　COMMIT 文と ROLLBACK 文を使用して、Ultra Light アプリケーションの中でトランザクションを制御できます。

- **START/STOP SYNCHRONIZATION DELETE 文**　これらの文を使用して、削除オペレーションの同期を一時的にサスペンドできます。

  詳細については、『Mobile Link クライアント』> 「削除同期の一時停止」を参照してください。

Ultra Light のその他の制限事項については、「Ultra Light データベースの制限事項」66 ページを参照してください。

Ultra Light のデータ型

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。

Adaptive Server Anywhere リファレンス・データベースから Ultra Light データベースを作成する場合、リファレンス・データベースの他のデータ型 (ユーザ定義のデータ型を含む) を使用できます。Ultra Light ジェネレータは、Ultra Light データベースでサポートされているデータ型にこれらのデータ型をキャストします。DEFAULT 値や CHECK 制約を含むユーザ定義のデータ型は使用できません。

動的 SQL を使用する場合や、スキーマ・ペインタを使用して Ultra Light データベースを作成する場合、使用できるのは、ここに記載されたデータ型に制限されます。

Adaptive Server Anywhere のデータ型については、『ASA SQL リファレンス・マニュアル』> 「SQL データ型」を参照してください。

データ型	説明説明
BIT	ブール値 (0 または 1)。『ASA SQL リファレンス・マニュアル』> 「BIT データ型」を参照してください。
{**CHAR** | **CHARACTER**} [**(** *max-length* **)**]	最大長 *max-length* 文字の文字データです。最大長は 2048 バイトです。『ASA SQL リファレンス・マニュアル』> 「CHAR データ型 [文字]」を参照してください。
{**VARCHAR** | **CHARACTER VARYING**} [**(** *max-length* **)**]	Ultra Light では、VARCHAR は CHAR と同じように実装されます。他のデータベースでは、VARCHAR は、最大長 *max-length* 文字の可変長文字データに対して使用されます。『ASA SQL リファレンス・マニュアル』> 「CHARACTER VARYING (VARCHAR) データ型 [文字]」を参照してください。

データ型	説明説明
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。 静的インタフェースの場合、LONG VARCHAR 値の最大サイズは 64 KB です。コンポーネント・インタフェースに明示的な制限はありません。 『ASA SQL リファレンス・マニュアル』> 「LONG VARCHAR データ型 [文字]」を参照してください。
[**UNSIGNED**] **BIGINT**	8 バイトの記憶領域を必要とする整数です。 『ASA SQL リファレンス・マニュアル』> 「BIGINT データ型 [数値]」を参照してください。
{**DECIMAL** | **DEC**} [**(** *precision* [, *scale*] **)**]	総桁数の *precision* と小数点以下の桁数の *scale* を持つ 10 進数です。『ASA SQL リファレンス・マニュアル』> 「DECIMAL データ型 [数値]」を参照してください。
NUMERIC [**(** *precision* [, *scale*] **)**]	DECIMAL と同じです。『ASA SQL リファレンス・マニュアル』> 「NUMERIC データ型 [数値]」を参照してください。
DOUBLE [**PRECISION**]	倍精度の浮動小数点数です。『ASA SQL リファレンス・マニュアル』> 「DOUBLE データ型 [数値]」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。『ASA SQL リファレンス・マニュアル』> 「FLOAT データ型 [数値]」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整数です。 『ASA SQL リファレンス・マニュアル』> 「INT または INTEGER データ型 [数値]」を参照してください。

データ型	説明説明
REAL	4 バイトで格納される単精度浮動小数点数。『ASA SQL リファレンス・マニュアル』>「REAL データ型 [数値]」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。『ASA SQL リファレンス・マニュアル』>「SMALLINT データ型 [数値]」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。『ASA SQL リファレンス・マニュアル』>「TINYINT データ型 [数値]」を参照してください。
DATE	年、月、日などの暦日です。『ASA SQL リファレンス・マニュアル』>「DATE データ型 [日付と時刻]」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。『ASA SQL リファレンス・マニュアル』>「TIME データ型 [日付と時刻]」を参照してください。
DATETIME	TIMESTAMP と同じです。『ASA SQL リファレンス・マニュアル』>「DATETIME データ型 [日付と時刻]」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。『ASA SQL リファレンス・マニュアル』>「TIMESTAMP データ型 [日付と時刻]」を参照してください。
VARBINARY [**(** *max-length* **)**]	BINARY と同じです。『ASA SQL リファレンス・マニュアル』>「VARBINARY データ型 [バイナリ]」を参照してください。
BINARY [**(** *max-length* **)**]	最大長さ *max-length* バイトのバイナリ・データです。最大長は 2048 バイトです。『ASA SQL リファレンス・マニュアル』>「BINARY データ型 [バイナリ]」を参照してください。

データ型	説明説明
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) は、LONG BINARY カラムでは実行できません。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。 静的インタフェースの場合、LONG BINARY 値の最大サイズは 64 KB です。コンポーネント・インタフェースに明示的な制限はありません。 『ASA SQL リファレンス・マニュアル』> 「LONG BINARY データ型 [バイナリ]」を参照してください。
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、あるコンピュータで生成される UUID 値が他のコンピュータで生成される UUID と一致しないように UUID 値を生成する機能が用意されています。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。 『ASA SQL リファレンス・マニュアル』> 「UNIQUEIDENTIFIER データ型 [バイナリ]」を参照してください。

Ultra Light SQL 関数

以下は、動的 SQL の関数を見つける上で役に立つリファレンスです。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。

Adaptive Server Anywhere の関数の詳細については、『ASA SQL リファレンス・マニュアル』> 「SQL 関数」を参照してください。

関数	説明
ABS (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』> 「ABS 関数 [数値]」を参照してください。
ACOS (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』> 「ACOS 関数 [数値]」を参照してください。
ARGN (*integer-expression* , *expression* [, ...] **)**	『ASA SQL リファレンス・マニュアル』> 「ARGN 関数 [その他]」を参照してください。
ASCII (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「ASCII 関数 [文字列]」を参照してください。
ASIN (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「ASIN 関数 [数値]」を参照してください。
ATAN (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「ATAN 関数 [数値]」を参照してください。
{ **ATN2** | **ATAN2** } **(** *numeric-expression1* , *numeric-expression2* **)**	『ASA SQL リファレンス・マニュアル』>「ATN2 関数 [数値]」を参照してください。

関数	説明
AVG (*numeric-expression* | **DISTINCT** *column-name* **)**	**DISTINCT** *column-name* は、動的 SQL からは使用できません。 『ASA SQL リファレンス・マニュアル』>「AVG 関数 [集合]」を参照してください。
BYTE_LENGTH (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「BYTE_LENGTH 関数 [文字列]」を参照してください。
BYTE_SUBSTR (*string-expression*, *start* [, *length*] **)**	『ASA SQL リファレンス・マニュアル』>「BYTE_SUBSTR 関数 [文字列]」を参照してください。
CAST (*expression* **AS** *data type* **)**	『ASA SQL リファレンス・マニュアル』>「CAST 関数 [データ型変換]」を参照してください。
CEILING (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「CEILING 関数 [数値]」を参照してください。
CHAR (*integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「CHAR 関数 [文字列]」を参照してください。
CHARINDEX (*string-expression1*, *string-expression2* **)**	『ASA SQL リファレンス・マニュアル』>「CHARINDEX 関数 [文字列]」を参照してください。
CHAR_LENGTH (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「CHAR_LENGTH 関数 [文字列]」を参照してください。
COALESCE (*expression*, *expression* [, ...] **)**	『ASA SQL リファレンス・マニュアル』>「COALESCE 関数 [その他]」を参照してください。
CONVERT (*data-type*, *expression* [, *format-style*] **)**	『ASA SQL リファレンス・マニュアル』>「CONVERT 関数 [データ型変換]」を参照してください。

関数	説明
COS (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「COS 関数 [数値]」を参照してください。
COT (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「COT 関数 [数値]」を参照してください。
COUNT (***** ｜ *expression* ｜ **DISTINCT** { *expression* *column-name* } **)**	**DISTINCT** *column-name* は、動的 SQL からは使用できません。 『ASA SQL リファレンス・マニュアル』>「COUNT 関数 [集合]」を参照してください。
DATALENGTH (*expression* **)**	『ASA SQL リファレンス・マニュアル』>「DATALENGTH 関数 [システム]」を参照してください。
DATE (*expression* **)**	『ASA SQL リファレンス・マニュアル』>「DATE 関数 [日付と時刻]」を参照してください。
DATEADD (*date-part*, *numeric-expression*, *date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DATEADD 関数 [日付と時刻]」を参照してください。
DATEDIFF (*date-part*, *date-expression1*, *date-expression2* **)**	『ASA SQL リファレンス・マニュアル』>「DATEDIFF 関数 [日付と時刻]」を参照してください。
DATEFORMAT (*datetime-expression*, *string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DATEFORMAT 関数 [日付と時刻]」を参照してください。
DATENAME (*date-part*, *date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DATENAME 関数 [日付と時刻]」を参照してください。

関数	説明
DATEPART (*date-part*, *date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DATEPART 関数 [日付と時刻]」を参照してください。
DATETIME (*expression* **)**	『ASA SQL リファレンス・マニュアル』>「DATETIME 関数 [日付と時刻]」を参照してください。
DAY (*date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DAY 関数 [日付と時刻]」を参照してください。
DAYNAME(*date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DAYNAME 関数 [日付と時刻]」を参照してください。
DAYS ([*datetime-expression*,] *datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DAYS 関数 [日付と時刻]」を参照してください。
DAYS (*datetime-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DAYS 関数 [日付と時刻]」を参照してください。
DEGREES (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DEGREES 関数 [数値]」を参照してください。
DIFFERENCE (*string-expression-1*, *string-expression-2* **)**	『ASA SQL リファレンス・マニュアル』>「DIFFERENCE 関数 [文字列]」を参照してください。
DOW (*date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「DOW 関数 [日付と時刻]」を参照してください。
EXP (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「EXP 関数 [数値]」を参照してください。
FLOOR (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「FLOOR 関数 [数値]」を参照してください。

関数	説明
GETDATE ( )	『ASA SQL リファレンス・マニュアル』>「GETDATE 関数 [日付と時刻]」を参照してください。
GREATER (*expression1* , *expression2* **)**	『ASA SQL リファレンス・マニュアル』>「GREATER 関数 [その他]」を参照してください。
HEXTOINT (*hexadecimal-string* **)**	『ASA SQL リファレンス・マニュアル』>「HEXTOINT 関数 [データ型変換]」を参照してください。
HOUR (*datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「HOUR 関数 [日付と時刻]」を参照してください。
HOURS ([*datetime-expression* ,] *datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「HOURS 関数 [日付と時刻]」を参照してください。
HOURS (*datetime-expression* , *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「HOURS 関数 [日付と時刻]」を参照してください。
IFNULL (*expression-1* , *expression-2* [, *expression-3*] **)**	『ASA SQL リファレンス・マニュアル』>「IFNULL 関数 [その他]」を参照してください。
INSERTSTR (*integer-expression* , *string-expression-1* , *string-expression-2* **)**	『ASA SQL リファレンス・マニュアル』>「INSERTSTR 関数 [文字列]」を参照してください。
INTTOHEX (*integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「INTTOHEX 関数 [データ型変換]」を参照してください。
ISDATE (*string* **)**	『ASA SQL リファレンス・マニュアル』>「ISDATE 関数 [データ型変換]」を参照してください。

関数	説明
ISNULL (*expression*, *expression* [, ...] **)**	『ASA SQL リファレンス・マニュアル』>「ISNULL 関数 [データ型変換]」を参照してください。
LCASE (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「LCASE 関数 [文字列]」を参照してください。
LEFT (*string-expression*, *para-expression*	『ASA SQL リファレンス・マニュアル』>「LEFT 関数 [文字列]」を参照してください。
LENGTH (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「LENGTH 関数 [文字列]」を参照してください。
LESSER (*expression1*, *expression2* **)**	『ASA SQL リファレンス・マニュアル』>「LESSER 関数 [その他]」を参照してください。
LIST ({ *string-expression* | **DISTINCT** *column-name* } [, *delimiter-string*] **)**	**DISTINCT** *column-name* は、動的 SQL からは使用できません。 『ASA SQL リファレンス・マニュアル』>「LIST 関数 [集合]」を参照してください。
LOCATE (*string-expression-1*, *string-expression-2* [, *integer-expression*] **)**	『ASA SQL リファレンス・マニュアル』>「LOCATE 関数 [文字列]」を参照してください。
LOG (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「LOG 関数 [数値]」を参照してください。
LOG10 (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「LOG10 関数 [数値]」を参照してください。
LOWER (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「LOWER 関数 [文字列]」を参照してください。

関数	説明
LTRIM (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「LTRIM 関数 [文字列]」を参照してください。
MAX (*expression* **)**	『ASA SQL リファレンス・マニュアル』>「MAX 関数 [集合]」を参照してください。
MIN (*expression* **)**	『ASA SQL リファレンス・マニュアル』>「MIN 関数 [集合]」を参照してください。
MINUTE (*datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「MINUTE 関数 [日付と時刻]」を参照してください。
MINUTES ([*datetime-expression*,] *datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「MINUTES 関数 [日付と時刻]」を参照してください。
MINUTES (*datetime-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「MINUTES 関数 [日付と時刻]」を参照してください。
MOD (*dividend*, *divisor* **)**	『ASA SQL リファレンス・マニュアル』>「MOD 関数 [数値]」を参照してください。
MONTH (*date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「MONTH 関数 [日付と時刻]」を参照してください。
MONTHNAME (*date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「MONTHNAME 関数 [日付と時刻]」を参照してください。
MONTHS ([*datetime-expression*,] *datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「MONTHS 関数 [日付と時刻]」を参照してください。
MONTHS (*datetime-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「MONTHS 関数 [日付と時刻]」を参照してください。

関数	説明
NEWID( )	この関数は、Ultra Light 静的型 Java API によってサポートされていません。 『ASA SQL リファレンス・マニュアル』>「NEWID 関数 [その他]」を参照してください。
NOW (*)	『ASA SQL リファレンス・マニュアル』>「NOW 関数 [日付と時刻]」を参照してください。
NULLIF (*expression-1* **,** *expression-2* **)**	『ASA SQL リファレンス・マニュアル』>「NULLIF 関数 [その他]」を参照してください。
PATINDEX (**'%***pattern***%'** **,** *string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「PATINDEX 関数 [文字列]」を参照してください。
PI (*)	『ASA SQL リファレンス・マニュアル』>「PI 関数 [数値]」を参照してください。
POWER (*numeric-expression-1* **,** *numeric-expression-2* **)**	『ASA SQL リファレンス・マニュアル』>「POWER 関数 [数値]」を参照してください。
QUARTER (*date-expression* **)**	『ASA SQL リファレンス・マニュアル』>「QUARTER 関数 [日付と時刻]」を参照してください。
RADIANS (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「RADIANS 関数 [数値]」を参照してください。
REMAINDER (*dividend* **,** *divisor* **)**	『ASA SQL リファレンス・マニュアル』>「REMAINDER 関数 [数値]」を参照してください。
REPEAT (*string-expression* **,** *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「REPEAT 関数 [文字列]」を参照してください。

関数	説明
REPLACE (*original-string*, *search-string*, *replace-string* **)**	『ASA SQL リファレンス・マニュアル』>「REPLACE 関数 [文字列]」を参照してください。
REPLICATE (*string-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「REPLICATE 関数 [文字列]」を参照してください。
RIGHT (*string-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「RIGHT 関数 [文字列]」を参照してください。
ROUND (*numeric-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「ROUND 関数 [数値]」を参照してください。
RTRIM (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「RTRIM 関数 [文字列]」を参照してください。
SECOND (*datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「SECOND 関数 [日付と時刻]」を参照してください。
SECONDS ([*datetime-expression*,] *datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「SECONDS 関数 [日付と時刻]」を参照してください。
SECONDS (*datetime-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「SECONDS 関数 [日付と時刻]」を参照してください。
SIGN (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「SIGN 関数 [数値]」を参照してください。
SIMILAR (*string-expression-1*, *string-expression-2* **)**	『ASA SQL リファレンス・マニュアル』>「SIMILAR 関数 [文字列]」を参照してください。
SIN (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「SIN 関数 [数値]」を参照してください。

関数	説明
SOUNDEX (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「SOUNDEX 関数 [文字列]」を参照してください。
SPACE (*integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「SPACE 関数 [文字列]」を参照してください。
SQRT (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「SQRT 関数 [数値]」を参照してください。
STR (*numeric-expression* [, *length* [, *decimal*]] **)**	『ASA SQL リファレンス・マニュアル』>「STR 関数 [文字列]」を参照してください。
STRING (*string-expression* [, ...] **)**	『ASA SQL リファレンス・マニュアル』>「STRING 関数 [文字列]」を参照してください。
STRTOUUID (*string-expression* **)**	この関数は、Ultra Light 静的型 Java API によってサポートされていません。 『ASA SQL リファレンス・マニュアル』>「STRTOUUID 関数 [文字列]」を参照してください。
STUFF (*string-expression1*, *start*, *length*, *string-expression2* **)**	『ASA SQL リファレンス・マニュアル』>「STUFF 関数 [文字列]」を参照してください。
{ **SUBSTRING** | **SUBSTR** }**(** *string-expression*, *start* [, *length*] **)**	『ASA SQL リファレンス・マニュアル』>「SUBSTRING 関数 [文字列]」を参照してください。

関数	説明
SUM (*expression* | **DISTINCT** *column-name* **)**	**DISTINCT** *column-name* は、動的 SQL からは使用できません。 『ASA SQL リファレンス・マニュアル』>「SUM 関数 [集合]」を参照してください。
TAN (*numeric-expression* **)**	『ASA SQL リファレンス・マニュアル』>「TAN 関数 [数値]」を参照してください。
TODAY (*)	『ASA SQL リファレンス・マニュアル』>「TODAY 関数 [日付と時刻]」を参照してください。
TRIM (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「TRIM 関数 [文字列]」を参照してください。
"TRUNCATE" (*numeric-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「TRUNCATE 関数 [数値]」を参照してください。
TRUNCNUM (*numeric-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「TRUNCNUM 関数 [数値]」を参照してください。
UCASE (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「UCASE 関数 [文字列]」を参照してください。
UPPER (*string-expression* **)**	『ASA SQL リファレンス・マニュアル』>「UPPER 関数 [文字列]」を参照してください。
UUIDTOSTR(*uuid-expression* **)**	この関数は、Ultra Light 静的型 Java API によってサポートされていません。 『ASA SQL リファレンス・マニュアル』>「UUIDTOSTR 関数 [文字列]」を参照してください。

関数	説明
WEEKS (［ *datetime-expression*, ］ *datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「WEEKS 関数 [日付と時刻]」を参照してください。
WEEKS (*datetime-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「WEEKS 関数 [日付と時刻]」を参照してください。
YEAR (［ *datetime-expression*, ］ *datetime-expression* **)**	『ASA SQL リファレンス・マニュアル』>「YEAR 関数 [日付と時刻]」を参照してください。
YEARS (*datetime-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「YEARS 関数 [日付と時刻]」を参照してください。
YMD (*integer-expression*, *integer-expression*, *integer-expression* **)**	『ASA SQL リファレンス・マニュアル』>「YMD 関数 [日付と時刻]」を参照してください。

第 8 章

動的 SQL

この章の内容

動的 SQL は、Ultra Light コンポーネントで使用可能な SQL のバージョンです。この章では、Ultra Light の動的 SQL の機能について説明します。

動的 SQL 文はランタイム時に構成できます。これは、コンパイル時にすべての SQL 文が指定されている必要がある Embedded SQL、静的型 C++ API、静的型 Java API アプリケーションで使用可能な静的型 SQL とは対照的です。

動的 SQL の紹介

アプリケーションは、クエリを使用して情報を検索したり、テーブルに新しいローを挿入するなどのデータベース・タスクを実行する場合に、SQL (Structured Query Language) を使用できます。SQL とは、規格制定機関である ANSI と ISO によって標準化されたリレーショナル・データベース言語です。Ultra Light の動的 SQL は、専有容量の小さいデバイスで使用するために設計された変形型です。

SQL 文は、使用しているプログラミング言語からの関数呼び出しの文字列として提供されます。Ultra Light コンポーネントは、SQL 文を構築および生成するための関数を提供します。SQL 文は、プログラミング・インタフェースによってデータベースに配信されます。データベースは文を受信するとこれを実行し、必要な情報 (クエリの結果など) をアプリケーションに戻します。

クエリは、SQL で使用されるデータ操作言語の 1 形式です。実際に、"SQL" の "Q" はクエリを表しています。SELECT 文を使用して、データベースからデータの問い合わせ (検索) をします。クエリによって、クエリの条件を満たすローの集合である結果セットが生成されます。リレーショナル・システム内の基本的なクエリ・オペレーションは、射影、制限、ジョインです。これらのオペレーションは、すべて SELECT 文によって実行できます。

射影とは、テーブル内のカラムのサブセットです。制限 (選択とも言う) は、いくつかの条件に基づいたテーブル内のローのサブセットのことです。たとえば、次の SELECT 文では、価格が $15 より高い製品すべての名前と価格が検索されます。

```
SELECT name, unit_price
 FROM product
 WHERE unit_price > 15
```

このクエリでは、射影 (SELECT 句) と制限 (WHERE 句) の両方を使用しています。

動的 SQL を使用すると、データのクエリだけでなく、さらに多くのことを実行できます。動的 SQL には、データを修正する文 (INSERT、UPDATE、DELETE 文)、トランザクションを制御する文 (COMMIT、ROLLBACK)、テーブルとインデックスを作成および削除する文 (CREATE、DROP TABLE、INDEX) も含まれています。

有効性

動的 SQL は、Ultra Light コンポーネントで使用可能な SQL のバージョンです。Ultra Light の静的型インタフェースは、異なる SQL のバージョンを使用します。Ultra Light コンポーネントは、テーブルベースのインタフェースと動的 SQL を使用できます。

これらのデータ・アクセス・メソッドの比較については、「コンポーネントと静的インタフェースの選択」14 ページを参照してください。

動的 SQL の使用

動的 SQL は、Ultra Light コンポーネントから使用できますが、静的型開発モデルからは使用できません。動的 SQL 文を実行する手順はすべてのコンポーネントで共通です。

1. 接続オブジェクト上で準備文メソッドを使用して文を準備します。メソッドの名前はインタフェースによって若干異なります。

   文を準備すると、文を表す文字列が解析および最適化 (準備) され、準備文を表すオブジェクトが返されます。最適化は、必ずしも Adaptive Server Anywhere ほど複雑な動作では行われません。

2. パラメータの値を設定します。

   必要に応じて、文に入力パラメータ (疑問符として指定) がある場合、アプリケーションは準備文オブジェクト上でメソッドを呼び出して、これらのパラメータの値を設定できます。値が設定されていないパラメータは NULL に設定されます。

3. 文を実行します。

   文が INSERT、UPDATE、または DELETE である場合は、ExecuteStatement メソッドを使用します。このメソッドは、文によって修正されたロー数を返します。

文が SELECT である場合は、ExecuteQuery メソッドを使用します。このメソッドは、クエリ結果セットを保持するオブジェクトを返します。

4. クエリについて、結果セットをナビゲーションし、結果セットの値にアクセスします。

   - 結果セット・オブジェクト上でメソッドを使用し、結果セット内の別のローに位置を設定できます。これには、MoveNext、MovePrevious、MoveFirst、MoveLast、Relative、BeforeFirst、AfterLast などがあります。
   - 現在の位置が結果セットのロー上にある場合、値を取得するメソッドを使用して結果セット内のカラムの値を取得できます。メソッドの名前はインタフェースによって異なります。メソッドは、データをアプリケーション・データ型に自動的に変換します。たとえば、結果が文字列変数に割り当てられる場合、整数結果式を自動的に文字列に変換できます。

5. 準備文を繰り返し実行する場合、手順 2 〜 4 を繰り返します。パラメータを使用すると、文全体をもう一度準備するよりも効率がよくなります。

   入力変数の値は準備文の実行後も保持されます。異なる値を使用する場合は、パラメータの値をリセットしてください。

動的 SQL 式

Ultra Light 動的 SQL の式は、カラム名、定数、演算子から成ります。式は値として評価されるため、データ型が関連付けられています。

構文

expression :
　　constant
　| *column-name*
　| **-** *expression*
　| *expression operator expression*
　| **(** *expression* **)**
　| *function-name* **(** *expression*, …**)**
　| *if-expression*
　| *case-expression*

参照

- 「式のサブクエリ」206 ページ
- 「IF 式」206 ページ
- 「CASE 式」207 ページ
- 「Ultra Light SQL 関数」188 ページ
- 「動的 SQL 演算子」209 ページ　「CASE 式」207 ページ「動的 SQL 演算子」209 ページ

集合式

集合式は、一連のローから単一の値を計算します。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
```

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* の式がすべて集合式であるか、いずれの式も集合式で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。

式のサブクエリ

サブクエリは、別の SELECT 文の内部でネストされた SELECT 文です。サブクエリは、FROM 句のテーブル式として使用できます。この場合、サブクエリは抽出テーブルとも呼ばれます。

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基準として作成できます。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への参照は行われません。これは、内部参照とも呼ばれます。抽出テーブルの場合、抽出テーブル名を指定し、SELECT リストの値をフェッチするためのカラム名を指定します。

サブクエリのその他の使用法については、「動的 SQL の探索条件」214 ページを参照してください。

IF 式

IF 式の構文は、次のとおりです。

IF *condition*
    THEN *expression1*
    [**ELSE** *expression2*]
    ENDIF

この式は次のように返します。

- *condition* が TRUE の場合、IF 式は *expression1* を返します。
- *condition* が FALSE の場合、IF 式は *expression2* を返します。
- *condition* が FALSE で *expression2* がない場合、IF 式は NULL を返します。
- 条件が UNKNOWN の場合、IF 式は NULL を返します。

TRUE、FALSE、UNKNOWN の各条件の詳細については、『ASA SQL リファレンス・マニュアル』> 「NULL 値」と『ASA SQL リファレンス・マニュアル』> 「探索条件」を参照してください。

CASE 式

CASE 式は条件付きの SQL 式を提供します。CASE 式は、式が使用できればどこでも使用できます。

構文 1

CASE *expression*
WHEN *expression* **THEN** *expression*, …
[**ELSE** *expression*]
END

CASE 文に続く式が WHEN 文に続く式と等しい場合、THEN 文の後の式が返されます。それ以外の場合、ELSE 文があればそれに続く式が返されます。

たとえば、次のコードでは CASE 式が SELECT 文の 2 番目の句として使用されています。

```
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 *expression*, …
[**ELSE** *expression*]
END

WHEN 文の後の探索条件が満たされた場合、THEN 文に続く式が返されます。それ以外の場合、ELSE 文があればそれに続く式が返されます。

たとえば、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。

```
SELECT id, name,
     ( CASE
         WHEN name='Tee Shirt' then 'Sale'
         WHEN quantity >= 50  then 'Big Sale'
         ELSE 'Regular price'
     END ) as Type
 FROM Product
```

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。

動的 SQL 演算子

演算子を使用して、式を比較、結合、または修正します。動的 SQL は、この項に記載されている演算子をサポートしています。Ultra Light の静的型インタフェースは、Adaptive Server Anywhere のすべての演算子にアクセスできます。

Adaptive Server Anywhere の演算子の詳細については、『ASA SQL リファレンス・マニュアル』> 「演算子」を参照してください。

バイナリ比較演算子

バイナリ比較条件の構文は、次のとおりです。

expression compare expression

ここで、*compare* は比較演算子です。次の比較演算子を使用できます。

演算子	説明
=	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
!<	以上

- **大文字と小文字の区別** 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。
- **NULL 演算子** NULL 式が関わる比較は、次の規則に従います。

  2 つの NULL 値は同等として比較されます。比較されるオペランドの 1 つが NULL である場合、結果は UNKNOWN になります。このため、SQL の比較によって、3 つの結果 (TRUE、FALSE、UNKNOWN) の 1 つが生成されます。同様に、論理式 (AND、OR、NOT) の場合もこれらの結果が生成されます。

算術演算子

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。

文字列演算子

expression || expression 文字列連結 (2 本の縦線)。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

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

たとえば、次の文は適切な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
```

論理演算子

論理演算子は、条件 (AND、OR、NOT) を比較したり、式 (IS) の真理値または NULL 値をテストしたりします。

AND を使用して、次のように条件を結合します。

condition1 **AND** *condition2*

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。

OR を使用して、次のように条件を結合します。

condition1 **OR** *condition2*

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 演算子の構文は、次のとおりです。

NOT *condition*

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 演算子では、論理値をテストできます。IS 演算子の構文は、次のとおりです。

expression **IS** [**NOT**] { *truth-value* | **NULL** }

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。

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。リストの最上部にある演算子から順に評価されます。

1. 名前、関数、定数
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

1 つの式に複数の演算子を使用する場合は、Ultra Light で同一の演算子の優先度に依存せず、カッコを使用して演算子の順序を明示指定することをおすすめします。

動的 SQL の探索条件

探索条件は、SQL クエリの WHERE 句または ON 句に記載されます。次の探索条件は、動的 SQL でサポートされています。

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*)

ALL 条件

ALL 条件の構文は、次のとおりです。

expression compare [**NOT**] **ALL** (*subquery*)

ここで、*compare* は比較演算子です。

ANY 条件

ANY 条件の構文は、次のとおりです。

expression compare [**NOT**] **ANY (** *subquery* **)**

ここで、*compare* は比較演算子です。

たとえば、等価演算子を持つ ANY 条件

expression = **ANY (** *subquery* **)**

は、*expression* がサブクエリ結果のいずれかの値と等しい場合、TRUE です。また、expression が NULL でなく、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、FALSE です。*expression* が NULL 値の場合、ANY 条件は UNKNOWN です。ただし、これはサブクエリの結果にローがない場合を除きます。この場合、条件は常に FALSE です。

BETWEEN 条件

BETWEEN 条件の構文は、次のとおりです。

expression [**NOT**] **BETWEEN** *start-expression* **AND** *end-expression*

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* の間にあればこの条件は TRUE です。NOT キーワードによってこの条件の意味は反対になりますが、UNKNOWN は変わりません。

BETWEEN 条件は、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*)

EXISTS 条件

EXISTS 条件の構文は、次のとおりです。

[**NOT**] **EXISTS(** *subquery* **)**

サブクエリ結果に少なくとも 1 つのローが含まれる場合、EXISTS 条件は TRUE で、サブクエリ結果にローが含まれない場合、EXISTS 条件は FALSE です。EXISTS 条件は、UNKNOWN にはなりません。

IN 条件

IN 条件の構文は、次のとおりです。

expression [**NOT**] **IN** { **(** *subquery* **)** | **(** *value-expr, …* **)** }

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。

NOT キーワードは、TRUE と FALSE を交換します。

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。

value-expr 引数は、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。

動的 SQL 文

Ultra Light では、次の動的 SQL 文を使用できます。

COMMIT 文

説明

この文を使用して、データベースを永続的に変更したり、ユーザ定義のトランザクションを終了します。

構文

COMMIT [**WORK**]

使用法

トランザクションは、COMMIT または ROLLBACK 文の間にデータベースへの 1 回の接続について行われる処理の論理単位です。COMMIT 文は、トランザクションを終了し、このトランザクション中で行われたすべての変更内容をデータベースに継続保管します。

CREATE および DROP 文は両方とも、COMMIT を自動的に実行します。

関連する動作

すべてのカーソルを閉じます。

CREATE INDEX 文

説明

この文を使用して、指定されたテーブル上にインデックスを作成します。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。

インデックスの詳細については、「クエリの最適化」234 ページを参照してください。

構文

CREATE [**UNIQUE**] **INDEX** *index-name*
ON [*owner*.]*table-name* **(** *column-name, …***)**

パラメータ

UNIQUE キーワード　UNIQUE 属性によって、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。外部キーは、複数の NULL のインスタンスを内包できるため、一意性制約があるプライマリ・キーまたはカラムを参照できます。

Ultra Light テーブルには所有者がいません。オプションの *owner* は、既存の SQL とプログラムで生成された SQL の便宜のためにサポートされています。Ultra Light では、*owner* は受け入れられますが無視されます。

使用法

CREATE INDEX 文は、指定されたテーブルの特定のカラムにソートされたインデックスを作成できます。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に役立たないインデックスの作成は避けてください。

- **排他的テーブルの使用** CREATE INDEX は、別の接続によって現在使用されているテーブルにこの文が影響する場合は常に阻止されます。CREATE INDEX には時間がかかることがあるため、この文が処理されている間、同じテーブルを参照している要求はサーバによって処理されません。
- **自動的に作成されたインデックス** 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。

CREATE TABLE 文

説明

この文を使用して、データベースにテーブルを作成します。

構文

CREATE TABLE [*owner*.]*table-name*
({ *column-definition* | *table-constraint* }, … **)**

column-definition :
　　column-name data-type [**NOT NULL**]
　　[**DEFAULT** *default-value*]
　　[**UNIQUE** | **PRIMARY KEY**]

default-value :
　　| *constant-expression*
　　| **AUTOINCREMENT**
　　| **GLOBAL AUTOINCREMENT** [**(** *partition-size* **)**]
　　| **NULL**
　　| **NEWID ( )**
　　| **CURRENT DATE**
　　| **CURRENT TIME**
　　| **CURRENT TIMESTAMP**

table-constraint :
　　UNIQUE (*column-name, …* **)**
　　| **PRIMARY KEY (** *column-name, …* **)**
　　| *foreign-key-constraint*

foreign-key-constraint :
　　[**NOT NULL**] **FOREIGN KEY (***column-name, …* **)**
　　REFERENCES *table-name* **(***column-name, …* **)**
　　[**CHECK ON COMMIT**]

パラメータ

column-definition　テーブル内のカラムを定義します。次は、カラム定義の一部を示します。

- **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。
- **data-type**　データ型については、「Ultra Light のデータ型」184 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 UNIQUE または PRIMARY KEY 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。
- **DEFAULT**　DEFAULT 値は、カラムの値を指定しない INSERT 文内のカラムの値として使用されます。DEFAULT を指定しない場合、DEFAULT NULL と等しくなります。

- **constant-expression** デフォルト値は、指定されている制約式です。DEFAULT 句で使用できるのは、データベース・オブジェクトを参照しない制約式のみです。式が単一値ではない場合 (加算演算子が含まれる式である場合など)、この式をカッコで囲みます。

- **AUTOINCREMENT** デフォルト値は、テーブル内のローごとにオートインクリメントした値です。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 がカラムの値を指定する場合、この値が使用されます。指定された値がカラムの現在の最大値より大きい場合、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。

  ローを削除しても AUTOINCREMENT カウンタは減りません。ローの削除によって作成されたギャップは、挿入を行うときに明示的に割り当てることによってのみ埋めることができます。最大値より小さい値のロー番号を明示的に挿入すると、明示的に割り当てられていない後続のローは、前の最大値より 1 大きい値を使用して自動的にインクリメントされます。

  各カラムに使用される次の値は整数として格納されます。$(2^{31} - 1)$ より大きい値を使用すると、ラップアラウンドによって誤った値が生成する可能性があります。このような場合、AUTOINCREMENT は使用しないでください。

- **GLOBAL AUTOINCREMENT** デフォルト値は、このデータベースに指定されている値の分割内でオートインクリメントした値です。GLOBAL AUTOINCREMENT のデフォルト値は、同期環境で複数のデータベースを使用するためのものです。

  このデフォルト値は、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。

AUTOINCREMENT キーワードの直後にカッコで分割サイズを指定できます。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ています。

カラムの型が BIGINT または UNSIGNED BIGINT である場合、デフォルトの分割サイズは 2^{32} = 4294967296 です。それ以外の型のカラムの場合、デフォルトの分割サイズは 2^{16} = 65536 です。特に、カラムの型が INT または BIGINT ではない場合は、これらのデフォルト値が適切では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。

このデフォルト値を使用する場合、各データベースのグローバル・データベース ID にユニークな正の整数を設定してください。この値は、データベースをユニークに識別し、デフォルト値の割り当て元の分割を示します。指定可能な値の範囲は、*n p* + 1 ～ (*n* + 1) *p* です。この場合、*n* はグローバル・データベース ID の値で、*p* は分割サイズです。たとえば、分割サイズに 1000 を、グローバル・データベース ID に 3 を設定すると、範囲は 3001 ～ 4000 になります。

前の値が (*n* + 1) *p* 未満であれば、このカラム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ります。カラムに値が含まれていない場合、最初のデフォルト値は *n p* + 1 になります。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ません。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ません。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。

グローバル・データベース ID には負の値は設定できないので、正の値が常に選択されます。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・データベース ID がデフォルト値の 2147483647 に設定されると、カラムには NULL 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。たとえば、テーブルのプライマリ・キーにカラムが含まれている場合に、この状況が発生します。

デフォルトの NULL 値は、分割で値が不足したときに生成されます。この場合には、データベースに新しいグローバル・データベース ID を割り当てて、別の分割からデフォルト値を選択できるようしてください。カラムで NULL が許可されていない場合に NULL 値を挿入しようとすると、エラーが発生します。

- **NULL** デフォルト値は NULL です。DEFAULT NULL であるカラムは、NULL を使用できて明示的なデフォルト値を持たないカラムと同じです。
- **NEWID** デフォルト値はユニバーサル・ユニーク識別子 (UUID) です。UUID を GLOBAL AUTOINCREMENT の代替物として使用して、多くのデータベースの同期環境でもテーブル内のローをユニークに識別できます。この値は、1 台のコンピュータ上で生成された値が別のコンピュータで生成された値と一致しないように生成されます。このため、この値は同期環境におけるキーとしても使用できます。

  カラム・タイプは、サイズが 16 バイト以上の UNIQUEIDENTIFIER または BINARY カラムです。

  NEWID は、非決定的関数です。NEWID を連続して呼び出すと、異なる値が返されることがあります。
- **CURRENT DATE** デフォルト値は、現在の日付です。
- **CURRENT TIME** デフォルト値は、現在の時間です。
- **CURRENT TIMESTAMP** デフォルト値は、日付と時間を示す現在のタイムスタンプです。

- **UNIQUE** カラム内のすべての値が別個でなければならないという制限を課します。UNIQUE と宣言されたカラムには NULL を使用できません。
- **PRIMARY KEY** カラムをテーブルのプライマリ・キーとして宣言します。プライマリ・キー・カラムには NULL を使用できず、各ローはユニークである必要があります。

1 つのテーブルが持つことができるプライマリ・キーは 1 つのみですが、このキーは複数のカラムから成ることがあります。複数のカラム・プライマリ・キーを持つテーブルを作成するには、テーブル制約のプライマリ・キーを宣言します。

table-constraint　テーブル制約は、テーブル内の 1 つまたは複数のカラムが保持できる値を制限します。

テーブル制約は、データベース内のデータの整合性を維持する上で役に立ちます。Ultra Light では、制約違反の原因となり、エラーをレポートする文は実行できません。

制約がテーブル内の複数のカラムを参照する場合、カラム制約の代わりにテーブル制約を使用します。

- **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。

  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。外部キーは、プライマリ・キーまたは一意性制約は参照できますが、ユニーク・インデックスは参照できません。ユニーク・インデックスは NULL の複数のインスタンスを含むことがあるからです。

  ユニーク・インデックスの詳細については、「CREATE INDEX 文」217 ページを参照してください。

- **PRIMARY KEY**　これは、1 つのテーブルが 1 つのプライマリ・キー制約のみを持つことができるという点を除いて、一意性制約と同じです。

  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。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。1 つのテーブルは 1 つの PRIMARY KEY のみを持つことができます。

- **foreign-key-constraint** 外部キー制約は、カラムのセットの値がプライマリ・キーの値、またはまれに別のテーブルの一意性制約 (プライマリ・テーブル) と一致するよう制限します。たとえば、外部キー制約を使用して、請求書テーブル内の顧客番号が顧客テーブル内の顧客番号と一致させることができます。

  外部キー・カラムを明示的に定義しないと、プライマリ・テーブル内の該当するカラムと同じデータ型を使用して作成されます。このように自動的に作成されたカラムは、外部テーブルのプライマリ・キーの一部になることはできません。このため、同じテーブルのプライマリ・キーと外部キーの両方で使用されるカラムは明示的に作成する必要があります。

  外部キー・カラム名を指定する場合は、プライマリ・キー・カラム名も指定する必要があり、カラム名はリスト内の位置に応じて組み合わされます。FOREIGN KEY テーブル制約にプライマリ・テーブルのカラム名が指定されていない場合、プライマリ・キー・カラムが使用されます。外部キー・カラム名が指定されていない場合、外部キー・カラムにはプライマリ・テーブル内のカラムと同じ名前が付けられます。

  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。

  CHECK ON COMMIT 句を使用すると、アクションが外部キー制約に違反していないかが確認されてから COMMIT が実行されます。

Ultra Light テーブルには所有者がいません。オプションの *owner* は、既存の SQL とプログラムで生成された SQL の便宜のためにサポートされています。Ultra Light では、*owner* は受け入れられますが無視されます。

DELETE 文

説明 この文は、データベースからローを削除するために使用します。

構文

DELETE
[**FROM**] [*owner*.]*table-name*
[**WHERE** *search-condition*]

使用法

DELETE 文は、指定したテーブルから、探索条件を満たすすべてのローを削除します。

警告
WHERE 句を指定しない場合、指定したテーブルからすべてのローは削除されます。

Ultra Light テーブルには所有者がいません。オプションの *owner* は、既存の SQL とプログラムで生成された SQL の便宜のためにサポートされています。Ultra Light では、*owner* は受け入れられますが無視されます。

関連する動作

なし。

例

データベースから従業員 105 を削除します。

```
DELETE
 FROM employee
 WHERE emp_id = 105
```

fin_data テーブルから、2000 より前のデータすべてを削除します。

```
DELETE
 FROM fin_data
 WHERE year < 2000
```

DROP INDEX 文

説明

この文を使用して、インデックス定義をデータベースから永続的に削除します。

構文

DROP INDEX [[*owner*.]*table-name*.]*index-name*

使用法

DROP INDEX 文は、指定されたインデックスの定義を削除します。

DROP INDEX は、別の接続によって現在使用されているテーブルにこの文が影響する場合は常に阻止されます。

Ultra Light テーブルには所有者がいません。オプションの *owner* は、既存の SQL とプログラムで生成された SQL の便宜のためにサポートされています。Ultra Light では、*owner* は受け入れられますが無視されます。

DROP TABLE 文

説明

この文を使用して、テーブル定義とテーブル内のすべてのデータをデータベースから永続的に削除します。

構文

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

使用法

DROP TABLE 文は、指定されたテーブルの定義を削除します。テーブル内のすべてのデータは、削除プロセスの一部として自動的に削除されます。また、テーブルのすべてのインデックスとキーは DROP TABLE 文によって削除されます。

DROP TABLE は、別の接続によって現在使用されているテーブルにこの文が影響する場合は常に阻止されます。

Ultra Light テーブルには所有者がいません。オプションの *owner* は、既存の SQL とプログラムで生成された SQL の便宜のためにサポートされています。Ultra Light では、*owner* は受け入れられますが無視されます。

INSERT 文

説明

1 つのローをテーブル (構文 1) に挿入したり、SELECT 文の結果をテーブル (構文 2) に挿入したりします。

構文 1

INSERT [**INTO**] [*owner*.]*table-name* [**(** *column-name, …* **)**]
VALUES (*expression, …* **)**

構文 2

INSERT [**INTO**] [*owner*.]*table-name* [**(** *column-name, …* **)**]
SELECT statement

使用法

INSERT 文を使って、データベース・テーブルに新しいローを追加します。

指定された式の値を使用して、1 つのローを挿入します。オプションのカラム名のリストを指定すると、指定したカラムの中に値が 1 つずつ挿入されます。カラム名のリストを指定しないと、作成順 (SELECT * を使って取り出すときと同じ順序) に値がテーブル・カラムの中に挿入されます。ローは、テーブル内の任意の位置に挿入されます。

カラム名を指定すると、select リストのカラムは、通常、カラム・リストに指定されたカラム、または作成順に並べたカラムと一致します。

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。したがって、テーブルの中へ挿入される文字列 **Value** は、常に大文字の V と残りの英字が小文字で格納されます。SELECT 文は、文字列を **Value** として返します。ただし、データベースで大文字と小文字が区別されない場合は、すべての比較において **Value** は **value**、**VALUE** などと同じとみなされます。さらに、単一カラムのプライマリ・キーにエントリ **Value** がある場合は、プライマリ・キーがユニークでなくなるので、INSERT 文による **value** の追加は拒否されます。

Ultra Light テーブルには所有者がいません。オプションの *owner* は、既存の SQL とプログラムで生成された SQL の便宜のためにサポートされています。Ultra Light では、*owner* は受け入れられますが無視されます。

関連する動作

なし。

例

データベースに Eastern Sales 部を追加します。

```
INSERT
 INTO department ( dept_id, dept_name )
 VALUES ( 230, 'Eastern Sales' )
```

ROLLBACK 文

説明

この文を使用して、トランザクションを終了し、最後に行った COMMIT または ROLLBACK 以降の変更を元に戻します。

構文

ROLLBACK [**WORK**]

使用法

トランザクションは、COMMIT または ROLLBACK 文の間にデータベースへの 1 回の接続について行われる処理の論理単位です。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。

SELECT 文

説明

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**] [**FIRST** | **TOP** *n*] *select-list*
[**FROM** *table-expression*]
[**WHERE** *search-condition*]
[**GROUP BY** *group-by-expression,...group-by-expression*]
[**HAVING** *search-condition*]
[**ORDER BY** *order-by-expression,...order-by-expression*]
[**OPTION (FORCE ORDER)**]

table-expression :
[*owner.*]*table-name* [[**AS**] *correlation-name*]
| *table-expression* { *join-operator table-expression*
[**ON** *join-condition*] ,... }
| **(** *table-expression, …* **)**
| **(** *select-statement* **)** [**AS**]
derived-table-name **(** [*column-name***,** ...] *column-name* **)**

join-operator :
, (ON condition not allowed)
| **CROSS JOIN** (ON condition not allowed)
| **INNER JOIN**
| **JOIN** (ON phrase required)
| **LEFT OUTER JOIN**

order-by-expression :
{ *integer* | *expression* } [**ASC** | **DESC**]

パラメータ

DISTINCT　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。DISTINCT を指定すると、重複した出力ローが削除されます。多くの場合、DISTINCT を指定すると、文の実行に時間が非常に長くかかります。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。

FIRST または TOP　クエリの最初のローまたはクエリの最初の *n* 個のローだけを明示的に取得できます。これらのキーワードは、主に ORDER BY クエリで使用されます。

select-list　*select-list* は、カンマで区切られた式のリストであり、データベースから何を取り出すかを指定します。アスタリスク (*) は、FROM 句に記述された全テーブルのすべてのカラムを選択することを意味します。サブクエリは、*select-list* では使用できません。

エイリアス名を *select-list* の式の後ろに指定することによって、この式を示すことができます。これによって、エイリアス名をクエリ内の別の位置 (WHERE 句や ORDER BY 句など) で使用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。*table-expression* は、上記の構文にリストされているようにベース・テーブルとサブクエリから成ります。

式の詳細については、「動的 SQL 式」205 ページを参照してください。

ON 条件　ON 条件は、単一のジョイン演算に対して指定され、ジョインを使用して結果セットにローを作成する方法を示します。WHERE 句は、ジョインによって潜在的なローが作成された後に結果セット内のローを制限するために使用します。INNER ジョインの場合、ON または WHERE を使用した制限は同じです。OUTER ジョインの場合、これらは同じではありません。

WHERE 句　この句は、FROM 句の中で指定したテーブルから選択するローを制限します。この句を使用して、複数のテーブル間のローを制限できます。

ON 句 (FROM 句の一部) も WHERE 句も両方とも結果セット内のローを制限しますが、WHERE 句はクエリ実行時の後半の段階で適用されるという点においてこれらは異なります。ON 句はテーブル間のジョイン演算の一部ですが、WHERE 句はジョインの完了後に適用されます。一部のクエリでは条件を WHERE 句と ON 句のどちらに指定しても最終的な結果は同じですが、それ以外のクエリでは結果が異なります。たとえば、外部ジョインの場合、WHERE 句に指定した条件の結果は、ON 句に指定した同じ条件の結果と異なります。

search-condition は、サブクエリを含む式から成ります。詳細については、「動的 SQL の探索条件」214 ページを参照してください。

GROUP BY 句　カラム、エイリアス名、または関数によってグループ分けできます。クエリの結果には、指定したカラム、エイリアス、または関数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。NULL を含むローはすべて 1 つのセットとして処理されます。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。集合関数をこれらのグループに適用して、意味のある結果を取得することができます。

group-by-expr は、*select-list* の式の 1 つとまったく同じように作成された (非集合) 式です。

GROUP BY を使う場合、*select-list* と ORDER BY 式が参照できるのは、GROUP BY 句の中で指定した識別子だけです。例外は、*select-list* が集合関数を持つ場合だけです。

HAVING 句　この句は、個々のロー値ではなくグループ値に基づいてローを選択します。HAVING 句を使用できるのは、文に GROUP BY 句があるか、select-list が集合関数のみから成る場合だけです。HAVING 句で参照されるカラム名は、GROUP BY 句に含まれるか、または HAVING 句の集合関数のパラメータとして使用されます。

ORDER BY 句　この句はクエリの結果をソートします。ORDER BY リストの各項目には、昇順の場合 (デフォルト) は ASC、降順の場合は DESC のラベルを付けることができます。式が整数 *n* である場合、クエリの結果は select リストの *n* 番目の項目でソートされます。

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。つまり、ローに最後にアクセスした日付やその他の要因によって、結果セットでの表示順序が異なることがあります。

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。一般に、テーブルのアクセス順序は Ultra Light によって決定するのが最適です。

詳細については、「クエリの最適化」234 ページを参照してください。

使用法

SELECT 文を使用して、データベースからの結果を取り出します。

Ultra Light テーブルには所有者がいません。オプションの *owner* は、既存の SQL とプログラムで生成された SQL の便宜のためにサポートされています。Ultra Light では、*owner* は受け入れられますが無視されます。

参照

『ASA SQL リファレンス・マニュアル』> 「SELECT 文」

例

従業員が何人いるかを示します。

```
SELECT count(*)
 FROM employee
```

UPDATE 文

説明

この文は、データベース・テーブルの既存のローを修正するために使用します。

構文

UPDATE [*owner*.]*table-name*
　SET *column-name* **=** *expression*
　[**WHERE** *search-condition*]

パラメータ

table-name　*table-name* は、更新され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。

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。探索条件の詳細については、「動的 SQL の探索条件」214 ページを参照してください。

警告
WHERE 句を指定しない場合、テーブル内のすべてのローが更新されます。

大文字と小文字の区別　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。文字列 **Value** で更新した CHAR データ型カラムは、常に大文字の V と残りの文字が小文字でデータベースに格納されます。SELECT 文は、文字列を **Value** として返します。ただし、データベースで大文字と小文字が区別されない場合は、すべての比較において **Value** は **value**、**VALUE** などと同じとみなされます。さらに、単一カラムのプライマリ・キーにエントリ **Value** がある場合は、プライマリ・キーがユニークでなくなるので、INSERT 文による **value** の追加は拒否されます。

使用法

UPDATE 文は、テーブル内の値を修正します。

Ultra Light テーブルには所有者がいません。オプションの *owner* は、既存の SQL とプログラムで生成された SQL の便宜のためにサポートされています。Ultra Light では、*owner* は受け入れられますが無視されます。

関連する動作

なし。

参照

- 「INSERT 文」226 ページ
- 「DELETE 文」224 ページ

例

従業員 Philip Chin (従業員 129) を販売部からマーケティング部に異動する例を示します。

```
UPDATE employee
 SET dept_id = 400
 WHERE emp_id = 129
```

クエリの最適化

Ultra Light がクエリを実行する方法は数多くあります。クエリの各実行方法は「**プラン**」と呼ばれます。Ultra Light では、プランは主に、テーブルのアクセス順序と、各テーブルがインデックスを使用して検索されるか、直接ローをスキャンすることによって検索されるかによって定義されます。一部のクエリでは、効率のよいプランと効率の悪いプランの間で実行時間に大きな違いが生じることがあります。

Ultra Light には、**クエリ・オプティマイザ**が含まれています。これは Ultra Light ランタイムの内部コンポーネントであり、別のプランを調べて、効率的なプランを選択しようとします。Ultra Light の最適化の主な目的は、データを効率的な順序でアクセスできるようなインデックスを選択することです。オプティマイザは、テンポラリ・テーブルを使用した中間結果の格納を回避しようとし、2 つのテーブルをジョインするときにテーブルの適切なサブセットのみアクセスされるようにします。

Ultra Light は、クエリを自動的に最適化します。実行時間を調整できる主な領域は、Ultra Light がクエリを最適化するときに利用できるインデックスをデータベースに作成することです。

クエリ・プランの検査

開発時のサポートとして、Ultra Light Interactive SQL を使用して、準備文の実行方法をまとめたプランを表示できます。プランは、ユーティリティの一番下のウィンドウ枠にあるタブに表示されます。プランをグラフィカルに表示するか、テキスト・フォーマットで表示するかを選択できます。

次の文を例にとります。

```
SELECT I.inv_no, I.name, T.quantity, T.prod_no
 FROM Invoice I, Transactions T
 WHERE I.inv_no = T.inv_no
```

下記のようなプランが生成されます。

```
join[scan(Invoice,0),index-scan(Transactions,1)]
```

このプランは、Invoice テーブルのすべてのロー (index[0] の後ろ) を読み込んでから、Transaction テーブルの index[1] を使用して、inv_no カラムが一致するローのみを読み込むことによって、ジョイン演算を実行することを示します。

Ultra Light Interactive SQL の詳細については、「Ultra Light Interactive SQL ユーティリティ」147 ページを参照してください。

オプティマイザの上書き

小型デバイスでも使用できるように、Ultra Light のクエリ最適化は Adaptive Server Anywhere のように広範囲には行われません。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して、クエリに出現する順序でテーブルにアクセスできます。*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。

第 3 部　静的プログラミング・インタフェース

第 4 部では、Embedded SQL、静的型 C++ API、Java 静的プログラミング・インタフェースについて説明します。

静的インタフェースを使用する場合、すべてのクエリをコンパイル時に指定してください。

第 9 章

Palm OS のアプリケーションの開発

この章の内容

この章では、Palm OS のアプリケーションを開発するときの一般的な問題について説明します。

Palm OS のデータベース保管の選択

Palm OS では、Ultra Light データベースと Palm データ・ストア (Palm データベースとも呼ばれます) を区別することが重要です。このマニュアルでは、「**PDB**」という語は、Palm データベースを表し、「**データベース**」という語は、Ultra Light リレーショナル・データベースを表します。Palm データ・ストア以外にも、Palm OS バージョン 4.0 以降も拡張カード上の仮想ファイル・システム (VFS) をサポートしています。

仮想ファイル・システム上の Palm データ・ストア

Ultra Light データベースは、Palm データ・ストアまたは拡張カード上の仮想ファイル・システムに保管できます。保管の指定方法は、使用しているインタフェースによって異なります。

- **Ultra Light for MobileVB** 仮想ファイル・システムを使用するには、VFSOnPalm パラメータを設定します。「VFSOnPalm パラメータ」105 ページを参照してください。
- **Ultra Light C++ コンポーネント、静的型 C++ API、および Embedded SQL** アプリケーションの初期に ULEnablePalmRecordDB または ULEnableFileDB を呼び出します。『Ultra Light C/C++ ユーザーズ・ガイド』> 「ULEnablePalmRecordDB 関数」と『Ultra Light C/C++ ユーザーズ・ガイド』> 「ULEnableFileDB 関数」を参照してください。

保管の詳細

Palm データ・ストアを使用する場合、Ultra Light は実際にはデータベース情報を複数の PDB に保管します。これらの PDB の名前は、指定されている作成者 ID を使用して作成されます。たとえば、ABCD という作成者 ID を使用して作成されたデータベースの場合、次のファイルが作成されます。

- *ul_state_ABCD*
- *ul_udb_ABCD*

Ultra Light は、**ステータス PDB**（*ul_state_ABCD*) を使用して、アプリケーションの終了時に開いている任意のテーブル内でアプリケーションが位置する現在のローを保持します。このようにして、Ultra Light では、アプリケーションを起動したときに、ユーザが最後に作業した場所から再開できるように、アプリケーションを作成できます。

Palm 作成者 ID の概要

すべての Palm OS アプリケーションと同様に、Palm OS の Ultra Light アプリケーションも「**作成者 ID**」を必要とします。この作成者 ID は開発時にアプリケーションに割り当てます。HotSync 同期を使用している場合は、Mobile Link 同期によって使用される作成者 ID を HotSync マネージャに登録します。

作成者 ID をアプリケーションに割り当てる方法については、開発ツールのマニュアルを参照してください。HotSync マネージャでの作成者 ID の登録については、『Mobile Link クライアント』> 「HotSync Manager への Mobile Link HotSync コンジットの登録」を参照してください。

Ultra Light は、作成者 ID を使用してデータベースと HotSync 同期を管理します。多くの場合、Ultra Light による作成者 ID の使用方法の詳細を把握する必要はありません。ただし、複数の Ultra Light データベースに接続し、同期に HotSync を使用するアプリケーションを作成する場合は、Ultra Light による作成者 ID の使用方法の詳細を把握する必要があります。この項では、このようなユーザを対象としています。

Palm OS は、作成者 ID を使用して関連する PDB とアプリケーションを関連付けます。たとえば、CustDB サンプル・アプリケーションの作成者 ID は Syb1 です。Mobile Link コンジットは、この Syb1 作成者 ID を使用して、デバイス上で関連付けられている Ultra Light データベースを検索します。

作成者 ID は、1 ～ 4 文字の文字列です。Palm OS ではシステム・ファイルの頭文字に小文字が使用されるため、先頭の文字を大文字にしてください。

Ultra Light データベースの名前

Ultra Light は、Database On Palm 接続パラメータに指定されている作成者 ID の値に基づいて、新規データベースごとに作成者 ID と PDB 名を割り当てます。また、Database On Palm 接続パラメータが指定されていない場合は、アプリケーションの作成者 ID の値に基づいて割り当てます。

- レコードベースの Palm OS ストアを使用するアプリケーションの場合、PDB 名は ul_udb_*creator-id* です。

- 仮想ファイル・システムを使用するアプリケーションの場合、ファイル名は ul_udb_*creator-id*.udb です。

詳細については、「DatabaseOnPalm 接続パラメータ」93 ページを参照してください。

HotSync と作成者 ID

アプリケーションが複数のデータベースに接続する場合、作成者 ID を使用してアプリケーションとその関連データベースの両方を識別すると、問題が発生することがあります。

HotSync 同期時には、HotSync マネージャは各アプリケーションの作成者 ID を確認します。HotSync マネージャは、Mobile Link コンジットによって登録された作成者 ID を同期用としてコンジットに渡します。このコンジットは、ul_udb_*creator-id* という名前のデータベースを検索します。この場合、*creator-id* は、HotSync マネージャによって指定された名前です。

アプリケーションが 2 つのデータベースに接続する場合は、少なくとも一方のデータベースがアプリケーションとは異なる作成者 ID を持つ必要があります。データベースとアプリケーションの作成者 ID が一致しないため、HotSync マネージャはデータベースに関連付けられているアプリケーションを見つけることができず、同期の対象になりません。

このような制約を回避し、Palm OS 上の複数の Ultra Light アプリケーションに対して HotSync を使用するには、作成者 ID ごとにダミーの Palm アプリケーションを作成します。ダミー Palm アプリケーションでは、処理を実行する必要はありません。ダミー Palm アプリケーションに必要なのは、同期用として HotSync が Mobile Link コンジットに渡し、Mobile Link コンジットが Ultra Light データベース作成者 ID にマッピングできる作成者 ID のみです。このアプローチによって、Mobile Link HotSync コンジットは、Ultra Light データベースの複数のコピーを識別して、それらを同期できます。

第 10 章

Ultra Light 静的インタフェースの使用

この章の内容

この章では、Ultra Light の静的プログラミング・インタフェースに関する概要を説明します。

静的インタフェースを使用する場合、アプリケーションで使用する SQL 文は、コンパイル時に指定します。動的 SQL モデルの場合、SQL 文はランタイム時に指定できます。静的インタフェースには、Embedded SQL、静的型 C++ API、および JDBC を使用する静的型 Java API があります。この章では、すべての静的 Ultra Light インタフェースにおける共通事項を説明します。

概要

この項では、Ultra Light 静的インタフェースの開発環境とプロセスについて説明します。

静的 Ultra Light アプリケーションの開発環境

静的インタフェースを使用して Ultra Light アプリケーションを開発するには、次のツールが必要です。

- **リファレンス・データベース** 作成する Ultra Light データベースのモデルとして機能する Adaptive Server Anywhere データベース。このデータベースは、Sybase Central などのツールを使って開発者が作成します。

  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。テーブルの配列と、テーブル間の外部キー関係の配列は、データベース「**スキーマ**」と呼ばれます。

  Ultra Light データベースをモデル化したら、そのリファレンス・データベースに対して、Ultra Light アプリケーションに含める SQL 文を追加する必要があります。

  詳細については、「リファレンス・データベースの準備」251 ページを参照してください。

- **サポートされる開発ツール** Ultra Light アプリケーションの開発には、標準の開発ツールを使用します。ユーザ・インタフェースなど、Ultra Light 固有でないアプリケーション部分では、通常の方法で開発ツールを使用します。Ultra Light 固有のデータ・アクセス部分では、Ultra Light 開発ツールも使用する必要があります。

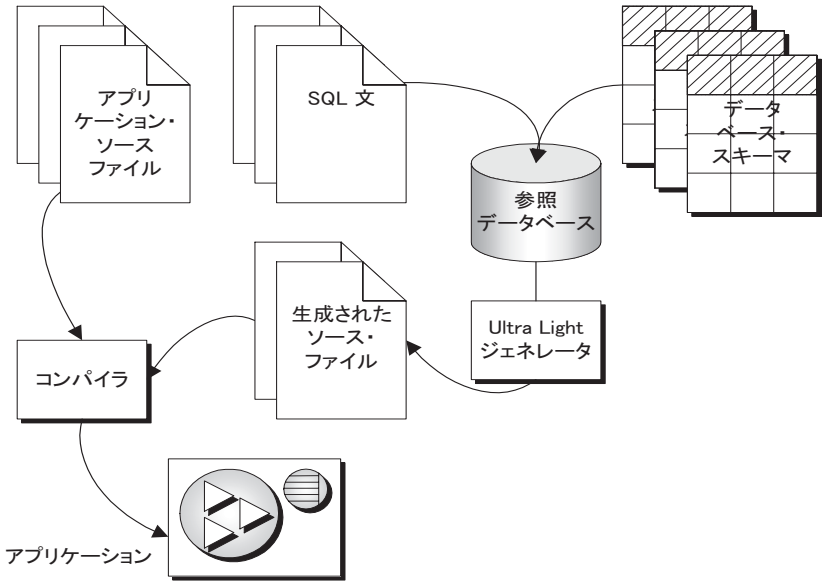
  データ・アクセス・コードは、ユーザ・インタフェースやアプリケーションの内部論理から独立させておくと便利です。

  サポートされているアプリケーション開発ツールの詳細については、『SQL Anywhere Studio の紹介』> 「Ultra Light 開発プラットフォーム」を参照してください。

- **Ultra Light 開発ツール**　Ultra Light には、静的インタフェースを使用して開発を行うためのツールがいくつか装備されています。
  - **Ultra Light ジェネレータ**　このアプリケーションは、アプリケーションの基本となるクエリの実行、データの保管、同期の機能を実装するソース・コードを生成します。静的 SQL を使用した Ultra Light の開発には、ジェネレータが必要です。
  - **SQL プリプロセッサ**　これは、Embedded SQL を使って Ultra Light アプリケーションを開発する場合にだけ必要です。SQL プリプロセッサは Embedded SQL ソース・ファイルを読み込み、標準の C/C++ ファイルを生成します。Embedded SQL ソース・ファイルをスキャンすると同時に、ジェネレータが使用しているリファレンス・データベースに情報を保管します。
- **Ultra Light ランタイム・ライブラリ**　Ultra Light には、各ターゲット・プラットフォームのランタイム・ライブラリが含まれています。一部のプラットフォームでは、このランタイム・ライブラリは静的ライブラリであり、アプリケーション実行プログラムの一部になります。その他のプラットフォームでは、ダイナミック・リンク・ライブラリです。Java の場合、ランタイム・ライブラリは jar ファイルになります。Ultra Light には、ランタイム・ライブラリの使用に必要なヘッダ・ファイルとインポート・ファイルが含まれています。

静的インタフェース Ultra Light 開発プロセス

開発プロセスの基本的な特徴は、すべての静的インタフェースに共通しています。次の図に、主な特徴をまとめます。



- リファレンス・データベースの作成。リファレンス・データベースには、アプリケーションに組み込まれるテーブルのスーパーセットが含まれます。また、アプリケーションの代理データも含まれます。リファレンス・データベースは開発プロセスの一環として必要とされるだけで、完成したアプリケーションには必要ありません。
- SQL 文をリファレンス・データベースの特殊テーブルに追加。選択したインタフェースによって、追加の方法は異なります。
  - 静的型 C++ API または Java を使用している場合は、Sybase Central またはストアド・プロシージャを使用して SQL 文をデータベースに追加します。
  - Embedded SQL を使用している場合は、SQL プリプロセッサによって SQL 文がリファレンス・データベースに追加されます。
- Ultra Light ジェネレータの実行。これにより、SQL 文の実行に必要なコードが含まれたソース・ファイルと、Ultra Light アプリケーションでのデータベース・スキーマの定義に必要なコード

が含まれたソース・ファイルが生成されます。生成されたコードには、Ultra Light ランタイム・ライブラリへの関数呼び出しが含まれています。

- アプリケーションのソース・ファイル作成。Embedded SQL を使用している場合は、SQL プリプロセッサが *.sqc* ファイルを読み込んで、リファレンス・データベースに SQL 文を挿入します。
- アプリケーションのソース・ファイルを、生成されたソース・ファイルとともにコンパイルして、Ultra Light アプリケーションを生成します。

同期機能の追加

ほとんどの Ultra Light アプリケーションには、アプリケーションのデータと統合データベースのデータの統合化を図るため、同期の機能が含まれています。

同期の詳細と使用可能な同期の種類については、『Mobile Link クライアント』>「Ultra Light クライアント」を参照してください。

Ultra Light 静的インタフェースの選択

Ultra Light アプリケーションの開発には、次の 3 つの静的インタフェースを使用できます。

- **静的型 C++ API** 結果セット・ベースの API を使用した、データ・アクセス機能付きの C または C++ を使った開発。
- **Embedded SQL** Embedded SQL 文を使用した、データ・アクセス機能付きの C または C++ を使った開発。
- **静的型 Java API** Java プログラミング言語を使った開発。

開発に Java と C/C++ のどちらを使用するかは、ターゲット・プラットフォームを第一に考慮して決定します。

Embedded SQL と静的型 C++ API のどちらを選択するかについて、検討事項を次に示します。

- Embedded SQL は業界標準のプログラミング手法であり、静的型 C++ API は各社独自の API です。
- Embedded SQL は、アプリケーション設計の細かい制御ができます。Embedded SQL での開発を経験すると、この手法を使うことで効率的なアプリケーションを設計できます。
- 多くのプログラマは、API ベースのプログラミングの方に、より精通しています。このような開発者にとっては、静的型 C++ API を習得する方がより簡単です。
- 静的型 C++ API は、データベースを取り扱うときに、クラスとそれに関連するメソッドを生成します。標準化された関数名を使用するので、開発時間の短縮につながります。

リファレンス・データベースの準備

アプリケーションで Ultra Light データベース・エンジンを実装するには、Ultra Light ジェネレータが Adaptive Server Anywhere リファレンス・データベースにアクセスできるようにします。また、このデータベースには、次の情報が含まれている必要があります。

- **データベース・スキーマ**　Ultra Light アプリケーションで使用するデータベース・オブジェクトです。これには、アプリケーションで使用するテーブルと、そのテーブルのインデックスが含まれます。

  詳細については、「既存のデータベースをリファレンス・データベースとして使用する」254 ページを参照してください。

- **データ**　(オプション) リファレンス・データベースには、Ultra Light データベースに保持したいデータと同様のものを格納できます。このデータは、Ultra Light アナライザがアプリケーションのパフォーマンスを自動的に最適化するときに使用されます。

  詳細については、「既存のデータベースをリファレンス・データベースとして使用する」254 ページを参照してください。

- **クエリ**　Ultra Light システム・テーブルには、アプリケーションで使用する SQL 文を含むようにしてください。

  詳細については、「アプリケーションに SQL 文を定義する」257 ページを参照してください。

- **パブリケーション**　複数の同期オプションを追加する場合に使用します。クエリを定義せずに C++ API アプリケーションを開発するときにも、データベースにパブリケーションを追加します。

  複数の同期オプションの詳細については、『Mobile Link クライアント』> 「別々に同期するためのデータ・セットの設計」 を参照してください。

- **データベース・オプション** 日付フォーマットなどの、アプリケーションに異なる動作をさせることのできるデータベースの動作をさまざまな側面から管理します。Ultra Light データベースは、リファレンス・データベースと同じオプション設定で生成されます。

  さまざまな目的で使用できるように、データベース・オプションはすべてデフォルト設定にしておきます。

  詳細については、「リファレンス・データベースでのデータベース・オプションの設定」253 ページを参照してください。

リファレンス・データベースの作成

Ultra Light アプリケーションを構築するとき、アナライザはリファレンス・データベースをテンプレートとして使用します。

❖ リファレンス・データベースを作成するには、次の手順に従います。

1. 既存の Adaptive Server Anywhere データベースを起動するか、*dbinit* コマンドを使用して新しいデータベースを作成します。

   データベースのアップグレードの詳細については、「既存のデータベースをリファレンス・データベースとして使用する」254 ページを参照してください。

2. アプリケーションで必要なテーブルと外部キー関係を追加します。Sybase Central や Sybase PowerDesigner Physical Architect (SQL Anywhere Studio に含まれている) などの便利なツール、あるいは Sybase PowerDesigner の完全なパッケージなどのさらに強力なデータベース設計ツールを使用できます。

パフォーマンスに関するヒント
リファレンス・データベースにデータを保管する必要はありません。ただし、一般的なアプリケーション・ユーザが保管するデータの見本データをデータベース・テーブルに移植すると、Ultra Light アナライザがこのデータを使用して、アプリケーションのパフォーマンスを自動的に最適化します。

データベースの設計とスキーマの作成については、『ASA SQL ユーザーズ・ガイド』>「データベースの設計」を参照してください。

例

1. データベースを作成します。

   コマンド・プロンプトから、次の文を実行します。

   ```
   dbinit path¥dbname.db
   ```

2. 必要に応じて、Sybase Central を使用して Ultra Light アプリケーションにテーブルを追加します。

3. サンプル・データを追加します。Interactive SQL には、[インポート] のメニュー項目があり、いくつかの一般的なファイル・フォーマットをインポートできます。

   詳細については、『ASA SQL ユーザーズ・ガイド』>「データのインポートとエクスポート」を参照してください。

リファレンス・データベースでのデータベース・オプションの設定

Ultra Light では、オプション値の取得や設定はサポートされません。

Ultra Light アプリケーションを生成した場合、そのコードの動作は、リファレンス・データベースの特定のオプション値により影響を受けます。影響を及ぼすオプションは次のとおりです。

- Date_format
- Date_order
- Nearest_century

- Precision
- Scale
- Time_format
- Timestamp_format

これらのオプションをリファレンス・データベースで設定することにより、Ultra Light データベースの動作を制御できます。リファレンス・データベースのオプション設定は、Ultra Light アプリケーションを生成するときに使用されます。

既存のデータベースをリファレンス・データベースとして使用する

多くの Ultra Light アプリケーションは、Mobile Link を介し、「**統合データベース**」と呼ばれる中央のマスタ・データベースとデータの同期を取ります。リファレンス・データベースと統合データベースを混同しないように注意してください。Ultra Light アプリケーションのリファレンス・データベースは、一般に、統合データベースとは別のデータベースです。

Adaptive Server Anywhere 統合データベースは、リファレンス・データベースとしても使用できる唯一のデータベースです。統合データベースのタイプが異なる場合、Adaptive Server Anywhere リファレンス・データベースを作成します。また、Adaptive Server Anywhere データベースを統合データベースとしている場合でも、Ultra Light アプリケーションで別のスキーマや設定を使用するときは、別のリファレンス・データベースを作成します。

統合データベースの作成と管理には、Adaptive Server Enterprise、Adaptive Server Anywhere、Oracle、Microsoft SQL Server、IBM DB2 など、サポートされている ODBC 準拠のデータベース管理製品の中から任意の製品を選択できます。

統合データベースとして使用している Adaptive Server Anywhere データベースがすでに存在する場合、そのデータベースをコピーして、リファレンス・データベースとして使用できます。

❖ **Adaptive Server Anywhere 以外のデータベースからリファレンス・データベースを作成するには、次の手順に従います。**

1　新しい Adaptive Server Anywhere データベースを作成します。

dbinit コマンドまたは Sybase Central を使用します。

2　統合データベースを参考にして、アプリケーションで必要なテーブルと外部キー関係を追加します。

Sybase Physical Data Architect などのツールを使用して、統合データベースをリエンジニアリングできます。

3　データベース・テーブルに統合データベースからの代理データを移植します。

統合データベースの情報をすべて転送する必要はありません。見本だけを転送します。開発の初期段階では、サンプル・データは必要ありません。運用アプリケーションでは、Ultra Light クエリのアクセス・プランが、リファレンス・データベースのデータ分散に基づいているため、見本データを使用できます。

Adaptive Server Anywhere 以外のデータベースからリファレンス・データベースを作成するための詳細については、『ASA SQL ユーザーズ・ガイド』>「Adaptive Server Anywhere へのデータベースの移行」を参照してください。

クエリ実行の最適化

静的 Ultra Light アプリケーションのパフォーマンスは、次の技術を使用して向上させることができます。

- **インデックスを追加する**　特定の順番で頻繁に情報を取り出している場合は、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。プライマリ・キーには自動的にインデックスが付けられていますが、他のカラムにはインデックスが付けられていません。特に遅いデバイスでは、インデックスによってパフォーマンスが大きく改善されます。

- **見本データを追加する** Adaptive Server Anywhere オプティマイザでは、自動的にクエリのパフォーマンスが最適化されます。また、リファレンス・データベースの情報に基づいて、アクセス・プランが選択されます。アプリケーションのパフォーマンスを改善するためには、アプリケーションが配備された後に保持されるであろうデータを、リファレンス・データベースに入力します。

アプリケーションに SQL 文を定義する

アプリケーションに対するデータ・アクセスの命令は、リファレンス・データベースに SQL 文を追加して定義します。

静的型 C++ API を使用している場合、パブリケーションを使用してデータ・アクセス・メソッドを定義することもできます。パブリケーションの使用の詳細については、『Ultra Light C/C++ ユーザーズ・ガイド』>「Ultra Light テーブルの定義」を参照してください。

Embedded SQL を使用している場合は、SQL プリプロセッサがこの項で説明するタスクを実行します。

Ultra Light プロジェクトの作成

リファレンス・データベースに SQL 文を追加するときに、SQL 文に Ultra Light「**プロジェクト**」を割り当てます。このように SQL 文をグループ分けすることで、同じリファレンス・データベースを使用する複数のアプリケーションを開発できます。

リファレンス・データベースに対して Ultra Light ジェネレータが実行され、データベースのソース・コード・ファイルが生成されると、ジェネレータはプロジェクト名を引数として取り、そのプロジェクトにおける SQL 文のコードを生成します。

Ultra Light プロジェクトは、Sybase Central で定義するか、システムのストアド・プロシージャを呼び出して直接定義できます。

Embedded SQL の場合、SQL プリプロセッサが Ultra Light プロジェクトを定義するため、Ultra Light プロジェクトを明示的に作成する必要はありません。

❖ Ultra Light プロジェクトを作成するには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central で、まだ接続していない場合はデータベースに接続します。
2. 左ウィンドウ枠で、データベース・コンテナを開きます。

3 左ウィンドウ枠で、[Ultra Light プロジェクト] フォルダを開きます。

4 [ファイル] メニューから [新規] ー [Ultra Light プロジェクト] を選択します。

[Ultra Light プロジェクト作成] ウィザードが表示されます。

5 Ultra Light プロジェクト名を入力して [完了] をクリックすると、データベースにプロジェクトが作成されます。

Ultra Light プロジェクトの命名規則については、「ul_add_project システム・プロシージャ」267 ページを参照してください。

❖ Ultra Light プロジェクトを作成するには、次の手順に従います (SQL の場合)。

- Interactive SQL や他のアプリケーションから、次のコマンドを入力します。

```
call ul_add_project( 'project-name' )
```

ここで、*project-name* はプロジェクトの名前です。

詳細については、「ul_add_project システム・プロシージャ」267 ページを参照してください。

❖ Ultra Light プロジェクトを作成するには、次の手順に従います (Embedded SQL の場合)。

- Embedded SQL インタフェースの場合、SQL プリプロセッサのコマンド・ラインに Ultra Light プロジェクト名を指定すると、プリプロセッサによってプロジェクトがデータベースに追加されます。

詳細については、『Ultra Light C/C++ ユーザーズ・ガイド』>「Embedded SQL アプリケーションの構築」を参照してください。

説明

Ultra Light プロジェクト名は、データベース識別子としての規則に従っている必要があります。プロジェクト名にスペースを使用する場合、名前を二重引用符で囲まないでください。二重引用符は、Sybase Central またはストアド・プロシージャによって追加されます。

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

Ultra Light プロジェクトに SQL 文を追加する

Ultra Light アプリケーションごとに、データ・アクセス要求が実行されます。データ・アクセス要求はインタフェースごとに異なった方法で実装されますが、定義する方法はどのモデルでも同じです。

Ultra Light アプリケーションが実行できるデータ・アクセス要求を定義するには、リファレンス・データベースで、そのアプリケーションの Ultra Light プロジェクトに SQL 文のセットを追加します。SQL 文が追加されると、Ultra Light ジェネレータによって SQL 文のセットを実行するデータベース・エンジンのコードが作成されます。

静的型 C++ API では、パブリケーションを使用してデータ・アクセス・メソッドを定義することもできます。パブリケーションの使用の詳細については、『Ultra Light C/C++ ユーザーズ・ガイド』> 「Ultra Light テーブルの定義」を参照してください。

SQL 文は、Sybase Central で Ultra Light プロジェクトに追加するか、システムのストアド・プロシージャを呼び出して直接追加します。Embedded SQL では、SQL プリプロセッサが Embedded SQL のソース・ファイルにある SQL 文をリファレンス・データベースに追加します。

❖ SQL 文を Ultra Light プロジェクトに追加するには、次の手順に従います (Sybase Central の場合)。

1. Sybase Central で、まだ接続していない場合はデータベースに接続します。
2. 左ウィンドウ枠で、データベース・コンテナを開きます。

3 左ウィンドウ枠で、[Ultra Light プロジェクト] フォルダを開きます。

4 アプリケーションのプロジェクトを開きます。

5 [ファイル] メニューから [新規] ー [Ultra Light 文] を選択します。

[新しい Ultra Light 文の作成] ウィザードが表示されます。

6 SQL 文の短い記述名を入力し、[次へ] をクリックします。

7 SQL 文を入力して [完了] をクリックすると、SQL 文がプロジェクトに追加されます。

文を右クリックし、ポップアップ・メニューから [Interactive SQL から実行] を選択して、データベースに対して SQL 文をテストできます。

使用できる SQL 文の種類については、「Ultra Light SQL 文の記述」261 ページを参照してください。

❖ SQL 文を Ultra Light プロジェクトに追加するには、次の手順に従います (SQL の場合)。

- Interactive SQL や他のアプリケーションから、次のコマンドを入力します。

```
call ul_add_statement( 'project-name',
     'statement-name',
     'sql-statement' )
```

ここで、*project-name* はプロジェクト名、*statement-name* は短い記述名、*sql-statement* は実際の SQL 文を表します。

詳細については、「ul_add_statement システム・プロシージャ」266 ページを参照してください。

❖ **SQL 文を Ultra Light プロジェクトに追加するには、次の手順に従います (Embedded SQL の場合)。**

- Embedded SQL インタフェースの場合、Ultra Light プロジェクト名は SQL プリプロセッサのコマンド・ラインに指定します。

  Embedded SQL の開発では、SQL 文は使用しません。

  詳細については、『Ultra Light C/C++ ユーザーズ・ガイド』>「Embedded SQL アプリケーションの構築」を参照してください。

説明

SQL 文には、簡潔でわかりやすい名前を付けます。この名前は、Ultra Light ジェネレータによって、Java または C++ API で使用する SQL 文の識別に利用されます。たとえば、**ProductQuery** という名前の SQL 文は、**ProductQuery** という名前の C++ API クラスと、**PRODUCT_QUERY** という名前の Java 定数を生成します。名前は有効な SQL 識別子にしてください。

SQL 文の構文は、SQL 文をデータベースに追加するときにチェックされます。構文エラーは、エラー・メッセージにより確認できます。

プロジェクト内の SQL 文の更新は、Sybase Central または ul_add_statement を使用して SQL 文の追加と同じ方法で行います。SQL 文がすでに存在する場合は、新しい構文によって上書きされます。SQL 文を修正するたびに、Ultra Light コードを再生成します。

Ultra Light SQL 文の記述

この項では、Ultra Light プロジェクトに追加できる SQL 文と、SQL 文でプレースホルダを使用する方法を説明します。

使用できる SQL の範囲については、「Ultra Light の SQL サポートの概要」180 ページを参照してください。

二重引用符を追加する方法

Sybase Central に入力する SQL 文や、**ul_add_statement** の引数として入力する SQL 文は、リファレンス・データベースに文字列として追加されます。したがって、SQL 文は SQL 文字列の規則に従っている必要があります。

SQL 文の中には、円記号を使用してエスケープすることが必要な文字があります。

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

SQL 文で変数を使用する

INSERT 文や UPDATE 文では、通常、事前に値がわかっているわけではありません。変数のプレースホルダとして疑問符を使用し、ランタイムに値を指定します。

```
call ul_add_statement(
     'ProductApp',
     'AddCap',
     'INSERT INTO ¥"DBA¥".product ( id, name, price )
         VALUES( ?, ?, ? )'
 )
```

プレースホルダは、クエリの WHERE 句でも使用できます。

```
call ul_add_statement(
     'ProductApp',
     'ProductQuery',
     'SELECT id, name, price
      FROM ¥"DBA¥".product
      WHERE price > ?'
 )
```

円記号を使用して、二重引用符をエスケープします。

Embedded SQL では、プレースホルダに「**ホスト変数**」を使用します。詳細については、『Ultra Light C/C++ ユーザーズ・ガイド』> 「ホスト変数の使用」を参照してください。

プレースホルダを含む SQL 文では、生成された C++ クラスの **Open** または **Execute** メソッドに関する追加のパラメータが、各パラメータに対して定義されます。Java アプリケーションでは、JDBC セット・メソッドを使用してパラメータへ値を割り当てます。

Ultra Light データ・アクセス・コードの生成

Ultra Light データベースへのアクセスやデータ保管を行うコードを生成するため、「**Ultra Light ジェネレータ**」がアプリケーションで使用するリファレンス・データベースと SQL 文を解析します。Ultra Light ジェネレータはコマンド・ライン・アプリケーションです。プロジェクトごとに動作をカスタマイズするために、コマンド・ライン・オプションのセットをとります。たとえば、指定したコマンド・ライン・オプションによって C/C++ または Java コードを生成することができます。

データ保管コードには、リファレンス・データベースのテーブルとカラムのうち、アプリケーションで使用されるものだけが含まれます。また、リファレンス・データベースにあるインデックスを使用するとアプリケーションの効率が向上する場合には、Ultra Light ジェネレータはそのインデックスをデータ保管コードに含めます。

データ・アクセス・コードには、リファレンス・データベースでプロジェクトに追加した SQL 文だけが含まれます。

結果として、アプリケーションに適合したカスタム・データベース・エンジンが作成されます。Ultra Light ジェネレータはアプリケーションで使用する機能だけを含めるため、このエンジンは、汎用のデータベース・エンジンと比べてかなり小さくなります。

Ultra Light ジェネレータの詳細については、「Ultra Light ジェネレータ」115 ページを参照してください。

開発ツールを静的 Ultra Light 開発用に設定する

ほとんどの開発ツールは依存モデルを使用しています。これは makefile として表現されることもあり、ソース・ファイルのタイムスタンプが、ターゲット・ファイル (ほとんどの場合、オブジェクト・ファイル) のタイムスタンプと比較され、ターゲット・ファイルを再生成すべきかどうかが決定されます。

Ultra Light の開発では、開発プロジェクトで SQL 文が変更されると、生成コードを再生成する必要があります。SQL 文はリファレンス・データベースに保存されるため、個々のソース・ファイルのタイムスタンプには変更が反映されません。

依存ベースの開発環境に Ultra Light プロジェクトを追加する方法については、『Ultra Light C/C++ ユーザーズ・ガイド』> 「開発ツールを Embedded SQL 開発用に設定する」を参照してください。

第 11 章

Ultra Light 静的インタフェースのリファレンス

この章の内容

この章では、Embedded SQL、静的型 C++ API、静的型 Java API などの Ultra Light 静的インタフェースのリファレンス情報について説明します。

リファレンス・データベースのストアド・プロシージャ

この項では、Adaptive Server Anywhere リファレンス・データベースのシステム・ストアド・プロシージャについて説明します。プロシージャは、SQL 文をプロジェクトに追加するのに使用します。

プロシージャを使って追加された各 SQL 文に対して、Ultra Light ジェネレータが C++ または Java クラスを定義します。

これらのシステム・プロシージャは、組み込みユーザ ID **dbo** が所有しています。

ul_add_statement システム・プロシージャ

機能　SQL 文を Ultra Light プロジェクトに追加します。

構文

```
ul_add_statement (in @project char(128),
    in @name char(128),
    in @statement text )
```

パーミッション　DBA 権限が必要です。

関連する動作　なし。

参照

「ul_add_project システム・プロシージャ」267 ページ

「ul_delete_statement システム・プロシージャ」268 ページ

説明　Ultra Light プロジェクトに文を追加または修正します。

project　この文を追加する Ultra Light プロジェクト。Ultra Light ジェネレータはプロジェクト内のすべての文にクラスを一度に定義します。

name　文の名前。この名前は、生成されたクラスで使用されます。

statement　SQL 文を含む文字列。

同じプロジェクト内に同じ名前の文がある場合は、新しい構文で更新されます。*project* が存在しない場合は、作成されます。

例　次の呼び出しは、文を TestSQL プロジェクトに追加します。

```
call ul_add_statement(
 'TestSQL', 'TestQuery',
 'select prod_id, price, prod_name from ulproduct where
price < ?' )
```

ul_add_project システム・プロシージャ

機能　Ultra Light プロジェクトを作成します。

構文　**ul_add_project** (in **@project** char(128))

パーミッション　DBA 権限が必要です。

関連する動作　なし。

参照　「ul_delete_statement システム・プロシージャ」268 ページ

説明　Ultra Light プロジェクトをデータベースに追加します。プロジェクトは、アプリケーション内の SQL 文のコンテナとして動作します。プロジェクト名は、Ultra Light ジェネレータのコマンド・ラインで指定して、プロジェクト内のすべての文にクラスが定義されるようにします。

project　Ultra Light プロジェクト名。

例　次の呼び出しは、**Product** というプロジェクトをデータベースに追加します。

```
call ul_add_project( 'Product' )
```

ul_delete_project システム・プロシージャ

機能　Ultra Light プロジェクトをデータベースから削除します。

構文 **ul_delete_project** (in **@project** char(128))

パーミッション DBA 権限が必要です。

関連する動作 なし。

参照 「ul_add_project システム・プロシージャ」267 ページ

「ul_delete_statement システム・プロシージャ」268 ページ

説明 Ultra Light プロジェクトをデータベースから削除します。

project データベースから削除される Ultra Light プロジェクト。

例 次の呼び出しは、**Product** プロジェクトを削除します。

```
call ul_delete_project( 'Product' )
```

ul_delete_statement システム・プロシージャ

機能 SQL 文を Ultra Light プロジェクトから削除します。

構文 **ul_delete_statement** (in **@project** char(128),
in **@name** char(128))

パーミッション DBA 権限が必要です。

関連する動作 なし。

参照 「ul_add_project システム・プロシージャ」267 ページ

「ul_add_statement システム・プロシージャ」266 ページ

説明 Ultra Light プロジェクトから文を削除します。

project 削除する文がある Ultra Light プロジェクト。

name 文の名前。この名前は、生成されたクラスで使用されます。

例 次の呼び出しは、文を **Product** プロジェクトから削除します。

```
call ul_delete_statement( 'Product', 'AddProd' )
```

ul_set_codesegment システム・プロシージャ

機能

C++ API を使用している Palm Computing Platform 開発の場合は、Ultra Light プロジェクトから特定のセグメントに SQL 文を割り当てます。

構文

ul_set_codesegment(in **@project** char(128),
in **@name** char(128), in **@segment_name** *char(8)*)

関連する動作

なし。

参照

「ul_add_statement システム・プロシージャ」266 ページ

『Ultra Light C/C++ ユーザーズ・ガイド』> 「セグメントの明示的な割り当て」

説明

C++ API の SQL 文について生成されるコードを、指定の Palm セグメントに明示的に割り当てます。

project　文を適用する Ultra Light プロジェクト。

name　「ul_add_statement システム・プロシージャ」266 ページに定義されている文の名前。

segment_name　文を割り当てるセグメントの名前。

例

次の呼び出しは、プロジェクト **myproject** 内の文 **mystmt** をセグメント **MYSEG1** に割り当てます。

```
call ul_set_codesegment(
      'myproject', 'mystmt', 'MYSEG1' )
```

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

て

と

な

に

は

ひ

ふ

へ

ほ

ま

め

も

や

ゆ

よ

ら

り

れ

ろ

わ