iAnywhere SOLUTIONS™
A SYBASE COMPANY

Ultra Light for MobileVB
ユーザーズ・ガイド

パート番号：DC37122-01-0902-01
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、Ultra Light for MobileVB について説明します。Ultra Light for MobileVB を使用すると、Palm OS または Windows CE を搭載しているハンドヘルド、モバイル、または埋め込みデバイスに対してデータベース・アプリケーションを開発、配備できます。

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 AppForge MobileVB および AppForge Crossfire アプリケーション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** }]

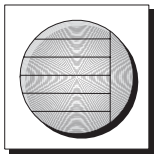
  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

グラフィック・アイコン

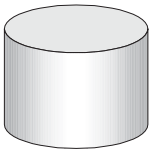
このマニュアルでは、次のアイコンを使用します。

- 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

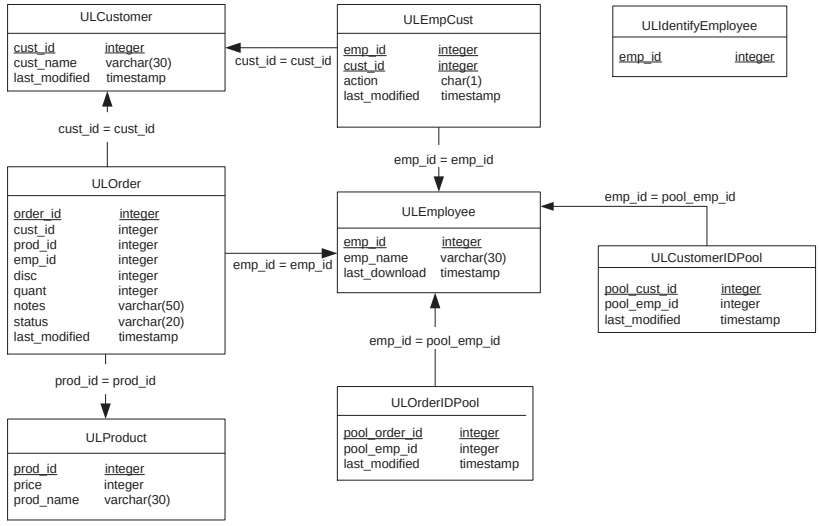
CustDB サンプル・データベース

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 章

Ultra Light for MobileVB の概要

この章の内容

この章では、Ultra Light for MobileVB について説明します。ここでは、『Ultra Light データベース・ユーザーズ・ガイド』> 「Ultra Light へようこそ」で説明した Ultra Light の機能について理解していることを前提としています。

Ultra Light for MobileVB の特徴

Ultra Light for MobileVB は、モバイル・デバイスのためのリレーショナル・データ管理システムです。ビジネス・アプリケーションに必要なパフォーマンス、リソース効率、堅牢性、セキュリティを備えています。Ultra Light では、エンタープライズ・データ・ストアとの同期も提供されます。

システムの稼働条件とサポートされるプラットフォーム

開発プラットフォーム

Ultra Light for MobileVB を使用してアプリケーションを開発するには、以下が必要です。

- Microsoft Visual Basic .NET または Visual Basic 6。

  お使いの AppForge MobileVB または AppForge Crossfire のバージョンの要件に合うサービス・パックをインストールしてください。詳細については、AppForge の Web サイトを参照してください。Visual Basic 6 を使用している場合は、少なくともサービス・パック 5 をインストールすることをおすすめします。

  AppForge Booster

  Ultra Light for MobileVB を使用してアプリケーションを配備するには、AppForge Booster が必要です。AppForge Booster がない場合は、//www.appforge.com/booster.html から入手できます。Ultra Light アプリケーションには、BoosterPlus は必要ありません。

- AppForge MobileVB Version 3.x 以降、または AppForge Crossfire。

  互換性

  3.0 より前のバージョンの MobileVB を使用し、ARM デバイス上の Windows CE 用に開発している場合は、SQL Anywhere ディレクトリの下の *ultralite¥UltraLiteForMobileVB¥ce¥arm¥ulmvb9.dll* をデバイスの *¥Program Files¥AppForge* ディレクトリにコピーしてください。

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。

ターゲット・プラットフォーム

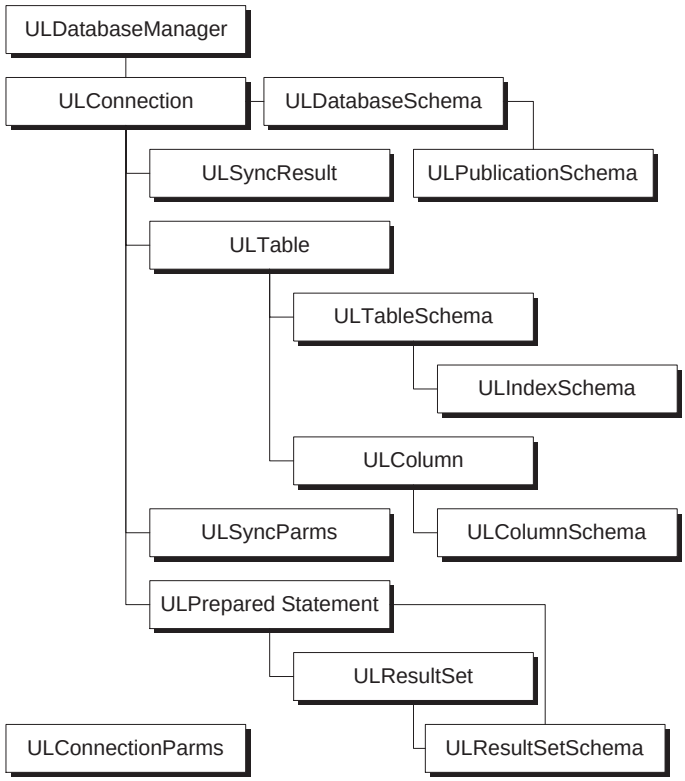
Ultra Light for MobileVB は、次のターゲット・プラットフォームをサポートしています。

- ARM プロセッサと MIPS プロセッサの Pocket PC に搭載した Windows CE 3.0 以上。
- Palm OS バージョン 3.5 以降。

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

Ultra Light for MobileVB のアーキテクチャ

Ultra Light プログラミング・インタフェースは、Ultra Light データベースを使用したデータ操作のためのオブジェクト・セットを公開しています。次の図は、オブジェクト階層を示します。



次のリストは、よく使用される高度なオブジェクトの一部を示します。

- **ULDatabaseManager** Ultra Light データベースへの接続を管理します。

詳細については、「ULDatabaseManager クラス」131 ページを参照してください。

- **ULConnectionParms**　接続パラメータのセットを格納します。

  Connection Parameters コントロールを使用し、Visual Basic プロパティ・シートに接続パラメータを指定できます。

  詳細については、「ULConnectionParms クラス」126 ページを参照してください。

- **ULConnection**　データベース接続を表し、トランザクションを管理します。

  詳細については、「ULConnection クラス」112 ページを参照してください。

- **ULPreparedStatement、ULResultSet、ULResultSetSchema**　SQL を使用してデータベース要求とその結果を管理します。

  詳細については、「ULPreparedStatement クラス」145 ページ、「ULResultSet クラス」153 ページ、「ULResultSetSchema クラス」161 ページを参照してください。

- **ULTable と ULColumn**　テーブル・ベースの API を使用してデータを管理します。

  詳細については、「ULTable クラス」185 ページと「ULColumn クラス」103 ページを参照してください。

- **ULSyncParms と ULSyncResult**　Mobile Link 同期サーバを介して同期を管理します。

  Mobile Link との同期の詳細については、『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。

第 2 章

Ultra Light for MobileVB の開発の概要

この章の内容

この章では、Ultra Light for MobileVB を使用してアプリケーションを開発する方法について説明します。

チュートリアルについては、「チュートリアル：Ultra Light for MobileVB アプリケーションのサンプル」55 ページを参照してください。

Ultra Light for MobileVB を使用するための準備

以下の手順では、Ultra Light for MobileVB を使用してアプリケーションを構築する前に実行する必要のある操作を示します。

MobileVB 設計環境への Ultra Light の追加

MobileVB または Crossfire プロジェクトから Ultra Light コントロールにアクセスするには、Ultra Light for MobileVB を設計環境に追加します。

❖ **Ultra Light 接続パラメータ・コントロールを追加するには、次の手順に従います。**

1 Visual Basic メニューから、[プロジェクト] ー [コンポーネント] を選択します。

2 [コントロール] タブをクリックします。

3 リストを下にスクロールして [UltraLite Connection Parameters 9.0] を選択します。[OK] をクリックします。

使用可能なコントロールのリストにこの項目が表示されない場合は、次の手順を実行します。

- MobileVB を閉じ、プロジェクトを保存します。
- *ultralite¥UltraLiteforMobileVB¥win32* でコマンド・プロンプトを開き、次のコマンドを実行します。

```
ulmvbreg -r
```

- MobileVB を再起動し、プロジェクトを開きます。
- [プロジェクト] ー [コンポーネント] を選択します。
- [UltraLite Connection Parameters 9.0] を選択します。

データベース・アイコンがツールバーに追加されます。このアイコンをダブルクリックして、ULConnectionParms オブジェクトをフォームに追加します。

Ultra Light for MobileVB への参照の追加

SQL Anywhere Studio がインストールされている場合、Ultra Light for MobileVB は新しい MobileVB プロジェクトに自動的に追加されます。このため、通常は Ultra Light for MobileVB への参照をプロジェクトに手動で追加する必要はありません。次の手順は、SQL Anywhere Studio のインストール後に MobileVB をインストールする場合など、参照を手動で追加する必要がある特別なケースに使用します。

❖ Ultra Light for MobileVB への参照を追加するには、次の手順に従います。

1. Visual Basic メニューから、[プロジェクト] ー [参照設定] を選択します。
2. [iAnywhere Solutions, UltraLite for MobileVB 9.0] コントロールが利用可能な参照のリストに含まれている場合、それを選択して [OK] をクリックします。

   [iAnywhere Solutions, UltraLite for MobileVB 9.0] コントロールが利用可能な参照のリストにない場合は、次の手順に従います。

   - *ultralite¥UltraLiteforMobileVB¥win32* でコマンド・プロンプトを開き、次のコマンドを実行します。

     ```
     ulmvbreg -r
     ```

   - [iAnywhere Solutions, UltraLite for MobileVB 9.0] を選択し、[OK] をクリックします。

Crossfire 設計環境への Ultra Light の追加

SQL Anywhere Studio のセットアップ・プログラムでは、Ultra Light が Crossfire 設計環境に自動的に追加されますが、Ultra Light を環境に手動で追加することが必要な場合があります。たとえば、SQL Anywhere Studio のインストール後に Crossfire をインストールした場合は、この手順を実行することが必要な場合があります。

Ultra Light を Crossfire に追加する必要があるかどうかを調べるには、新しい Crossfire プロジェクトに iAnywhere.UltraLiteForAppForge への参照が含まれているかどうかをチェックします。含まれていない場合は、Ultra Light を環境に追加する必要があります。また、ULConnectionParms クラスがツールボックスの AppForge パネルに表示されるかどうかをチェックします。表示されていない場合は、Ultra Light を環境に追加する必要があります。

❖ Crossfire プロジェクトに Ultra Light の参照とコントロールを追加するには、次の手順に従います。

1. Ultra Light for MobileVB を Crossfire に登録します。
   a. Crossfire が閉じていることを確認します。
   b. SQL Anywhere インストール環境の *ultralite¥UltraLiteforMobileVB¥win32* サブディレクトリでコマンド・プロンプトを開き、次のコマンドを実行します。

      ```
      ulmvbreg -r
      ```

   c. MobileVB プロジェクトをアップグレードした場合は、Visual Basic.NET ソリューション・エクスプローラから UltraLiteAFLib への参照を削除します。
   d. *iAnywhere.UltraLiteForAppForge.dll* への参照を追加します。
      i. [Microsoft Development Environment] メニューから、[プロジェクト] ー [参照の追加] を選択し、SQL Anywhere インストール環境の *ultralite¥UltraLiteforMobileVB¥win32* サブディレクトリを参照します。

ii. *iAnywhere.UltraLiteForAppForge.dll* を選択し、[開く] をクリックします。

iii. [OK] を選択して参照を追加します。

2 AppForge ツールボックスに ULConnectionParms コントロールを追加します。

a. Microsoft Development Environment で、AppForge ツールボックスを右クリックし、[Add/Remove Items] を選択します。ダイアログが表示されます。

b. [COM Components] タブをクリックします。

c. [ULConnectionParms Class] という名前のエントリにスクロールします。このコンポーネントの横のボックスをオンにして、[OK] をクリックします。

d. ULConnectionParms コントロールがツールボックスに追加されます。

データベース・スキーマの使用

スキーマは、データベースの構造です。スキーマは、データベース内のテーブル定義、インデックス定義、パブリケーション定義の集合であり、それらの間のすべての関係を示します。

Ultra Light データベースを作成するには、アプリケーションの関数を呼び出して、Ultra Light データベース・スキーマ・ファイルを作成し、このファイルをデータベースに適用します。

Ultra Light データベース・スキーマ・ファイルの作成の詳細については、以下を参照してください。

Ultra Light データベース・スキーマ・ファイルの作成

Ultra Light スキーマ・ペインタまたは *ulinit* ユーティリティを使用して、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**　Ultra Light スキーマ・ペインタは、Ultra Light スキーマ・ファイルの作成と編集に使用するグラフィカル・ユーティリティです。

  スキーマ・ペインタを起動するには、[スタート] － [プログラム] － [SQL Anywhere 9] － [Ultra Light] － [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 (*.usm*) をダブルクリックします。

  Ultra Light スキーマ・ペインタの使用の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「レッスン 1 : Ultra Light データベース・スキーマの作成」を参照してください。

- **ulinit ユーティリティ**　Adaptive Server Anywhere データベース管理システムがある場合は、*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。

  ulinit ユーティリティの使用の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「ulinit ユーティリティ」を参照してください。

データベースのスキーマの変更

既存のデータベースのスキーマを変更するには、新しいスキーマを使用してスキーマ・ファイルを作成し、このスキーマを既存のデータベースに適用します。ほとんどの場合、データ損失は起こりません。ただし、カラムを削除したり、カラムのデータ型が互換性のない型に変更された場合などは、データ損失が起こる場合があります。

これらのメソッドの詳細については、「ApplyFile メソッド」140 ページと「ApplyFileWithParms メソッド」140 ページを参照してください。

新規スキーマ・ファイルの配備の準備については、『Ultra Light データベース・ユーザーズ・ガイド』>「Ultra Light データベース・スキーマのアップグレード」を参照してください。

例

次のコードで、新規スキーマ・ファイルが適用されます。

```
Connection.Schema.ApplyFile("schema_file=¥My
Documents¥myschema.usm")
```

Ultra Light データベースへの接続

Ultra Light アプリケーションをデータベースに接続しないと、データベースのデータを操作できません。この項では、Ultra Light データベースに接続する方法について説明します。

ULConnection オブジェクトの使用

ULConnection オブジェクトの次のプロパティは、アプリケーションのグローバルな動作を管理します。

ULConnection オブジェクトの詳細については、「ULConnection クラス」112 ページを参照してください。

- **コミット動作** デフォルトでは、Ultra Light アプリケーションは AutoCommit モードに設定されています。insert 文、update 文、delete 文はすべて、すぐにデータベースにコミットされます。ULConnection.AutoCommit を false に設定し、アプリケーションにトランザクションを構築します。AutoCommit を off に設定しコミットを実行すると、アプリケーションのパフォーマンスを直接向上できます。

  詳細については、「Commit メソッド」115 ページを参照してください。

- **ユーザ認証** GrantConnectTo メソッドと RevokeConnectFrom メソッドを使用すると、アプリケーションのユーザ ID とパスワードをデフォルト値の DBA と SQL から別の値に変更できます。

  詳細については、「ユーザの認証」41 ページを参照してください。

- **同期** 同期を管理するオブジェクトのセットは、ULConnection オブジェクトからアクセスできます。

  詳細については、「データの同期」42 ページを参照してください。

- **テーブル** ULConnection.GetTable メソッドを使用して、Ultra Light テーブルにアクセスします。

  詳細については、「GetTable メソッド」116 ページを参照してください。

データベースへの接続

ULConnectionParms オブジェクトまたは接続文字列を使用して、データベースに接続できます。ULConnectionParms オブジェクトを使用するメソッドでは、接続パラメータを簡単かつ正確に操作できます。接続文字列を使用するメソッドの場合、接続文字列を正しく作成することが要求されます。

次の手順では、ULConnectionParms オブジェクトを使用して Ultra Light データベースに接続します。

ULConnectionParms オブジェクトを使用した Ultra Light データベースへの接続の詳細については、「CreateDatabaseWithParms メソッド」133 ページと「OpenConnectionWithParms メソッド」137 ページを参照してください。

❖ ULConnectionParms を使用して Ultra Light データベースに接続するには、次の手順に従います。

1　ULDatabaseManager オブジェクトを作成します。

DatabaseManager オブジェクトは、1 つのアプリケーションに 1 つだけ作成してください。このオブジェクトは、オブジェクト階層のルートにあります。そのため、DatabaseManager オブジェクトは、アプリケーションに対してグローバルに、またはクラスレベル変数として宣言するのが最も効果的です。

```
'MobileVB
 Public DatabaseMgr As ULDatabaseManager
 Set DatabaseMgr = New ULDatabaseManager

'Crossfire
 Public DatabaseMgr As New
UltraLiteAFLib.ULDatabaseManager
```

2　ULConnection オブジェクトを宣言します。

ほとんどのアプリケーションは、Ultra Light データベースへの単一接続を使用し、接続を常時開いています。そのため、ULConnection オブジェクトは、アプリケーションに対してグローバルに宣言するのが最も効果的です。

```
'MobileVB
 Public Connection As New ULConnection
```

```
'Crossfire
 Public Connection As
UltraLiteAFLib.ULDatabaseManager
```

3 ULConnectionParms オブジェクトを作成します。

MobileVB ツール・パレットの ULConnectionParms オブジェクトをダブルクリックします。ULConnectinParms オブジェクトがフォームに表示されます。

4 ULConnectionParms オブジェクトの必須プロパティを設定します。

ULConnectionParms プロパティ・ウィンドウで、データベースのロケーション、スキーマ・ファイル、ユーザ名、パスワードなどのプロパティを指定します。

次のプロパティを使用して、CreateDatabaseWithParms のスキーマ・ファイルまたは OpenConnectionWithParms のデータベース・ファイルを指定します。追加プロパティの詳細については、「プロパティ」126 ページを参照してください。

キーワード	説明
DatabaseOnCE	Windows CE 上の Ultra Light データベースのパスとファイル名です。
DatabaseOnDesktop	デスクトップ・マシン上の Ultra Light データベースのパスとファイル名です。
SchmaOnCE	Windows CE 上の Ultra Light スキーマのパスとファイル名です。
SchmeOnDesktop	デスクトップ・マシン上の Ultra Light スキーマのパスとファイル名です。

5 データベースへの接続を開きます。

CreateDatabaseWithParms および OpenConnectionWithParms は、開いた接続を ULConnection オブジェクトとして返します。各メソッドは、1 つの ULConnectionParms オブジェクトを引数として取ります。

次のコードは、既存のデータベースに接続しようとします。データベースが存在しない場合、OpenConnectionWithParms はエラーを返します。これにより CreateDatabaseWithParms は、指定されたスキーマ・ファイルを使用してデータベースを作成します。

Crossfire では、GetOcx メソッドを ULConnectionParms オブジェクトに含めてください。

```
'MobileVB
 On Error Resume Next
 Set Connection =
DatabaseMgr.OpenConnectionWithParms( LoginParms)
 If Err.Number <> ULSQLCode.ulSQLE_NOERROR Then
   Set Connection =
DatabaseManager.CreateDatabaseWithParms( LoginParms
)
 End If

'Crossfire
 Try
   Connection = _
     DatabaseMgr.OpenConnectionWithParms( _
         ULConnectionParms1.GetOcx)
 Catch
   If Err.Number = _

UltraLiteAFLib.ULSQLCode.ulSQLE_ULTRALITE_DATABASE_
NOT_FOUND _
   Then
     Err.Clear()
     Connection = _
       DatabaseMgr.CreateDatabaseWithParms( _
       ULConnectionParms1.GetOcx)
 End Try
```

暗号化と難読化

Ultra Light for MobileVB を使用する場合、Ultra Light データベースを暗号化または難読化できます。

暗号化

暗号化を使用してデータベースを作成するには、ULConnectionParms.EncryptionKey プロパティを設定します。CreateDatabaseWithParms を呼び出し、ConnectionParms オブジェクトを渡すと、作成されるデータベースは、指定されたキーを使用し暗号化されます。

EncryptionKey プロパティの詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「EncryptionKey 接続パラメータ」と「ChangeEncryptionKey メソッド」114 ページを参照してください。

例

暗号化キーを変更するには、新しい暗号化キーを Connection オブジェクトで指定します。この例では「apricot」が暗号化キーです。

```
Connection.ChangeEncryptionKey("apricot")
```

データベースが暗号化された後は、データベースへの接続で正しい暗号化キーを指定する必要があります。そうしないと、接続は失敗します。

難読化

データベースを難読化するには、作成パラメータとして obfuscate=1 を指定します。

データベース暗号化の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「Ultra Light データベースの暗号化」を参照してください。

例

次のコードは、新しいデータベースを難読化します。

```
open_parms =
"ce_file=¥tutorial.udb;ce_schema=¥tutorial.usm;obfuscat
e=1"
 Set Connection = DatabaseManager.CreateDatabase(
open_parms )
```

動的 SQL を使用したデータの使用

Ultra Light アプリケーションは、動的 SQL またはテーブル API を使用して、テーブル・データにアクセスできます。この項では、動的 SQL を使用したデータ・アクセスについて説明します。

テーブル API の詳細については、「テーブル API を使用したデータの使用」27 ページを参照してください。

この項では、動的 SQL を使用して次の操作を行う方法を説明します。

- テーブルのローのスクロール
- 現在のローの値へのアクセス
- テーブルのローの検索
- ローの挿入、削除、更新

この項では、SQL 言語そのものについては説明しません。動的 SQL 機能の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「動的 SQL」を参照してください。

動的 SQL に必要な操作手順は、SQL の操作と変わりません。概要については、『Ultra Light データベース・ユーザーズ・ガイド』>「動的 SQL の使用」を参照してください。

データ操作：INSERT、UPDATE、DELETE

Ultra Light では、SQL データ操作言語の操作を実行できます。これらの操作は、ULPreparedStatement クラスのメンバである ExecuteStatement メソッドを使用して実行します。

ULPreparedStatement クラスの詳細については、「ULPreparedStatement クラス」145 ページを参照してください。

準備文でのパラメータの使用
パラメータのプレースホルダは、? 文字を使用して指定します。INSERT、UPDATE、DELETE では必ず、各 ? は準備文での並び順を参照します。たとえば、最初の ? は 1、2 番目の ? は 2、のようになります。

❖ ローを挿入するには、次の手順に従います。

1 ULPreparedStatement オブジェクトを宣言します。

```
'MobileVB
 Dim PrepStmt As ULPreparedStatement

'Crossfire
 Dimr PrepStmt As UltraLiteAFLib.ULPreparedStatement
```

2 INSERT 文を準備文オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

```
'MobileVB
 Set PrepStmt = Connection.PrepareStatement( _
     "INSERT INTO TableName(ColumnName) VALUES ( ?
)")

'Crossfire
 PrepStmt = Connection.PrepareStatement( _
     "INSERT INTO TableName(ColumnName) VALUES( ? )")
```

3 その文にパラメータ値を割り当てます。

```
Dim NewValue As String
 NewValue = "Bob"
 PrepStmt.SetStringParameter 1, NewValue
```

4 文を実行します。

```
PrepStmt.ExecuteStatement
```

❖ ローを更新するには、次の手順に従います。

1　ULPreparedStatement オブジェクトを宣言します。

```
Dim PrepStmt As ULPreparedStatement
```

2　UPDATE 文を準備文オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

```
Set PrepStmt = Connection.PrepareStatement( _
     "UPDATE TableName SET ColumnName = ? WHERE ID =
?")
```

3　その文にパラメータ値を割り当てます。

```
Dim NewValue As String
 NewValue = "Bob"
 PrepStmt.SetParameter 1, NewValue
 PrepStmt.SetParameter 2, "6"
```

4　文を実行します。

```
PrepStmt.ExecuteStatement
```

❖ ローを削除するには、次の手順に従います。

1　ULPreparedStatement オブジェクトを宣言します。

```
'MobileVB
 Dim PrepStmt As ULPreparedStatement

'Crossfire
 Dim PrepStmt As UltraLiteAFLib.ULPreparedStatement
```

2　DELETE 文を準備文オブジェクトに割り当てます。

```
'MobileVB
 Set PrepStmt = Connection.PrepareStatement( _
     "DELETE FROM customer WHERE ID = ?")

'Crossfire
 PrepStmt = Connection.PrepareStatement( _
     "DELETE FROM customer WHERE ID = ?")
```

3 その文にパラメータ値を割り当てます。

```
Dim IDValue As String
 IDValue = "6"
 PrepStmt.SetParameter 1, IDValue
```

4 文を実行します。

```
PrepStmt.ExecuteStatement
```

データ検索：SELECT

SELECT 文を実行すると、ULPreparedStatement.ExecuteQuery メソッドは ULResultSet オブジェクトを返します。

ULResultSet クラスには、結果セット内をナビゲーションするためのメソッドが含まれています。次に、ULResultSet クラスのメソッドを使用して、その値にアクセスします。

ULResultSet オブジェクトの詳細については、「ULResultSet クラス」153 ページを参照してください。

例

次のコードでは、SELECT クエリの結果に ULResultSet を使用してアクセスします。最初に割り当てられたとき、ULResultSet は最初のローの前に配置されます。次に ULResultSet.MoveFirst メソッドが呼び出され、結果セットの最初のレコードをナビゲーションします。

結果セットのナビゲーションの詳細については、「動的 SQL を使用したナビゲーション」24 ページを参照してください。

```
'MobileVB
 Dim MyResultSet As ULResultSet
 Dim PrepStmt As ULPreparedStatement
 PrepStmt = Connection.PrepareStatement( _
     "SELECT ID, Name FROM customer")
 MyResultSet = PrepStmt.ExecuteQuery
 MyResultSet.MoveFirst
```

```
'Crossfire
 Dim MyResultSet As UltraLiteAFLib.ULResultSet
 Dim PrepStmt As UltraLiteAFLib.ULPreparedStatement
 PrepStmt = Connection.PrepareStatement( _
     "SELECT ID, Name FROM customer")
 MyResultSet = PrepStmt.ExecuteQuery
 MyResultSet.MoveFirst
```

Ultra Light for MobileVB では、Ultra Light データベースから結果セットへ特定の型のデータを取得するためのメソッドが提供されています。MobileVB は、Variant データ型の使用をサポートしません。このため、Ultra Light for MobileVB にはすべてのデータ型を処理する関数が備わっています。これらの各メソッドは、次のテンプレートを使用して呼び出されます。ここで、*Index* は SELECT 文でのカラム名の並び順です。

```
MyResultSetName.MethodName( Index )
```

例

次のコードは、GetString メソッドを使用して、現在のローのカラム値を取得する方法を説明します。

GetString メソッドは次の構文を使用します。ここで、*Index* は SELECT 文でのカラム名の並び順です。

```
MyResultSetName.GetString(Index)
```

データ修正の結果が反映されるように、MoveRelative(0) メソッドが呼び出され、結果セットの現在のバッファの内容をリフレッシュします。

```
If MyResultSet.RowCount = 0 Then
   lblID.Caption = ""
   txtName.Text = ""
 Else
   lblID.Caption = MyResultSet.GetString(1)
   txtName.Text = MyResultSet.GetString(2)
         MyResultSet.MoveRelative(0)
 End If
```

次の手順では、SELECT 文を使用して、データベースから情報を取り出します。クエリの結果は、ULResultSet オブジェクトに割り当てられます。

❖ SELECT 文を実行するには、次の手順に従います。

1 ULPreparedStatement オブジェクトを宣言します。

```
'MobileVB
 Dim PrepStmt As ULPreparedStatement

'Crossfire
 Dim PrepStmt As UltraLiteAFLib.ULPreparedStatement
```

2 準備文を ULPreparedStatement オブジェクトに割り当てます。次のコードでは、TableName と ColumnName がテーブルとカラムの名前です。

```
Set PrepStmt = Connection.PrepareStatement( _
     "SELECT ColumnName FROM TableName")
```

3 クエリを実行します。

下のコードでは、AFListBox が SELECT クエリの結果を取得します。

```
Dim MyResultSet As ULResultSet
 Set MyResultSet = PrepStmt.ExecuteQuery
 While MyResultSet.MoveNext
   aflistbox.AddItem MyResultSet.GetString(1)
 Wend
```

動的 SQL を使用したナビゲーション

Ultra Light for MobileVB は、幅広いナビゲーション作業を行うため、結果セットをナビゲーションする方法を多数提供します。

次の ULResultSet オブジェクトのメソッドを使うと、結果セット内をナビゲーションできます。

- **MoveAfterLast** 最後のローの後に移動します。
- **MoveBeforeFirst** 最初のローの前に移動します。
- **MoveFirst** 最初のローに移動します。
- **MoveLast** 最後のローに移動します。

- **MoveNext** 次のローに移動します。
- **MovePrevious** 前のローに移動します。
- **MoveRelative** いくつかのローを、現在のローを基準にして相対的に移動します。正のインデックス値は結果セット内を前に移動し、負のインデックス値は結果セット内を後ろに移動し、0 はカーソルを移動しません。ロー・バッファを再配置する場合は、0 が便利です。

例

次のコードは、MoveFirst メソッドを使用して、結果セット内をナビゲーションする方法を説明します。

```
'MobileVB
 Set PrepStmt = Connection.PrepareStatement( _
     "SELECT ID, Name FROM customer")
 Set MyResultSet = PrepStmt.ExecuteQuery
 MyResultSet.MoveFirst

'Crossfire
 PrepStmt = Connection.PrepareStatement( _
     "SELECT ID, Name FROM customer")
 MyResultSet = PrepStmt.ExecuteQuery
 MyResultSet.MoveFirst
```

すべての Move メソッドで同じ方法を使用できます。

これらのナビゲーション・メソッドの詳細については、「ULResultSet クラス」153 ページを参照してください。

ULResultSet schema プロパティ

ULResultSet.Schema プロパティを使うと、クエリのカラムに関する情報を取り出すことができます。ULResultSetSchema オブジェクトのプロパティは、ColumnName、ColumnCount、ColumnPrecision、ColumnScale、ColumnSize、ColumnSQLType です。

例

次の例は、ULResultSet.Schema を使用して、MobileVB グリッドのスキーマ情報を表示する方法を示しています。この例では、MyResultSet という名前の ULResultSet と grdSchema という名前の MobileVB グリッドが存在すると仮定しています。

```
Dim i As Integer
 For i = 1 To MyResultSet.Schema.ColumnCount
   grdSchema.AddItem (MyResultSet.Schema.ColumnName(i)
_
   & Chr(9) & MyResultSet.Schema.ColumnSQLType(i)), 0
 Next i
 grdSchema.AddItem _
   ("Column Name" & Chr(9) & "Column Type"), 0
```

テーブル API を使用したデータの使用

Ultra Light アプリケーションは、動的 SQL またはテーブル API を使用して、テーブル・データにアクセスできます。この項では、テーブル API を使用したデータ・アクセスについて説明します。

動的 SQL の詳細については、「動的 SQL を使用したデータの使用」19 ページを参照してください。

この項では、テーブル API を使用して次の操作を行う方法について説明します。

- テーブルのローのスクロール
- 現在のローの値へのアクセス
- find メソッドと lookup メソッドを使用したテーブルのローの検索
- ローの挿入、削除、更新

テーブル API を使用したナビゲーション

Ultra Light for MobileVB は、幅広いナビゲーション作業を行うため、テーブルをナビゲーションする方法を多数提供します。

次の ULTable オブジェクトのメソッドを使うと、結果セット内をナビゲーションできます。

- **MoveAfterLast** 最後のローの後に移動します。
- **MoveBeforeFirst** 最初のローの前に移動します。
- **MoveFirst** 最初のローに移動します。
- **MoveLast** 最後のローに移動します。
- **MoveNext** 次のローに移動します。
- **MovePrevious** 前のローに移動します。

- **MoveRelative** いくつかのローを、現在のローを基準にして相対的に移動します。正のインデックス値はテーブル内を前に移動し、負のインデックス値はテーブル内を後ろに移動し、0 はカーソルを移動しません。ロー・バッファを再配置する場合は、0 が便利です。

例

次のコードは、customer テーブルを開き、そのローをスクロールします。次に、各顧客の姓を示すメッセージ・ボックスを表示します。

```
'MobileVB
 Dim TCustomer as ULTable
 Set TCustomer = Conn.GetTable("customer")
 TCustomer.Open
 While TCustomer.MoveNext
    MsgBox TCustomer.Column( "lname" ).StringValue
 Wend

'Crossfire
 Dim TCustomer as UltraLiteAFLib.ULTable
 Set TCustomer = Conn.GetTable("Customer")
 TCustomer.Open
 While TCustomer.MoveNext
    MsgBox TCustomer.Column("LName").StringValue
 Wend
```

例

インデックスの指定

テーブル・オブジェクトを開くと、テーブルのローがアプリケーションに公開されます。デフォルトでは、ローはプライマリ・キー値の順に公開されますが、インデックスを指定すると特定の順序でローにアクセスできます。

例

次のコードは、ix_name インデックスで順序付けられた Customer テーブルの最初のローに移動します。

```
'MobileVB
 Set TCustomer = Conn.GetTable("customer")
 TCustomer.Open "ix_name"
 TCustomer.MoveFirst

'Crossfire
 TCustomer = Conn.GetTable("customer")
 TCustomer.Open "ix_name"
 TCustomer.MoveFirst
```

現在のローの値へのアクセス

ULTable オブジェクトは、次のいずれかの位置に常に置かれています。

- テーブルの最初のローの前
- テーブルのいずれかのローの上
- テーブルの最後のローの後ろ

ULTable オブジェクトがローの上に置かれている場合は、Column メソッドを適切なプロパティと一緒に使用して、現在のローのそのカラムの値を取得できます。

例

次のコードは、tCustomer ULTable オブジェクトから 3 つのカラムの値を取り出して、テキスト・ボックスに表示します。

```
Dim colID, colFirstName, colLastName As ULColumn
 Set colID = tCustomer.Column("ID")
 Set colFirstName = tCustomer.Column("fname")
 Set colLastName = tCustomer.Column("lname")
 txtID.Text = colID.IntegerValue
 txtFirstName.Text = colFirstName.StringValue
 txtLastName.Text = colLastName.StringValue
```

ULColumn のプロパティを使用して、値を設定することもできます。

```
colLastName.StringValue = "Kaminski"
```

これらのプロパティへの値の割り当てによって、データベース内のデータの値が変更されることはありません。

位置がテーブルの最初のローの前または最後のローの後ろにある場合でも、プロパティに値を割り当てることができます。しかし、カラムから値を取得することはできません。たとえば、次のコードはエラーを生成します。

```
' This code is incorrect
 TCustomer.MoveBeforeFirst
 id = TCustomer.Column( "ID" ).IntegerValue
```

バイナリ・データを操作するには、プロパティではなく GetByteChunk メソッドを使用します。

詳細については、『Ultra Light for MobileVB ユーザーズ・ガイド』>「GetByteChunk メソッド」を参照してください。

値のキャスト

選択する ULColumn プロパティは、割り当てる Visual Basic データ型に一致させてください。Ultra Light は自動的に互換性のないデータ型をキャストするため、StringValue メソッドを使用して整数値を文字列変数にフェッチしたりできます。

現在のローの値へのアクセスの詳細については、「ULColumn クラス」103 ページを参照してください。

find と lookup を使用したローの検索

Ultra Light には、データを操作するための操作モードがいくつかあります。これらのモードのうちの 2 つ (検索モードとルックアップ・モード) は、検索に使用されます。ULTable オブジェクトには、テーブル内の特定のローを検索するために、これらのモードに対応するメソッドがあります。

注意
find メソッドや lookup メソッドを使用して検索されるカラムは、テーブルを開くのに使用されたインデックスにあることが必要です。

- **find メソッド** ULTable オブジェクトを開いたときに指定したソート順に基づいて、指定された検索値と正確に一致する最初のローに移動します。

  find メソッドの詳細については、「FindBegin メソッド」187 ページを参照してください。

- **lookup メソッド** ULTable オブジェクトを開いたときに指定したソート順に基づいて、指定された検索値と一致するか、それより大きい値の最初のローに移動します。

  lookup メソッドの詳細については、「LookupBackward メソッド」191 ページを参照してください。

❖ ローを検索するには、次の手順に従います。

1　検索モードまたはルックアップ・モードを開始します。

FindBegin メソッドまたは LookupBegin メソッドを呼び出します。たとえば、次のコードは ULTable.FindBegin を呼び出します。

```
tCustomer.FindBegin
```

2　検索値を設定します。

検索値は、現在のローの値を設定することで設定します。これらの値を設定すると、バッファに影響しますが、データベースには影響しません。たとえば、次のコードは、バッファの姓のカラムを Kaminski に設定します。

```
tCustomer.Column("lname").StringValue = "Kaminski"
```

マルチカラム・インデックスの場合は、最初のカラムの値が必要ですが、ほかのカラムは省略できます。

3　ローを検索します。

適切なメソッドを使用して検索を実行します。たとえば、次の指示は、現在のインデックスで指定された値と正確に一致する最初のローを検索します。

```
tCustomer.FindFirst
```

ローの挿入、更新、削除

Ultra Light は、テーブルのローを一度に 1 つずつアプリケーションに公開します。ULTable オブジェクトにはカレント・ポジションがあります。カレント・ポジションは、テーブルのローの上、最初のローの前、または最後のローの後ろになります。

アプリケーションがロケーションを変更すると、Ultra Light はバッファにそのローのコピーを作成します。値を取得または設定する操作はすべて、このバッファにあるデータのコピーにのみ影響します。データベースのデータには影響しません。

例

次の文は、バッファの ID カラムの値を 3 に変更します。

```
colID.IntegerValue = 3
```

Ultra Light のモードの使用

Ultra Light モードによって、バッファ内の値を使用する目的が決まります。Ultra Light には、デフォルト・モードに加えて、次の 4 つの操作モードがあります。

- **挿入モード** ULTable.Insert メソッドを呼び出すと、バッファ内のデータが新しいローとしてテーブルに追加されます。
- **更新モード** ULTable.Update メソッドを呼び出すと、現在のローがバッファ内のデータに置き換えられます。
- **検索モード** ULTable.Find メソッドの 1 つが呼び出されたときに、値がバッファ内のデータに正確に一致するローの検索に使用されます。
- **ルックアップ・モード** いずれかの ULTable.Lookup メソッドが呼び出されたときに、バッファ内のデータと一致するか、それより大きい値のローを検索します。

❖ ローを更新するには、次の手順に従います。

1 更新するローに移動します。

テーブルをスクロールするか、find メソッドや lookup メソッドを使用して検索し、ローに移動できます。

2 更新モードを開始します。

たとえば、次の指示は、tCustomer テーブル上で更新モードを開始します。

```
tCustomer.UpdateBegin
```

3 更新するローの新しい値を設定します。

たとえば、次の指示は新しい値を Elizabeth に設定します。

```
ColFirstName.StringValue = "Elizabeth"
```

4 Update を実行します。

```
tCustomer.Update
```

更新操作が終了すると、直前に更新したローが現在のローになります。ULTable オブジェクトを開いたときに指定したインデックスのカラム値を変更した場合は、現在の位置は不確定です。

デフォルトでは、Ultra Light は AutoCommit モードで動作するため、更新は永続的な記憶領域のローに即時適用されます。AutoCommit モードを無効にした場合は、コミット操作を実行するまで、更新は適用されません。詳細については、「Ultra Light でのトランザクション処理」35 ページを参照してください。

警告
ローのプライマリ・キーを更新しないでください。代わりに、ローを削除して新しいローを追加してください。

ローの挿入

ローの挿入手順は、ローの更新手順とほぼ同じです。ただし、挿入操作の場合は、テーブル内の特定のローにあらかじめ指定する必要はありません。ローは、テーブルを開くときに使用したインデックスで自動的にソートされます。

❖ ローを挿入するには、次の手順に従います。

1　挿入モードを開始します。

たとえば、次の指示は、CustomerTable テーブル上で更新モードを開始します。

```
CustomerTable.InsertBegin
```

2　新しいローの値を設定します。

カラムの値を設定しない場合、そのカラムにデフォルト値があるときはデフォルト値が使用されます。カラムにデフォルト値がない場合は、NULL が使用されます。カラムが NULL を許可しない場合は、次のデフォルトが使用されます。

- 数値カラムの場合は 0
- 文字カラムの場合は空の文字列

明示的に値を NULL に設定するには、setNull メソッドを使用します。

```
CustomerTable.Column("FName").StringValue = fname
 CustomerTable.Column("LName").StringValue = lname
```

3 挿入を実行します。

挿入されたローは、Commit を実行したときに永続的にデータベースに保存されます。AutoCommit モードでは、Insert メソッドの一部として Commit が実行されます。

```
CustomerTable.Insert
```

ローの削除

挿入モードや更新モードに対応する削除モードはありません。

次のプロシージャは、ローを削除します。

❖ ローを削除するには、次の手順に従います。

1 削除するローに移動します。

2 削除を実行します。

```
tCustomer.Delete
```

BLOB データの処理

GetByteChunk メソッドを使用して、BINARY または LONG BINARY と宣言された、カラムの BLOB データをフェッチできます。

詳細については、「GetByteChunk メソッド」105 ページを参照してください。

例

次のコードは、ULColumn.GetByteChunk メソッドを使用して、BLOB データを取得する方法を説明します。

```
'MobileVB
 Dim table as ULTable
 Dim col As ULColumn
 Dim data(1 to 1024) As Byte
 Dim data_fit As Boolean
```

```
 Dim size As Long
 Set table = Conn.GetTable("image")
 table.Open
 size = 1024
 Set col = table.Column("img_data")
 data_fit = col.GetByteChunk(VarPtr(data(1)), size)
 If data_fit Then
   'No truncation
 Else
   'data truncated at 1024
 End if
 table.Close

'Crossfire
 Dim table as ULTable
 Dim col As ULColumn
 Dim data(1 to 1024) As Byte
 Dim data_fit As Boolean
 Dim size As Long
 Set table = Conn.GetTable("image")
 table.Open
 size = 1024
 Set col = table.Column("img_data")
 ' The data argument must be a local variable
 data_fit = col.GetByteChunk(data, size)
 If data_fit Then
   'No truncation
 Else
   'data truncated at 1024
 End if
 table.Close
```

Ultra Light でのトランザクション処理

Ultra Light のトランザクション処理は、データベース内のデータの整合性を保証します。トランザクションは、作業の論理単位です。トランザクション全体が実行されるか、トランザクション内の文がどれも実行されないかのいずれかです。

デフォルトでは、Ultra Light は AutoCommit モードで動作します。AutoCommit モードでは、挿入、更新、削除はそれぞれ独立したトランザクションとして実行されます。操作が完了すると、データベースに変更が加えられます。

ULConnection.AutoCommit プロパティを false に設定すると、複数文のトランザクションを使用できます。たとえば、2 つの口座間で資金を移動するアプリケーションでは、振り込み元の口座からの引き落としと振り込み先口座への振り込みが、1 つのトランザクションを構成します。AutoCommit が false に設定されている場合は、ULConnection.Commit 文を実行してトランザクションを完了し、データベースへの変更を永続的なものにするか、ULConnection.Rollback 文を実行してトランザクションのすべての処理をキャンセルしてもかまいません。AutoCommit をオフにすると、パフォーマンスが向上します。

注意
Autocommit を False に設定していても、同期はコミットを実行します。

スキーマ情報へのアクセス

ULConnection、ULTable、ULColumn オブジェクトにはそれぞれ、スキーマ・プロパティが含まれます。これらのスキーマ・オブジェクトは、データベース内のテーブル、カラム、インデックス、パブリケーションに関する情報を提供します。

注意
API によるスキーマの変更はできません。スキーマに関する情報の取得のみが可能です。

スキーマの修正については、「データベースのスキーマの変更」13 ページを参照してください。

- **ULDatabaseSchema**　データベース内のテーブルの数と名前、日付と時刻のフォーマットなどのグローバル・プロパティ。

  ULDatabaseSchema オブジェクトを取得するには、ULConnection.Schema プロパティにアクセスします。

- **ULTableSchema**　テーブル内のカラムの数と名前、テーブルの Indexes コレクション。

  ULTableSchema オブジェクトを取得するには、ULTable.Schema プロパティにアクセスします。

- **ULColumnSchema**　デフォルト値、名前、オートインクリメントかどうかなど、個々のカラムに関する情報。

  ULColumnSchema オブジェクトを取得するには、ULColumn.Schema プロパティにアクセスします。

- **ULIndexSchema**　インデックス内のカラムに関する情報。インデックスには直接対応するデータがないので、個別の ULIndex オブジェクトはなく、ULIndexSchema オブジェクトだけが存在します。

  ULIndexSchema オブジェクトは、ULTableSchema.GetIndex メソッドを使用してアクセスできます。

- **ULPublicationSchema** パブリケーションに含まれるテーブルとカラムの数と名前。パブリケーションもスキーマのみで構成されているため、ULPublication オブジェクトではなく、ULPublicationSchema オブジェクトが存在します。

  ULPublicationSchema オブジェクトは、ULDatabaseSchema.GetPublicationSchema メソッドを使用してアクセスできます。

- **ULResultSetSchema** 結果セットのカラムの数と名前。

  ULResultSetSchema オブジェクトは、ULPreparedStatement.ResultSetSchema プロパティを使用してアクセスできます。

エラー処理

通常の操作では、Ultra Light for MobileVB はエラーをスローできます。エラーは SQLCODE 値として表現され、負の数字は特定の種類のエラーを示します。

Ultra Light for MobileVB によってスローされるエラー・コードのリストについては、「ULSQLCode 列挙」163 ページを参照してください。

UltraLite for MobileVB は、ULDatabaseManager オブジェクトと ULConnection オブジェクトからしか、エラーをスローしません。ULDatabaseManager の次のメソッドは、エラーをスローできます。

- CreateDatabase
- CreateDatabaseWithParms
- DropDatabase
- DropDatabaseWithParms
- OpenConnection
- OpenConnectionWithParms

Ultra Light for MobileVB 内での他のエラーや例外はすべて、ULConnection オブジェクトを経由します。

ULDatabaseManager オブジェクトと ULConnection オブジェクトからのエラー番号へのアクセスの詳細については、「ULConnection クラス」112 ページと 「ULDatabaseManager クラス」131 ページを参照してください。

MobileVB または Crossfire の標準エラー処理機能を使用して、エラーを処理できます。Ultra Light オブジェクトがエラーの原因である場合、Err オブジェクトには ULSQLCode 番号が割り当てられます。ULSQLCodes エラーは、特定の種類のエラーを示す負の数字です。ULSQLCode 列挙は、これらの値に関連付けられている一連の説明的な定数を提供します。

詳細については、「ULSQLCode 列挙」163 ページを参照してください。

MobileVB 環境で型の補完を使用するには、次のようなエラー処理関数を作成します。

```
'MobileVB
 Public Function GetError() As ULSQLCode
     GetError = Err.Number
 End Function
```

その後、GetError 関数を使用して Ultra Light エラーに簡単にアクセスできます。

ユーザの認証

新しいユーザは既存の接続から追加します。Ultra Light のすべてのデータベースは、デフォルトのユーザ ID DBA とパスワード SQL を使用して作成されるため、最初はこの初期ユーザとして接続します。

ユーザ ID の変更はできません。ユーザを 1 人追加して既存のユーザを削除します。Ultra Light ではデータベースごとにユーザ ID が 4 つまで許可されます。

接続権限の付与または取り消しの詳細については、「GrantConnectTo メソッド」116 ページと「RevokeConnectFrom メソッド」121 ページを参照してください。

❖ ユーザを追加する、または既存のユーザのパスワードを変更するには、次の手順に従います。

1　DBA 権限のあるユーザとしてデータベースに接続します。

2　希望するパスワードでユーザに接続権限を付与します。

```
conn.GrantConnectTo("Robert", "newPassword")
```

❖ 既存のユーザを削除するには、次の手順に従います。

1　DBA 権限のあるユーザとしてデータベースに接続します。

2　次のように、ユーザの接続権限を取り消します。

```
conn.RevokeConnectFrom("Robert")
```

データの同期

Ultra Light アプリケーションは、統合データベースと同期できます。同期には、Mobile Link 同期サーバと適切なライセンスが必要です。

この項では、同期の概要について簡単に説明し、Ultra Light for MobileVB のユーザにとって特に関心があるいくつかの機能について説明します。同期の詳細な説明については、『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。

また、CustDB サンプル・アプリケーションには、同期の実例もあります。このサンプルは、『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」で説明されています。

Ultra Light for MobileVB は、TCP/IP、HTTP、HTTPS 通信による同期をサポートします。同期は、Ultra Light アプリケーションによって開始されます。いずれの場合でも、ULConnection オブジェクトのメソッドとプロパティを使用して同期を制御します。

注意
暗号化された同期 (HTTPS) を使用して同期する、または TCP/IP で暗号化を使用するには、別途ライセンスを取得できるセキュリティ・オプションが必要です。このオプションのご注文については、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。

詳細については、『SQL Anywhere Studio の紹介』>「SQL Anywhere Studio へようこそ」を参照してください。

❖ TCP/IP または HTTP で同期するには、次の手順に従います。

1 同期情報を準備します。

ULConnection.SyncParms オブジェクトの必須プロパティに値を割り当てます。

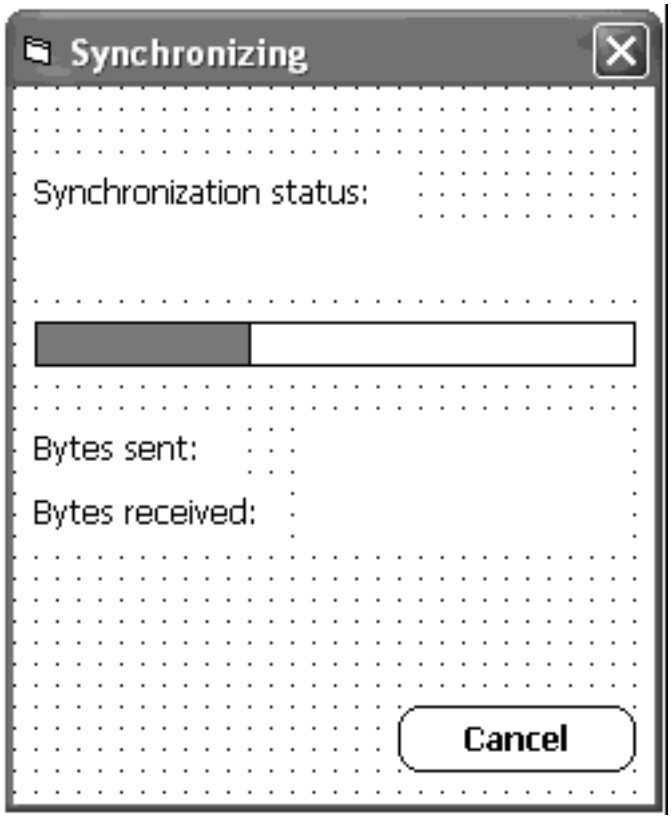
設定するプロパティと値の詳細については、『Mobile Link クライアント』>「Ultra Light クライアント」を参照してください。

2　同期を実行します。

ULConnection.Synchronize メソッドを呼び出します。

同期テンプレートの追加

Ultra Light for MobileVB には、同期セッションのステータスをモニタするのに使用できる、テンプレート・フォームがあります。このフォームは、Palm OS 用と Pocket PC 用の両方に用意されています。アプリケーションにあるこれらのテンプレートを使用したり、カスタマイズしたり、Ultra Light の同期イベントが動作する様子を学ぶのに、テンプレートを調べたりできます。



このテンプレートをアプリケーションに追加する方法は、MobileVB と Crossfire のどちらを使用しているかによって決まります。

❖ アプリケーションに同期テンプレートを追加するには、次の手順に従います (MobileVB)。

1 [プロジェクト] メニューから [Add Form] を選択します。

2 [Ultra Light for MobileVB Sync Form (Windows CE)] または [Ultra Light for MobileVB Sync Form (Palm)] を選択します。

3 [開く] をクリックします。フォームのコピーがアプリケーションに追加されます。

❖ アプリケーション (Crossfire) に同期テンプレートを追加するには、次の手順に従います。

1　[プロジェクト] メニューで [新しい項目の追加] を選択します。

2　[ローカル プロジェクト アイテム] ー [Ultralite_Crossfire Forms] で、アプリケーションに応じて [UltraLite Crossfire 9 Sync Form for CE]、[UltraLite Crossfire 9 Sync Form for Palm]、または [UltraLite Crossfire 9 Sync Form for PalmHires] を選択します。

3　[開く] をクリックします。フォームのコピーがアプリケーションに追加されます。

同期フォームを使用するコードの記述

InitSyncForm 関数を呼び出し、ULConnection オブジェクトに渡します。これは、各同期の前に行います。

例

次のコードは、Form_Sync という名前の同期ステータス・フォーム、Connection という名前の ULConnection オブジェクトを使用します。

```
Form_Sync.InitSyncForm Connection
 Connection.Synchronize
```

アプリケーションが同期するたびに、同期ステータス・フォームが表示されます。同期が進行する様子を、エンド・ユーザは進行状況を示すバーとバイト数で確認できます。同期が完了したら、フォームが閉じます。[キャンセル] ボタンをクリックすると、Ultra Light は現在の同期をキャンセルします。

詳細については、『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」を参照してください。

Ultra Light アプリケーションの配備

アプリケーションを完了したら、またはアプリケーションをテストしたい場合、アプリケーションをデバイスに配備する必要があります。この項では、デバイスに Ultra Light アプリケーションを配備するための手順について説明します。

Windows CE への Ultra Light for MobileVB アプリケーションの配備

Ultra Light アプリケーションを Windows CE に配備するには、次の手順を実行する必要があります。

- アプリケーションと Ultra Light コンポーネントを配備します。

  1. アプリケーションを設定します。

     - [MobileVB] メニューから、[MobileVB 設定] を選択します。ダイアログが表示されます。
     - 左ウィンドウ枠で、[依存性] を選択し、[ユーザ依存] タブをクリックします。
     - [追加] ボタンをクリックし、*c:¥tutorial¥mvb¥tutCustomer.usm* を選択します。これは、このファイルが配備に組み込まれるように MobileVB に指定します。
     - 左ウィンドウ枠で [Pocket PC 設定] 項目を選択します。
     - [デバイスのインストール・パス] に *¥Tutorial¥mvb* と入力します。
     - [OK] をクリックしてダイアログを閉じます。

  2. [MobileVB] - [Deploy to Device] を選択し、[PocketPC] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。

  3. MobileVB 3.0 以降を実行している場合、Ultra Light コンポーネントはアプリケーションとともに自動的に配備されます。

3.0 より前のバージョンの MobileVB を実行している場合は、Ultra Light コンポーネントをデバイスにコピーする必要があります。ファイル *ulmvb9.dll* をデバイスの *¥Program Files¥AppForge* にコピーします。このファイルは、SQL Anywhere 9.0 インストール環境のプラットフォーム固有のサブディレクトリである、*¥UltraLite¥UltraLiteForMobileVB¥ce¥arm* または *¥UltraLite¥UltraLiteForMobileVB¥ce¥mips* にあります。この手順は、デバイスあたり 1 回だけ実行する必要があります。

- Ultra Light データベースまたはスキーマの初期コピーを配備します。

  多くの場合、Ultra Light スキーマ・ファイルのみを配備すれば十分です。データベース・ファイルは、初回の接続試行時にスキーマから作成されます。このため、同期を使用してデータの初期コピーをロードできます。Palm Desktop Organizer Software に付属の Install Tool を使用して、標準的な方法で *.prc* ファイルを配備できます。

  データベースまたはスキーマ・ファイルは、アプリケーションが特定できるロケーションに配備する必要があります。このロケーションは、Database On CE および Schema On CE 接続パラメータによって定義されます。

  『Ultra Light データベース・ユーザーズ・ガイド』> 「DatabaseOnCE 接続パラメータ」と『Ultra Light データベース・ユーザーズ・ガイド』> 「SchemaOnCE 接続パラメータ」を参照してください。

- ActiveSync 用の Mobile Link プロバイダを配備します。

  この手順が必要なのは、ActiveSync を使用してアプリケーションを同期する場合のみです。

  手順については、『Mobile Link クライアント』> 「ActiveSync 用 Mobile Link プロバイダのインストール」を参照してください。

Palm OS への Ultra Light for MobileVB の配備

Ultra Light アプリケーションを Palm OS デバイスに配備するには、次の手順を実行する必要があります。

- アプリケーションと Ultra Light コンポーネントを配備します。

  1. アプリケーションを設定します。

     - [MobileVB] メニューから、[MobileVB 設定] を選択します。ダイアログが表示されます。
     - 左ウィンドウ枠で、[依存性] を選択し、[ユーザ依存] タブをクリックします。
     - [追加] ボタンをクリックし、*c:¥tutorial¥mvb¥tutCustomer.pdb* を選択します。これは、このファイルが配備に組み込まれるように MobileVB に指定します。
     - 左ウィンドウ枠で [Palm OS 設定] 項目を選択し、アプリケーションの作成者 ID を入力します。有効な HotSync 名を選択します。[OK] をクリックしてダイアログを閉じます。

  2. [MobileVB] - [Deploy to Device] を選択し、[Palm OS] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。

  3. MobileVB 3.0 以降を実行している場合、Ultra Light コンポーネントはアプリケーションとともに自動的に配備されます。

     3.0 より前のバージョンの MobileVB を実行している場合は、Ultra Light コンポーネントをデバイスにコピーする必要があります。ファイル *ulmvb9.prc* をデバイスの *¥Program Files¥AppForge* にコピーします。ファイルは、SQL Anywhere インストール環境の *¥UltraLite¥UltraLiteForMobileVB¥palm¥68k* サブディレクトリにあります。この手順は、デバイスあたり 1 回だけ実行する必要があります。

- Ultra Light データベースまたはスキーマの初期コピーを配備します。

多くの場合、Ultra Light スキーマ・ファイルのみを配備すれば十分です。データベース・ファイルは、初回の接続試行時にスキーマから作成されます。このため、同期を使用してデータの初期コピーをロードできます。

Palm OS へ配備する *.prc* ファイルは、スキーマ・ペインタ、ulxml、および ulinit を含む任意の Ultra Light ユーティリティを使用して作成できます。

スキーマまたはデータベースは、アプリケーションがロケーションを特定できるように正しい作成者 ID を使用して指定する必要があります。Database On Palm および Schema On Palm 接続パラメータは、この作成者 ID を使用してスキーマまたはデータベースを検索します。

『Ultra Light データベース・ユーザーズ・ガイド』> 「DatabaseOnPalm 接続パラメータ」と『Ultra Light データベース・ユーザーズ・ガイド』> 「SchemaOnPalm 接続パラメータ」を参照してください。仮想ファイル・システムの使用については、『Ultra Light データベース・ユーザーズ・ガイド』> 「VFSOnPalm パラメータ」を参照してください。

- HotSync 用 Mobile Link 同期コンジットを配備します。

  この手順が必要なのは、HotSync を使用してアプリケーションを同期する場合のみです。

  手順については、『Mobile Link クライアント』> 「Mobile Link HotSync コンジットの配備」を参照してください。

Ultra Light Palm アプリケーションのステータスの管理

Palm OS デバイスは、一度に 1 つのアプリケーション上でのみ動作します。ただし、一般的には、ユーザが別のアプリケーションに切り替えてから元のアプリケーションに戻ってきた場合、このユーザが他のアプリケーションで作業を行っていた間中、元のアプリケーションがそのまま保留されていたかのように表示します。このような錯覚を実現するには、ユーザが別のアプリケーションに切り替えたときにアプリケーションは内部ステータスを保存する必要があります。アプリケーションは、再起動されたときに内部ステータスを復帰する必要があります。

アプリケーションは結果セットを再度開いてこれらの結果セット内のアプリケーションを再配置する必要があるため、データベース・アプリケーションのステータスを保存して復帰する作業は容易ではありません。Ultra Light には、アプリケーションのステータスを保存して復帰する機能が用意されています。

結果セット上のカーソルのステータスは、自動的に管理されます。テーブル・ベースの API を使用する Mobile VB アプリケーションは、ULTable オブジェクトの Open メソッドの永続的な名前のパラメータに値を指定します。

ステータスの管理方法の知識

Ultra Light は、ステータス情報が保存してあるテーブルの名前とテーブルをそのステータスにリストアするのに十分な情報を保管しています。テーブルと対応する名前は、テーブルの名前であってもなくてもかまいません。これは、「**永続的な名前**」と呼ばれます。

Ultra Light アプリケーションは、1 つのテーブル内の複数のインスタンスを同時に開くことができます。この場合、テーブル名はユニークではありません。たとえば、次のコード (MobileVB を使用) は、同じテーブルを 2 回開きます。

```
' MobileVB
 Set table1 = Connection.GetTable( "ULCustomer" )
 table1.Open "", "customer1"
 ' operations here
 Set table2 = Connection.GetTable( "ULCustomer" )
 table2.Open "", "customer2"
```

テーブルを開く場合、アプリケーションは必要に応じて永続的な名前をパラメータとして指定できます。永続的な名前のステータスが保存されている場合、そのテーブルを開いて適切なローに配置します。対応するテーブルがなければ、そのテーブルを開いて最初のローの前に配置します。

アプリケーションが終了する場合、アプリケーションは、開いているテーブルと接続を明示的に閉じることも閉じないこともあります。開いているテーブルを閉じない場合、Ultra Light は、永続的な名前が提供されていて開いている各テーブルの現在のローを記録します。永続的な名前を持たないテーブルは閉じます。

Connection オブジェクトが ULConnection 型で、ULCustomer という名前のテーブルがデータベースに存在するとします。

```
'MobileVB
 Dim table As ULTable
 Set table = Connection.GetTable( "ULCustomer" )
 table.Open "", "customer"
```

コードの 2 番目の行は、ULCustomer テーブルを表すテーブル・オブジェクトを取得します。テーブルは、まだ読み書きするために開かれていません。

Open 呼び出し (コードの 3 行目) では、最初のパラメータは空の文字列で、データがプライマリ・キーによって順序付けられることを示します。2 番目のパラメータは、テーブルに割り当てられている永続的な名前です。このテーブルがまだ開いている間にアプリケーションが終了した場合、ステータス PDB は *customer* を ULCustomer テーブルと対応させ、現在の位置を保存します。

永続的な名前に関する注意

- 永続的な名前が空である場合、Ultra Light は、このテーブルのステータス情報を保存しません。また、このテーブルを開けるときにステータス情報を検索しません。

テーブルのステータスを保存する必要がない場合、永続的な名前は空にしておきます。そうすると、ステータス・データベースでステータスが検索されません。

- HotSync 同期は、開いているテーブルのステータスに影響しません。ユーザがデバイスの HotSync ボタンを押すと、オペレーティング・システムは、他のアプリケーションが起動するときと同様に、アプリケーションを閉じます。その結果、開いているテーブルのステータスはステータス PDB に記録され、ユーザがアプリケーションに戻ってテーブルが再び開いている場合、ユーザは予期したローに配置されます。そのローが同期の一部として削除された場合、ユーザは次のローに (または、そのローが最後のローならば、最後のローの後に) 配置されます。

- オートコミットをオフにしたアプリケーションは、コミットされていないトランザクションで終了することがあります。Ultra Light は、アプリケーションが再起動したときに失われないように、これらのトランザクションを管理します。

- Ultra Light は、指定した永続的な名前に一致するテーブルをステータス PDB で見つけると、そのテーブルとインデックスが、位置情報が記録されたときに使用されたテーブルとインデックスと同じであるか確認します。同じでない場合、Open の呼び出しは失敗します。

- 永続的な名前の使用は、Palm OS に固有です。Ultra Light for MobileVB アプリケーションを Windows CE 用に作成する場合、永続的な名前は使用しません。Windows CE 上のアプリケーションは、デスクトップ・マシンと同様に動作するからです。

例：永続的な名前を使用したステータス情報の管理

PersistentName のプログラム例は、比較的簡単なプログラムで、管理されたステータス情報の使用方法を示します。プログラム例は、http://www.sybase.com/detail?id=1022734 から入手できます。その例の重要な部分を次に示します。

```
'MobileVB
 CustomerTable.Open
 AddRow "John", "Doe", "Atlanta"
 AddRow "Mary", "Smith", "Toronto"
 AddRow "Jane", "Anderson", "New York"
 AddRow "Margaret", "Billington", "Vancouver"
 AddRow "Fred", "Jones", "London"
 AddRow "Jack", "Frost", "Dublin"
 AddRow "David", "Reiser", "Berlin"
 AddRow "Kathy", "Stevens", "Waterloo"
 AddRow "Rebecca", "Gable", "Paris"
 AddRow "George", "Jenkins", "Madrid"
 CustomerTable.Close
```

このコードは、ULCustomer テーブルに 10 のローを追加します。テーブルの Open を呼び出しますが、この場合、テーブルの位置の管理は考慮しないので、永続的な名前は提供されていません。このコードはデータを挿入するだけなので、テーブルの位置は関係ありません。

次の行は、ULCustomer テーブルを開き、ローをプライマリ・キーによって順序付け、customer の永続的な名前を割り当てます。

```
CustomerTable.Open "" , "customer"
```

アプリケーションを以前に実行した場合、customer のステータス・データベースで検索が行われます。そうでない場合、customer はこのテーブルと対応します。

customer テーブルは、アプリケーションが動作している間、開いたままです。ユーザが別のアプリケーションに切り替えた場合、Ultra Light によって、ユーザが離れたテーブルの位置が記録されます。アプリケーションが再び起動したときに、テーブルが開かれ、永続的な名前 customer を持つテーブルの位置情報が既知であることが Ultra Light によって判断されるため、ユーザはそのローに位置設定されます。

ユーザが [終了] ボタンをクリックすると、customer テーブルと接続は、アプリケーションが消える前に閉じられます。これは、customer テーブルのステータス情報を捨てるという影響があり、アプリケーションが再起動したとき、ユーザは最初のローに配置されます。

第 3 章

チュートリアル：Ultra Light for MobileVB アプリケーションのサンプル

この章の内容

この章はチュートリアル形式で、PocketPC または Palm OS デバイス用の Ultra Light for MobileVB アプリケーションを構築するプロセスを解説します。

概要

このチュートリアルでは、テーブル API を使用して Ultra Light for MobileVB アプリケーションを構築するプロセスを解説します。チュートリアルを終了すると、アプリケーションと小規模なデータベースが Windows CE デバイス上に構築されます。これを、中央のデータベースと同期します。

テーブル API の詳細については、『Ultra Light for MobileVB ユーザーズ・ガイド』> 「Ultra Light for MobileVB API リファレンス」を参照してください。

所要時間

このチュートリアルは、コードをコピーして貼り付ける場合、約 30 分で終了します。自分でコードを入力する場合、これより長い時間が必要です。

能力と経験

このチュートリアルは、次のことを前提にしています。

- MobileVB と Microsoft Visual Basic 6 がコンピュータにインストールされている
- MobileVB アプリケーションの作成、テスト、トラブルシューティングができる
- Ultra Light スキーマ・ペインタを使用して、Ultra Light スキーマを作成する方法を知っている

  詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「Ultra Light スキーマ・ペインタ」を参照してください。
- AppForge Booster がインストールされている

  Booster がない場合は、http://www.appforge.com/booster.html から入手できます。

注意

SQL Anywhere Studio がなくても、このチュートリアルの大部分を実行できます。このチュートリアルの同期の項では、SQL Anywhere Studio が必要です。

目的

このチュートリアルの目的は、Ultra Light アプリケーションの開発プロセスについて、知識と経験を得ることです。

レッスン 1：プロジェクト・アーキテクチャの作成

最初の手順では、Ultra Light データベース・スキーマを作成する方法を説明します。データベース・スキーマは、データベースに関する記述です。データベース・スキーマは、データベース内のテーブル、インデックス、キー、パブリケーションとそれらの間のすべての関係を記述します。

データベース・スキーマの詳細については、「Ultra Light データベース・スキーマ・ファイルの作成」12 ページを参照してください。

❖ Ultra Light データベース・スキーマを作成するには、次の手順に従います。

1 このチュートリアル用のディレクトリを作成します。

このチュートリアルでは、保存先ディレクトリを *c:¥Tutorial¥mvb* とします。別の名前のディレクトリを作成した場合は、チュートリアルを通じて、*c:¥Tutorial¥mvb* の代わりにそのディレクトリを使用してください。

2 Ultra Light スキーマ・ペインタを使用して、データベース・スキーマを作成します。

データベース・スキーマの作成の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「レッスン 1：Ultra Light データベース・スキーマの作成」を参照してください。

- **スキーマのファイル名** tutcustomer.usm
- **テーブル名** customer
- **customer のカラム**

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

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
lname	char (20)	いいえ	なし
city	char (20)	はい	なし
phone	char (12)	はい	555-1234

- **プライマリ・キー** 昇順 id

3 Palm デバイスを使用している場合、作成者 id が *Syb3* のスキーマを Palm へエクスポートします。

a. [ファイル] - [Palm 用にスキーマをエクスポート] を選択します。

b. [作成者 ID] に Syb3 と入力します。

c. チュートリアルのディレクトリに、そのファイルを *tutcustomer.pdb* として保存します。

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 を使用してください。

MobileVB プロジェクトの作成

次の手順では、アプリケーションの MobileVB プロジェクトを作成し、Ultra Light for MobileVB コントロールへの参照を追加します。

Visual Basic 環境の左側にある Visual Basic ツールバーに、標準の Visual Basic ツールに加えて MobileVB ツールが表示されます。

❖ Ultra Light 接続パラメータ・コントロールを追加するには、次の手順に従います。

1 MobileVB を起動します。

- [スタート] ― [プログラム] ― [AppForge MobileVB] ― [Start MobileVB] を選択します。

MobileVB プロジェクト・マネージャが表示されます。

2 eMbedded Visual Basic メニューから、[プロジェクト] ― [コンポーネント] を選択します。

3 [コントロール] タブをクリックします。

4 リストを下にスクロールして [UltraLite Connection Parameters 9.0] を選択します。[OK] をクリックします。

使用可能なコントロールのリストにこの項目が表示されない場合は、次の手順を実行します。

- MobileVB を閉じ、プロジェクトを保存します。
- *ultralite¥UltraLiteforMobileVB¥win32* でコマンド・プロンプトを開き、次のコマンドを実行します。

```
ulmvbreg -r
```

- MobileVB を再起動し、プロジェクトを開きます。
- [プロジェクト] ― [コンポーネント] を選択します。
- [UltraLite Connection Parameters 9.0] を選択します。

データベース・アイコンがツールバーに追加されます。このアイコンをダブルクリックし、ULConnectionParms オブジェクトをフォームに追加します。

❖ Ultra Light for MobileVB コントロールへの参照を追加するには、次の手順に従います。

1 MobileVB を起動します。

- [スタート] ー [プログラム] ー [AppForge MobileVB] ー [Start MobileVB] を選択します。

MobileVB プロジェクト・マネージャが表示されます。

2 新規プロジェクトを作成します。

[新規プロジェクト] をクリックします。設計ターゲットを求められたら、適切なターゲットを選択します。このチュートリアルでは、Palm OS および PocketPC デバイスについて説明します。

3 Ultra Light for MobileVB に参照を追加します。

- [プロジェクト] ー [参照] を選択します。
- [iAnywhere Solutions, UltraLite for MobileVB 9.0] を選択し、[OK] をクリックします。

  コントロールが使用可能な参照のリストに表示されない場合は、MobileVB を終了し、次のコマンドを実行します。

```
ulmvbreg -r
```

4 プロジェクトに名前を付けます。

- [プロジェクト] ー [MobileVBProject1 プロパティ] をクリックします。
- [プロジェクト名] に **UltraLiteTutorial** と入力します。これは、アプリケーションがデバイスに表示されるときの名前になります。
- [OK] をクリックします。

5 プロジェクトを保存します。

- [ファイル] ー [プロジェクトの保存] を選択します。
- フォームを *c:¥tutorial¥mvb¥Tutorial.frm* として保存します。

- プロジェクトを *c:¥tutorial¥mvb¥Tutorial.vbp* として保存します。

レッスン 2：フォームの作成

「レッスン 1：プロジェクト・アーキテクチャの作成」58 ページの手順を完了すると、プロジェクトにフォームが 1 つ表示されます。次の手順は、フォームを使用してユーザ・インタフェースを作成します。この例では、実際のアプリケーションと同様に、ラベルを出力として、テキスト・ボックスとボタンを入力として使用します。

❖ プロジェクトにコントロールを追加するには、次の手順に従います。

1　次の表のコントロールとプロパティをフォームに追加します。

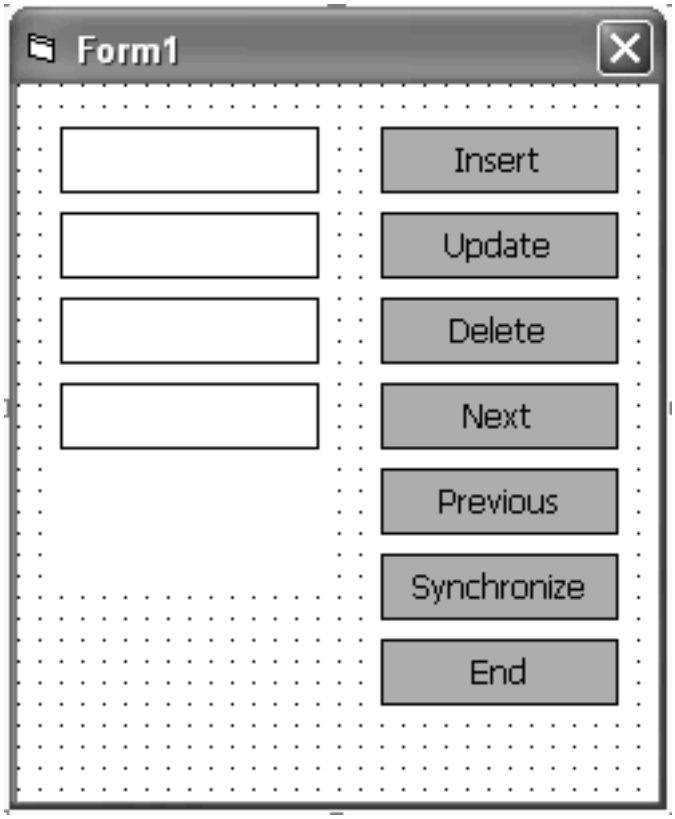
タイプ	名前	キャプションまたはテキスト
AFTextBox	txtfname	
AFTextBox	txtlname	
AFTextBox	txtcity	
AFTextBox	txtphone	
AFLabel	lblID	
AFButton	btnInsert	挿入
AFButton	btnUpdate	更新
AFButton	btnDelete	削除
AFButton	btnNext	次へ

タイプ	名前	キャプションまたはテキスト
AFButton	btnPrevious	前へ
AFButton	btnSync	同期
AFButton	btnDone	終了

2　アプリケーションを確認します。

- [MobileVB] ー [コンパイルと検証] を選択します。

フォームは、次の図のようになります。


Form1
Insert
Update
Delete
Next
Previous
Synchronize
End

レッスン 3：サンプル・コードの作成

この章では、データベースに接続し、データベース内をナビゲーションし、データベース内のデータを操作するための、Visual Basic コードを作成するプロセスを解説します。

このレッスンには、アプリケーションを Adaptive Server Anywhere データベースと同期するための指示も含まれています。レッスンのこの部分はオプションで、SQL Anywhere Studio を必要とします。

データベースとの接続のためのコードの作成

このアプリケーションでは、Form_Load イベントの間にデータベースに接続します。一般のモジュールを使用しても、データベースに接続できます。

この例は、データベースに接続するのに ULConnectionParms オブジェクトを使用します。接続文字列も使用できます。

参照情報については、「ULConnectionParms クラス」126 ページを参照してください。

❖ Ultra Light データベースに接続するためのコードを作成するには、次の手順に従います。

1. フォームをダブルクリックして、[コード] ウィンドウを開きます。
2. 必要な Ultra Light オブジェクトを宣言します。

   フォームの [declarations] 領域に、次のコードを入力します。

```
Public DatabaseMgr As New ULDatabaseManager
 Public Connection As ULConnection
 Public CustomerTable As ULTable
```

3. 接続パラメータを指定します。

- ULConnectionParms1 という名前のフォームに ULConnectionParms オブジェクトを追加します。ULConnectionParms コントロールは、ツールボックスにあるデータベース・アイコンです。
- [プロパティ] ウィンドウで、ULConnectionParms のプロパティを指定します。

  Windows CE デバイスに配備する場合、次のプロパティを指定します。

プロパティ	値
DatabaseOnDesktop	c:\tutorial\mvb\tutCustomer.udb
DatabaseOnCE	\tutorial\mvb\tutCustomer.udb
SchmeOnDesktop	c:\tutorial\mvb\tutCustomer.usm
SchmaOnCE	\tutorial\mvb\tutCustomer.usm

  Palm デバイスに配備する場合、次のプロパティを指定します。

プロパティ	値
DatabaseOnDesktop	c:\tutorial\mvb\tutCustomer.pdb
DatabaseOnPalm	Syb3
SchmeOnDesktop	c:\tutorial\mvb\tutCustomer.usm
SchemaOnPalm	tutCustomer

4　データベースに接続するためにのコードを Form_Load イベントに追加します。

  データベース・マネージャは、ULConnectionParms1 オブジェクトで指定されるデータベースへの接続を開こうとします。データベースが存在しない場合は、指定されたスキーマを使用して新しいデータベースを作成します。

```
Sub Form_Load()
   ' enable error handling
   On Error Resume Next
   ' try to connect to an existing database
   Set Connection = _
     DatabaseMgr.OpenConnectionWithParms( _
       ULConnectionParms1)
   ' if the database does not exist, create one
   If Err.Number = ULSQLCode.ulSQLE_NOERROR Then
     MsgBox "Connected to an existing database"
   ElseIf Err.Number = _
     ULSQLCode.ulSQLE_ULTRALITE_DATABASE_NOT_FOUND _
   Then
     Err.Clear
     Set Connection = _
     DatabaseMgr.CreateDatabaseWithParms( _
      ULConnectionParms1)
     If Err.Number = ULSQLCode.ulSQLE_NOERROR _
      Then
        MsgBox "Connected to a new database"
      Else
        MsgBox Err.Description
      End If
    Else
     MsgBox Err.Description
    End If
End Sub
```

5 [終了] ボタンをクリックされたときに、アプリケーションを終了して接続を閉じるコードを追加します。

```
Sub btnDone_Click()
     Connection.Close
     End
End Sub
```

6 アプリケーションを実行します。

- [実行] ー [実行] を選択します。
- 最初のメッセージ・ボックスの後に、フォームがロードされます。
- [終了] をクリックしてアプリケーションを終了します。

データ操作とナビゲーションのためのコードの作成

次の手順は、データの操作とナビゲーションを実装します。

❖ テーブルを開くには、次の手順に従います。

1　テーブルを初期化して最初のローに移動するためのコードを記述します。

このコードは、データベースの customer テーブルを CustomerTable 変数に割り当てます。Open を呼び出すとテーブルが開き、テーブル・データの読み込みや操作ができます。このコードは、アプリケーションをテーブル内にある最初のローの前に置きます。

次のコードを、Form_Load イベントの End Sub 命令の直前に挿入します。

```
Set CustomerTable = Connection.GetTable("customer")
 CustomerTable.Open
 If Err.Number <> ULSQLCode.ulSQLE_NOERROR Then
     MsgBox Err.Description
 End If
```

2　新しいプロシージャ DisplayCurrentRow を作成し、次のように実装します。

テーブルにローが存在しない場合は、次のプロシージャによって、アプリケーションは空のコントロールを表示します。ローが存在する場合は、データベースの現在のローの各カラムに格納された値を表示します。

```
Private Sub DisplayCurrentRow()
   If CustomerTable.RowCount = 0 Then
     txtFname.Text = ""
     txtLname.Text = ""
     txtCity.Text = ""
     txtPhone.Text = ""
     lblID.Caption = ""
   Else
     lblID.Caption = _
          CustomerTable.Column("Id").StringValue
     txtFname.Text = _
          CustomerTable.Column("Fname").StringValue
```

```
    txtLname.Text = _
         CustomerTable.Column("Lname").StringValue
    If CustomerTable.Column ("City").IsNull Then
      txtCity.text =""
    Else
      txtCity.Text = _
      CustomerTable.Column("City").StringValue
    End If
    If CustomerTable.Column("Phone").IsNull Then
      txtphone.Text = ""
    Else
      txtphone.Text = _
      CustomerTable.Column("Phone").StringValue
    End If
  End If
End Sub
```

3 Form_Activate オブジェクトから DisplayCurrentRow を呼び出します。この関数を呼び出すと、アプリケーションの起動時にフィールドが更新されます。

```
Private Sub Form_Activate()
     DisplayCurrentRow
End Sub
```

❖ テーブルにローを挿入するには、次の手順に従います。

1 [挿入] ボタンを実装するためのコードを記述します。

次のプロシージャでは、InsertBegin を呼び出すと、アプリケーションが挿入モードになり、ローのすべての値がデフォルトに設定されます。たとえば、ID カラムは、次のオートインクリメント値を受け取ります。カラム値が設定されると、新しいローが挿入されます。

次のプロシージャをフォームに追加します。

```
Private Sub btnInsert_Click()
     Dim fname As String
     Dim lname As String
     Dim city As String
     Dim phone As String

     fname = txtFname.Text
```

```
        lname = txtLname.Text
        city = txtCity.Text
        phone = txtPhone.Text

        On Error GoTo InsertError
        CustomerTable.InsertBegin
        CustomerTable.Column("Fname").StringValue = _
          fname
        CustomerTable.Column("Lname").StringValue = _
          lname
        If Len(city) > 0 Then
            CustomerTable.Column("City").StringValue =
_
          city
        End If
        If Len(phone) > 0 Then
           CustomerTable.Column("Phone").StringValue =
_
           phone
        End If
        CustomerTable.Insert
        CustomerTable.MoveLast
        DisplayCurrentRow
        Exit Sub

    InsertError:
        MsgBox "Error:  " & CStr(Err.Description)
    End Sub
```

2　アプリケーションを実行します。

最初のメッセージ・ボックスの後にフォームが表示されます。

3　2 つのローをデータベースに挿入します。

- 先頭のテキスト・ボックスに名前 Jane を入力し、2 つめのテキスト・ボックスに姓 Doe を入力します。[挿入] をクリックします。

  テーブルに、これらの値を持つローが追加されます。アプリケーションはテーブルの最後のローに移動し、そのローを表示します。ラベルには、Ultra Light がローに割り当てた ID カラムの自動的にインクリメントされた値が表示されます。

- 先頭のテキスト・ボックスに名前 John を入力し、2 つめのテキスト・ボックスに姓 Smith を入力します。[挿入] をクリックします。

4 [終了] をクリックしてプログラムを終了します。

❖ テーブルのローを移動するには、次の手順に従います。

1 [次へ] と [前へ] の各ボタンを実装するためのコードを記述します。

次のプロシージャをフォームに追加します。

```
Private Sub btnNext_Click()
     If Not CustomerTable.MoveNext Then
         CustomerTable.MoveLast
     End If
     DisplayCurrentRow
 End Sub
 Private Sub btnPrevious_Click()
     If Not CustomerTable.MovePrevious Then
         CustomerTable.MoveFirst
     End If
     DisplayCurrentRow
 End Sub
```

2 アプリケーションを実行します。

最初にフォームが表示されると、現在位置が最初のローの前にあるため、コントロールは空です。

フォームが表示されたら、[次へ] と [前へ] をクリックして、テーブルのローの間を移動します。

❖ テーブルのローを更新、削除するには、次の手順に従います。

1 [更新] ボタンを実装するためのコードを記述します。

下のコードでは、UpdateBegin を呼び出すと、アプリケーションが更新モードに設定されます。Update が呼び出されると、カラム値が更新された後にロー自体が更新されます。

次のプロシージャをフォームに追加します。

```
Private Sub btnUpdate_Click()
     Dim fname As String
     Dim lname As String
     Dim city As String
     Dim phone As String

     fname = txtFname.Text
     lname = txtLname.Text
     city = txtCity.Text
     phone = txtPhone.Text
     On Error GoTo UpdateError
     CustomerTable.UpdateBegin
     CustomerTable.Column("Fname").StringValue =
fname
     CustomerTable.Column("Lname").StringValue =
lname
     If Len(city) > 0 Then
         CustomerTable.Column("City").StringValue =
city
     Else
          CustomerTable.Column("City").SetNull
     End If
     If Len(phone) > 0 Then
        CustomerTable.Column("Phone").StringValue =
phone
     End If
     CustomerTable.Update
     DisplayCurrentRow
     Exit Sub

 UpdateError:
     MsgBox "Error:  " & CStr(Err.Description)
 End Sub
```

2　[削除] ボタンを実装するためのコードを記述します。

下のコードでは、Delete を呼び出すと、アプリケーションの現在のローが削除されます。

次のプロシージャをフォームに追加します。

```
Private Sub btnDelete_Click()
     If CustomerTable.RowCount = 0 Then
         Exit Sub
     End If
     CustomerTable.Delete
     CustomerTable.MoveRelative 0
     DisplayCurrentRow
 End Sub
```

3 アプリケーションを実行します。

同期のためのコードの記述

次の手順は、同期を実装します。同期には、SQL Anywhere Studio が必要です。チュートリアルのこの部分はオプションです。

❖ [同期] ボタンのコードを記述するには、次の手順に従います。

- [同期] ボタンを実装するためのコードを記述します。

  下のコードでは、ULSyncParms オブジェクトに同期パラメータが含まれています。たとえば、ULSyncParms.UserName プロパティは、Mobile Link が起動したら新しいユーザを追加することを指定します。ULSyncParms.SendColumnNames プロパティは、カラム名が Mobile Link に送信されることを指定し、アップロード・スクリプトとダウンロード・スクリプトを生成できるようにします。

  次のプロシージャをフォームに追加します。

```
Private Sub btnSync_Click()
     With Connection.SyncParms
         .UserName = "afsample"
         .Stream = ULStreamType.ulTCPIP
         .Version = "ul_default"
         .SendColumnNames = True
     End With
     Connection.Synchronize
     DisplayCurrentRow
 End Sub
```

アプリケーションの同期

ASA 9.0 サンプル・データベースの Customer テーブルは、作成した Ultra Light データベースの **customer** テーブルとカラムが一致しています。次の手順は、データベースを ASA 9.0 サンプル・データベースと同期します。

❖ アプリケーションを同期するには、次の手順に従います。

1. コマンド・プロンプトから、次のコマンド・ラインを実行して、Mobile Link 同期サーバを起動します。

   ```
   dbmlsrv9 -c "dsn=ASA 9.0 Sample" -v+ -zu+ -za
   ```

   `-zu+` と `-za` コマンド・ライン・オプションによって、ユーザの追加と同期スクリプトの生成が自動的に行われます。これらのオプションの詳細については、『Mobile Link 管理ガイド』>「Mobile Link 同期サーバのオプション」を参照してください。

2. Ultra Light アプリケーションを起動します。

3. テーブルのすべてのローを削除します。

   テーブルにローがあると、ASA 9.0 サンプル・データベースの Customer テーブルに、ローがアップロードされます。

4. アプリケーションを同期します。

   [同期] をクリックします。

   Mobile Link 同期サーバのウィンドウでは、同期の進行状況が表示されます。

5. 同期が完了したら、[次へ] と [前へ] をクリックしてテーブルのローの間を移動します。

レッスン 4：デバイスへの配備

次の手順は、アプリケーションを Palm OS または PocketPC デバイスに配備します。

❖ PocketPC デバイスに配備するには、次の手順に従います。

1 アプリケーションを設定します。

- [MobileVB] メニューから、[MobileVB 設定] を選択します。ダイアログが表示されます。
- 左ウィンドウ枠で、[依存性] を選択し、[ユーザ依存] タブをクリックします。
- [追加] ボタンをクリックし、*c:¥tutorial¥mvb¥tutCustomer.usm* を選択します。これは、このファイルが配備に組み込まれるように MobileVB に指定します。
- 左ウィンドウ枠で [Pocket PC 設定] 項目を選択します。
- [デバイスのインストール・パス] に *¥Tutorial¥mvb* と入力します。
- [OK] をクリックしてダイアログを閉じます。

2 [MobileVB] - [Deploy to Device] を選択し、[PocketPC] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。

3 3.0 より前のバージョンの MobileVB を実行している場合は、Ultra Light コントロールをデバイスにコピーする必要もあります。

ファイル *ulmvb9.dll* をデバイスの *¥Program Files¥AppForge* にコピーします。このファイルは、SQL Anywhere 9.0 インストール環境のプラットフォーム固有のサブディレクトリである、*¥UltraLite¥UltraLiteForMobileVB¥ce¥arm* または *¥UltraLite¥UltraLiteForMobileVB¥ce¥mips* にあります。この手順は、デバイスあたり 1 回だけ実行する必要があります。

4　デバイスで、[スタート] ー [プログラム] を選択します。

5　[UltraLiteTutorial] を選択し、アプリケーションを実行します。

❖ Palm デバイスに配備するには、次の手順に従います。

1　アプリケーションを設定します。

- [MobileVB] メニューから、[MobileVB 設定] を選択します。
- 表示されるダイアログの左ウィンドウ枠で [依存性] を選択し、[ユーザ依存] タブをクリックします。
- [追加] ボタンをクリックし、*c:¥tutorial¥mvb¥tutCustomer.pdb* を選択します。これは、このファイルが配備に組み込まれるように MobileVB に指定します。
- 左ウィンドウ枠で [Palm OS 設定] 項目を選択し、作成者 ID として Syb3 と入力します。有効な HotSync 名を選択します。
- [OK] をクリックしてダイアログを閉じます。

2　[MobileVB] - [Deploy to Device] を選択し、[Palm OS] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。

3　デバイスに対して HotSync を実行し、アプリケーションがデバイスに送信されることを確認します。HotSync 処理の完了後に、アプリケーション・ファイルはデバイスに抽出されます。

4　デバイスで [ホーム] をクリックし、[UltraLightTutorial] を選択してアプリケーションを実行します。

まとめ

学習の成果

このチュートリアルでは、以下の作業を行いました。

- データベース・スキーマの作成
- Ultra Light for MobileVB アプリケーションの作成
- Ultra Light リモート・データベースと Adaptive Server Anywhere 統合データベースの同期
- Ultra Light for MobileVB アプリケーションの開発プロセスに関する知識の取得

その他のサンプル

その他のサンプル・アプリケーションとユーティリティについては、iAnywhere CodeXchange を参照してください。

第 4 章

チュートリアル：AppForge Crossfire のサンプル・アプリケーション

この章の内容

このチュートリアルでは、AppForge Crossfire を使用して PocketPC または Palm OS デバイスの Ultra Light アプリケーションを構築するプロセスを解説します。

概要

このチュートリアルでは、AppForge Crossfire を使用して Ultra Light アプリケーションを構築する方法について説明します。チュートリアルを終了すると、アプリケーションと小規模なデータベースが Windows CE デバイス上に構築されます。これを、中央の統合データベースと同期します。

テーブル API の詳細については、『Ultra Light for MobileVB ユーザーズ・ガイド』> 「Ultra Light for MobileVB API リファレンス」を参照してください。

所要時間

このチュートリアルは、コードをコピーして貼り付ける場合、約 30 分で終了します。自分でコードを入力する場合、これより長い時間が必要です。

前提条件

このチュートリアルは、コンピュータに AppForge Crossfire がすでにインストールされていることを前提としています。また、Crossfire 開発についての基本的な知識があることも前提としています。

チュートリアルでは、Ultra Light スキーマ・ペインタを使用して、Ultra Light スキーマを作成する方法を知っていることも前提としています。詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「Ultra Light スキーマ・ペインタ」を参照してください。

注意

Crossfire ユーザは、SQL Anywhere Studio がなくても、このチュートリアルの大部分を実行できます。このチュートリアルの同期の項では、SQL Anywhere Studio が必要です。

レッスン 1：プロジェクト・アーキテクチャの作成

最初の手順では、Ultra Light データベース・スキーマを作成する方法を説明します。データベース・スキーマは、データベースに関する記述です。データベース・スキーマは、データベース内のテーブル、インデックス、キー、パブリケーションとそれらの間のすべての関係を記述します。

データベース・スキーマの詳細については、「Ultra Light データベース・スキーマ・ファイルの作成」12 ページを参照してください。

❖ Ultra Light データベース・スキーマを作成するには、次の手順に従います。

1　このチュートリアル用のディレクトリを作成します。

このチュートリアルでは、保存先ディレクトリを *c:¥Tutorial¥crossfire* とします。別の名前のディレクトリを作成した場合は、チュートリアルを通じてそのディレクトリを使用してください。

2　Ultra Light スキーマ・ペインタを使用して、データベース・スキーマを作成します。

データベース・スキーマの作成の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』>「レッスン 1：Ultra Light データベース・スキーマの作成」を参照してください。

- **スキーマのファイル名** tutcustomer.usm
- **テーブル名** Customer
- **Customer のカラム**

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

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
LName	char (20)	いいえ	なし
City	char (20)	はい	なし
Phone	char (12)	はい	555-1234

同期しているアプリケーションでは、通常はプライマリ・キーにグローバル・オートインクリメントまたは UUID データ型を選択します。ここでは、チュートリアルを高速にするためにオートインクリメント・メソッドを選択しています。

- **プライマリ・キー** 昇順 ID

3 データベース・スキーマを保存します。

Windows または Windows CE 用のアプリケーションを開発している場合は、[ファイル] ー [保存] を選択し、ファイル名としてチュートリアル・ディレクトリ内の *tutcustomer.usm* を選択します。

Palm OS のアプリケーションを開発している場合は、次の手順に従います。

a. [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。

b. [作成者 ID] に Syb3 と入力します。

c. チュートリアルのディレクトリに、そのファイルを *tutcustomer.pdb* として保存します。

Palm 作成者 ID に関する注意
作成者 ID は、Palm によってユーザに割り当てられます。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 を使用してください。

Crossfire プロジェクトの作成

次の手順では、アプリケーションの AppForge Crossfire プロジェクトを作成し、Ultra Light コントロールへの参照を追加します。

開発環境の左側にあるツールバーに、標準の Visual Basic ツールに加えて AppForge ツールが表示されます。

❖ Ultra Light の Crossfire プロジェクトを作成するには、次の手順に従います。

1　Crossfire を起動します。

a. [スタート] ー [プログラム] ー [AppForge] ー [Crossfire] を選択します。Crossfire Project Manager が表示されます。

b. [新しいプロジェクト] を選択します。[Microsoft Development Environment 新しいプロジェクト] ダイアログが表示されます。

c. [プロジェクトの種類] ウィンドウで、[Visual Basic プロジェクト] フォルダを開きます。

d. [テンプレート] ウィンドウで、[Crossfire Application] をクリックします。

e. プロジェクト名を CrossfireApp1 のままにし、チュートリアル・ディレクトリ (*c:¥tutorial¥crossfire*) をロケーションとして入力します。[OK] をクリックします。

f. 使用する配備プラットフォームを選択して [OK] をクリックすると、プロジェクトが作成されます。

Microsoft Visual Basic .NET Development Environment に [Crossfire] フォームが表示されます。

2　ツールボックスが表示されない場合は、[表示] ー [ツールボックス] を選択して開きます。[AppForge] タブを開きます。

3　AppForge コントロールのリストを下にスクロールし、[ULConnectionParms Class] をダブルクリックします。

トラブルシューティング

Crossfire プロジェクトに iAnywhere.UltraLiteForAppForge.dll への参照が含まれていない場合、また ULConnectionParms クラスが AppForge コントロールのリストに表示されない場合は、Ultra Light を Crossfire に登録する必要があります。この状況は、SQL Anywhere をインストールしてから Crossfire をインストールした場合などに発生することがあります。

Crossfire への Ultra Light の追加の手順については、「Crossfire 設計環境への Ultra Light の追加」10 ページを参照してください。

次の作業

これで、Ultra Light データベース・スキーマと、フォームに Ultra Light コントロールのある Crossfire プロジェクトが作成されました。次の手順では、アプリケーション・インタフェースを作成します。

レッスン 2：アプリケーション・インタフェースの作成

次の手順は、フォームを使用してユーザ・インタフェースを作成します。この例では、実際のアプリケーションと同様に、ラベルを出力として、テキスト・ボックスとボタンを入力として使用します。

❖ プロジェクトにコントロールを追加するには、次の手順に従います。

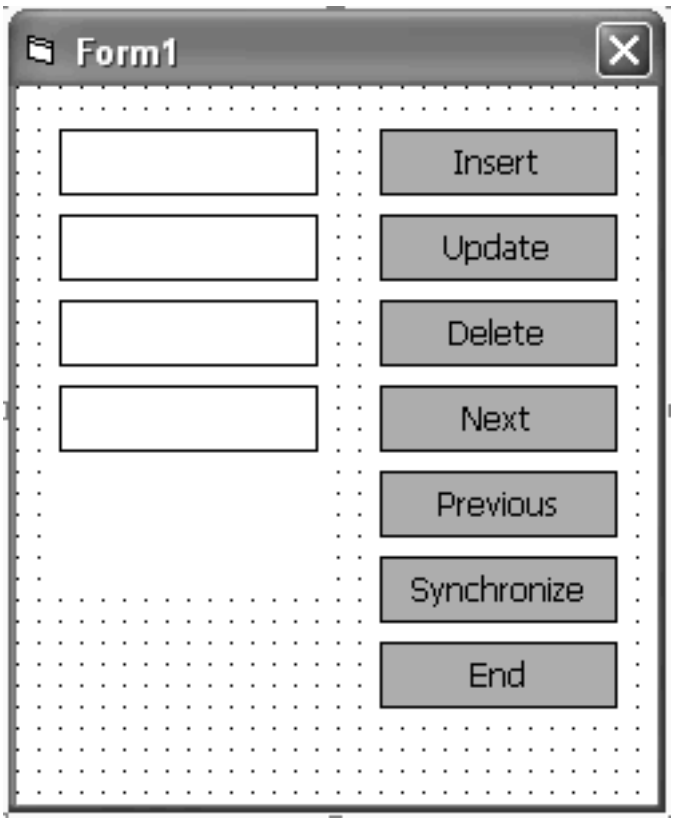
1　AppForge コントロールから、次のコントロールをフォームに追加します。

タイプ	名前	キャプションまたはテキスト
TextBox	txtFName	
TextBox	txtLName	
TextBox	txtcity	
TextBox	txtphone	
Label	lblID	
Button	btnInsert	挿入
Button	btnUpdate	更新
Button	btnDelete	削除
Button	btnNext	次へ
Button	btnPrevious	前へ
Button	btnSync	同期
Button	btnDone	終了

2　アプリケーションを確認します。

- [ビルド] ー [ソリューションのビルド] を選択します。

フォームは次のように表示されます。



レッスン 3：サンプル・コードの作成

このレッスンでは、データベースへの接続、データベース内のナビゲーション、データベース内のデータの操作を行うためのコードの作成について解説します。

このレッスンには、アプリケーションを Adaptive Server Anywhere データベースと同期するための指示も含まれています。レッスンのこの部分はオプションで、SQL Anywhere Studio を必要とします。

データベースとの接続のためのコードの作成

このアプリケーションでは、Form_Load イベントの間にデータベースに接続します。一般のモジュールを使用しても、データベースに接続できます。

この例では、ULConnectionParms オブジェクトを使用して *tutcustomer* データベースに接続します。

❖ Ultra Light データベースに接続するためのコードを作成するには、次の手順に従います。

1　フォームをダブルクリックして、[コード] ウィンドウを開きます。

2　必要な Ultra Light オブジェクトを宣言します。

`Public Class CrossfireForm1 Inherits System.Windows.Forms.Form` 行の直後に次のコードを入力します。

```
Public DatabaseMgr As New
UltraLiteAFLib.ULDatabaseManager
 Public Connection As UltraLiteAFLib.ULConnection
 Public CustomerTable As UltraLiteAFLib.ULTable
```

3　接続パラメータを指定します。

- ULConnectionParm1 コントロールを選択します。[プロパティ] ウィンドウで、このデータベースの接続プロパティを指定します。

  Windows CE デバイスに配備する場合、次のプロパティを指定します。

プロパティ	値
DatabaseOnCE	\tutorial\crossfire\tutCustomer.udb
DatabaseOnDesktop	c:\tutorial\crossfire\tutCustomer.udb
SchemaOnCE	\tutorial\crossfire\tutCustomer.usm
SchemaOnDesktop	c:\tutorial\crossfire\tutCustomer.usm

  Palm デバイスに配備する場合、次のプロパティを指定します。

プロパティ	値
DatabaseOnDesktop	c:\tutorial\crossfire\tutCustomer.pdb
DatabaseOnPalm	Syb3
SchemaOnDesktop	c:\tutorial\crossfire\tutCustomer.usm
SchemaOnPalm	tutCustomer

4 Form_Load イベントに、データベースに接続するためのコードを追加します。

  標準的な接続方法では、接続文字列で指定されるデータベースへの接続を開こうとします。データベースが存在しない場合は、スキーマを使用して新しいデータベースを作成します。Crossfire では、.GetOcx を ULConnectionParms1 オブジェクトに追加します。

```
Private Sub CrossfireForm1_Load(ByVal sender As
System.Object, _
     ByVal e As System.EventArgs _
 ) Handles MyBase.Load
    Try
       Connection = _
          DatabaseMgr.OpenConnectionWithParms( _
          ULConnectionParms1.GetOcx)
       ' if the database does not exist, create one
       MsgBox("Connected to an existing database")
    Catch
       If Err.Number = _

UltraLiteAFLib.ULSQLCode.ulSQLE_ULTRALITE_DATABASE_
NOT_FOUND _
       Then
          Err.Clear()
          Connection = _
          DatabaseMgr.CreateDatabaseWithParms( _
             ULConnectionParms1.GetOcx)
          If Err.Number =
UltraLiteAFLib.ULSQLCode.ulSQLE_NOERROR _
          Then
             MsgBox("Connected to a new database")
          Else
             MsgBox(Err.Description)
          End If
       Else
          MsgBox(Err.Description)
       End If
    End Try
 End Sub
```

5　次のコードを、[終了] ボタン (btnDone) の Click イベントに追加します。

```
Connection.Close
 End
```

6　アプリケーションを実行します。

- [デバッグ] ー [開始] を選択します。
- 最初のメッセージ・ボックスの後に、フォームがロードされます。

- アプリケーションを終了するには、[終了] をクリックします。

データベースに最初に接続したときに、Ultra Light はスキーマ・ファイルからデータベース・ファイル (*tutCustomer.udb*) を作成します。アプリケーションをもう一度実行した場合は、メッセージ・ボックスに、接続が既存のデータベースに対するものであることが示されます。

トラブルシューティング

これで、データベースに接続する基本コードが完成しました。

接続しようとするとデータ型変換エラーが発生する場合は、ULConnectionParms1 オブジェクトに GetOcx メソッドを提供したことを確認してください。

スキーマ・ファイルが見つからない場合は、ULConnectionParms1 SchemaOnDesktop 設定で指されているロケーションにファイルが存在することを確認してください。

データ操作とナビゲーションのためのコードの作成

次の手順は、データの操作とナビゲーションを実装します。コードでは、テーブル API を使用します。テーブル API では、テーブルのローを一度に 1 つずつ移動して変更する方法が提供されます。さらに複雑なアプリケーションでは、Ultra Light によって SQL の実装が提供されます。

❖ テーブルを開くには、次の手順に従います。

1 テーブルを初期化して最初のローに移動するためのコードを記述します。

このコードは、データベースの Customer テーブルを CustomerTable 変数に割り当てます。Open を呼び出すとテーブルが開き、テーブル・データの読み込みや操作ができます。このコードは、アプリケーションをテーブル内にある最初のローの前に置きます。

次のコードを、Form_Load イベントの End Sub 命令の直前に挿入します。

```
Try
    CustomerTable = Connection.GetTable("Customer")
    CustomerTable.Open()
 Catch
    If Err.Number <>
UltraLiteAFLib.ULSQLCode.ulSQLE_NOERROR _
    Then
       MsgBox(Err.Description)
    End If
 End Try
```

2　新しいプロシージャ DisplayCurrentRow を作成し、次のように実装します。

テーブルにローが存在しない場合は、次のプロシージャによって、アプリケーションは空のコントロールを表示します。ローが存在する場合は、データベースの現在のローの各カラムに格納された値を表示します。

```
Private Sub DisplayCurrentRow()
   If CustomerTable.RowCount = 0 Then
     txtFname.Text = ""
     txtLname.Text = ""
     txtCity.Text = ""
     txtPhone.Text = ""
     lblID.Caption = ""
   Else
     lblID.Caption = _
          CustomerTable.Column("ID").StringValue
     txtFname.Text = _
          CustomerTable.Column("FName").StringValue
     txtLname.Text = _
          CustomerTable.Column("LName").StringValue
     If CustomerTable.Column ("City").IsNull Then
       txtCity.text =""
     Else
       txtCity.Text = _
       CustomerTable.Column("City").StringValue
     End If
     If CustomerTable.Column("Phone").IsNull Then
       txtphone.Text = ""
     Else
```

```
        txtphone.Text = _
        CustomerTable.Column("Phone").StringValue
      End If
    End If
End Sub
```

3 フォームの Activated イベントから DisplayCurrentRow を呼び出します。この関数を呼び出すと、アプリケーションの起動時にフィールドが更新されます。

```
DisplayCurrentRow
```

❖ テーブルにローを挿入するには、次の手順に従います。

1 [挿入] ボタンを実装するためのコードを記述します。

次のプロシージャでは、InsertBegin を呼び出すと、アプリケーションが挿入モードになり、ローのすべての値がデフォルトに設定されます。たとえば、ID カラムは、次のオートインクリメント値を受け取ります。カラム値が設定されると、新しいローが挿入されます。

次のプロシージャを、[挿入] ボタン (btnInsert) の Click イベントに追加します。

```
Dim fname As String
 Dim lname As String
 Dim city As String
 Dim phone As String

 fname = txtFname.Text
 lname = txtLname.Text
 city = txtCity.Text
 phone = txtPhone.Text

 Try
     CustomerTable.InsertBegin
     CustomerTable.Column("FName").StringValue = _
       fname
     CustomerTable.Column("LName").StringValue = _
       lname
     If Len(city) > 0 Then
         CustomerTable.Column("City").StringValue =
_
```

```
            city
        End If
        If Len(phone) > 0 Then
           CustomerTable.Column("Phone").StringValue =
_
           phone
        End If
        CustomerTable.Insert
        CustomerTable.MoveLast
        DisplayCurrentRow
        Exit Sub
    Catch
        MsgBox "Error:  " & CStr(Err.Description)
    End Try
```

2　アプリケーションを実行します。

最初のメッセージ・ボックスの後にフォームが表示されます。

3　2 つのローをデータベースに挿入します。

- 先頭のテキスト・ボックスに名前 Jane を入力し、2 つめのテキスト・ボックスに姓 Doe を入力します。[挿入] をクリックします。

  テーブルに、これらの値を持つローが追加されます。アプリケーションはテーブルの最後のローに移動し、そのローを表示します。ラベルには、Ultra Light がローに割り当てた ID カラムの自動的にインクリメントされた値が表示されます。

- 先頭のテキスト・ボックスに名前 John を入力し、2 つめのテキスト・ボックスに姓 Smith を入力します。[挿入] をクリックします。

4　[終了] をクリックしてプログラムを終了します。

❖ テーブルのローを移動するには、次の手順に従います。

1　[次へ] と [前へ] の各ボタンを実装するためのコードを記述します。

次のコードを、[次へ] ボタン (btnNext) の Click イベントに追加します。

```
If Not CustomerTable.MoveNext Then
     CustomerTable.MoveLast
 End If
 DisplayCurrentRow
```

次のコードを、[前へ] ボタン (btnPrevious) の Click イベントに追加します。

```
If Not CustomerTable.MovePrevious Then
     CustomerTable.MoveFirst
 End If
 DisplayCurrentRow
```

2 アプリケーションを実行します。

最初にフォームが表示されると、現在位置が最初のローの前にあるため、コントロールは空です。

フォームが表示されたら、[次へ] と [前へ] をクリックして、テーブルのローの間を移動します。

この段階で、データの入力とテーブルのローのスクロールを行うことができます。

❖ テーブルのローを更新、削除するには、次の手順に従います。

1 [更新] ボタンを実装するためのコードを記述します。

下のコードでは、UpdateBegin を呼び出すと、アプリケーションが更新モードに設定されます。Update が呼び出されると、カラム値が更新された後にロー自体が更新されます。

次のコードを、[更新] ボタン (btnUpdate) の Click イベントに追加します。

```
Dim fname As String
 Dim lname As String
 Dim city As String
 Dim phone As String
```

```
fname = txtFname.Text
lname = txtLname.Text
city = txtCity.Text
phone = txtPhone.Text
Try
    CustomerTable.UpdateBegin
    CustomerTable.Column("FName").StringValue =
fname
    CustomerTable.Column("LName").StringValue =
lname
    If Len(city) > 0 Then
        CustomerTable.Column("City").StringValue =
city
    Else
        CustomerTable.Column("City").SetNull
    End If
    If Len(phone) > 0 Then
       CustomerTable.Column("Phone").StringValue =
phone
    End If
    CustomerTable.Update
    DisplayCurrentRow
    Exit Sub
Catch
    MsgBox "Error:  " & CStr(Err.Description)
End Try
```

2　[削除] ボタンを実装するためのコードを記述します。

下のコードでは、Delete を呼び出すと、アプリケーションの現在のローが削除されます。

次のコードを、[削除] ボタン (btnDelete) の Click イベントに追加します。

```
If CustomerTable.RowCount = 0 Then
    Exit Sub
End If
CustomerTable.Delete
CustomerTable.MoveRelative 0
DisplayCurrentRow
```

3　アプリケーションを実行します。

同期のためのコードの記述

次の手順は、同期を実装します。同期には、SQL Anywhere Studio が必要です。

❖ [同期] ボタンのコードを記述するには、次の手順に従います。

- [同期] ボタンを実装するためのコードを記述します。

  下のコードでは、ULSyncParms オブジェクトに同期パラメータが含まれています。たとえば、ULSyncParms.UserName プロパティは、Mobile Link が起動したら新しいユーザを追加することを指定します。ULSyncParms.SendColumnNames プロパティは、カラム名が Mobile Link に送信されることを指定し、アップロード・スクリプトとダウンロード・スクリプトを生成できるようにします。

  次のコードを、[同期] ボタン (btnSync) の Click イベントに追加します。

```
With Connection.SyncParms
    .UserName = "CrossfireSample"
    .Stream = UltraLiteAFLib.ULStreamType.ulTCPIP
    .Version = "ul_default"
    .SendColumnNames = True
End With
Connection.Synchronize
DisplayCurrentRow
```

アプリケーションの同期

ASA 9.0 サンプル・データベースの Customer テーブルは、作成した Ultra Light データベースの **customer** テーブルとカラムが一致しています。次の手順は、データベースを ASA 9.0 サンプル・データベースと同期します。

❖ アプリケーションを同期するには、次の手順に従います。

1　コマンド・プロンプトから、次のコマンド・ラインを実行して、Mobile Link 同期サーバを起動します。

```
dbmlsrv9 -c "dsn=ASA 9.0 Sample" -v+ -zu+ -za
```

-zu+ と -za コマンド・ライン・オプションによって、ユーザの追加と同期スクリプトの生成が自動的に行われます。これらのオプションの詳細については、『Mobile Link 管理ガイド』> 「Mobile Link 同期サーバのオプション」を参照してください。

2　Ultra Light Crossfire アプリケーションを起動します。

3　[削除] を繰り返しクリックして、テーブルのすべてのローを削除します。

テーブルにローがあると、ASA 9.0 サンプル・データベースの Customer テーブルに、ローがアップロードされます。

4　アプリケーションを同期します。

[同期] をクリックします。

Mobile Link 同期サーバのウィンドウでは、同期の進行状況が表示されます。

5　同期が完了したら、[次へ] と [前へ] をクリックしてテーブルのローの間を移動します。

レッスン 4：デバイスへの配備

次の手順は、アプリケーションを Palm OS または PocketPC デバイスに配備します。

❖ PocketPC デバイスに配備するには、次の手順に従います。

1 アプリケーションを設定します。

- [AppForge] メニューから、[Crossfire Settings] を選択します。ダイアログが表示されます。
- 左ウィンドウ枠で、[Dependencies] を選択し、[User Dependencies] タブをクリックします。
- [Add] ボタンをクリックし、*c:¥tutorial¥crossfire¥tutCustomer.usm* を選択します。これは、このファイルが配備に組み込まれるように Crossfire に指定します。
- 左ウィンドウ枠で [Pocket PC Settings] 項目を選択します。
- [Device Installation Path] に *¥tutorial¥crossfire* と入力します。
- [OK] をクリックしてダイアログを閉じます。

2 [AppForge] ー [Deploy to Device] を選択し、[PocketPC/Windows Mobile] を選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。

3 デバイスで、[スタート] ー [プログラム] を選択します。

4 [UltraLiteTutorial] を選択し、アプリケーションを実行します。

❖ Palm デバイスに配備するには、次の手順に従います。

1 アプリケーションを設定します。

- [AppForge] メニューから、[Crossfire Settings] を選択します。

- 表示されるダイアログの左ウィンドウ枠で [Dependencies] を選択し、[User Dependencies] タブをクリックします。
- [Add] ボタンをクリックし、*c:¥tutorial¥mvb¥tutCustomer.pdb* を選択します。これは、このファイルが配備に組み込まれるように Crossfire に指定します。
- 左ウィンドウ枠で [Palm OS Settings] 項目を選択し、作成者 ID として Syb3 と入力します。有効な HotSync 名を選択します。
- [OK] をクリックしてダイアログを閉じます。

2 [AppForge] － [Deploy to Device] を選択し、[Palm OS] デバイスを選択します。プロジェクトを保存するかどうかを確認するダイアログが表示されたら、[はい] を選択します。

3 デバイスに対して HotSync を実行し、アプリケーションがデバイスに送信されることを確認します。HotSync 処理の完了後に、アプリケーション・ファイルはデバイスに抽出されます。

4 デバイスで [ホーム] をクリックし、[UltraLightTutorial] を選択してアプリケーションを実行します。

まとめ

学習の成果

このチュートリアルでは、以下の作業を行いました。

- データベース・スキーマの作成
- Crossfire 用の Ultra Light アプリケーションの作成
- Ultra Light リモート・データベースと Adaptive Server Anywhere 統合データベースの同期

その他のサンプル

その他のサンプル・アプリケーションとユーティリティについては、iAnywhere CodeXchange を参照してください。

第 5 章

Ultra Light for MobileVB API リファレンス

この章の内容

この章では、UltraLite MobileVB API について説明します。これは、Ultra Light データベースを使用するアプリケーション用の MobileVB のコードを作成するためのクラスとメソッドのセットです。各トピックには、個別のクラス、メソッド、定数、または列挙に関する情報が含まれます。リファレンスはクラス別に編成されており、そのクラスに関連するメソッドが続けて示されます。

ULAuthStatusCode 列挙

ULAuthStatusCode は、ULSyncResult オブジェクトで使用される auth_status 同期パラメータです。

定数	値
ulAuthStatusUnknown	0
ulAuthStatusValid	1000
ulAuthStatusValidButExpiresSoon	2000
ulAuthStatusExpired	3000
ulAuthStatusInvalid	4000
ulAuthStatusInUse	5000

ULColumn クラス

ULColumn オブジェクトでは、データベース内のテーブルの値を取得、設定できます。各カラム・オブジェクトは、テーブル内の特定の値を表します。ローは、ULTable オブジェクトによって決定されます。

Ultra Light データベースの型を Visual Basic の型に変換する場合の注意

Ultra Light では、データベースのカラムのデータ型を Visual Basic のデータ型に変換しようとします。変換が成功しなかった場合は、ulSQLE_CONVERSION_ERROR が発生します。

テーブル・オブジェクトの詳細については、「ULTable クラス」185 ページを参照してください。

プロパティ

プロトタイプ	説明
BooleanValue as Boolean	現在のローのこのカラムの値を Boolean として取得または設定する。
ByteValue As Byte	現在のローのこのカラムの値を Byte として取得または設定する。
DatetimeValue As Date	現在のローのこのカラムの値を Date として取得または設定する。
DoubleValue As Double	現在のローのこのカラムの値を Double として取得または設定する。
IntegerValue As Integer	現在のローのこのカラムの値を Integer として取得または設定する。
IsNull As Boolean (読み込み専用)	カラム値が NULL かどうかを示す。
LongValue As Long	現在のローのこのカラムの値を Long として取得または設定する。

プロトタイプ	説明
RealValue As Single	現在のローのこのカラムの値を Single として取得または設定する。
Schema As ULColumnSchema (読み込み専用)	カラムのスキーマを表すオブジェクトを取得する。
StringValue As String	現在のローのこのカラムの値を String として取得または設定する。
UUIDValue As String	このカラムの値を UNIQUEIDENTIFIER データ型として取得または設定する。 このプロパティを取得する場合、Ultra Light は、UUID を表す文字列にカラム値を変換します。値が有効な UUID でない場合は、SQLE_CONVERSION_ERROR が発生します。 このプロパティを設定する場合、Ultra Light は、文字列形式の UUID をバイナリ値に変換してから、データベースに格納します。

AppendByteChunk メソッド

プロトタイプ

AppendByteChunk(_
data As Long**, _**
data_len As Long **_**
)
Member of **UltraLiteAFLib.ULColumn**

説明

カラムの型が ulTypeLongBinary または TypeBinary の場合、バイトをローのカラムに追加します。

パラメータ

data MobileVB では、バイトの配列へのポインタ。バイトの配列へのポインタを取得するには、Visual Basic VarPtr() 関数を使用します。Crossfire では、バイトの配列であるローカル変数です。

data_len 追加する配列からのバイトの数。

エラー・セット　**ulSQLE_INVALID_PARAMETER**　データ長が 0 より小さい場合は、エラーが発生します。

ulSQLE_CONVERSION_ERROR　カラムのデータ型が LONG BINARY でない場合、エラーが発生します。

例　次の例では、データが edata カラムに追加されます。

```
'MobileVB
 Dim data (1 to 512) As Byte
 ' ...
 table.Column("edata").AppendByteChunk( _
                   VarPtr(data(1)), 232)

'Crossfire
 Dim data (1 to 512) As Byte
 ' ...
 table.Column("edata").AppendByteChunk(data, 232)
```

AppendStringChunk メソッド

プロトタイプ　**AppendStringChunk(** *chunk* As String **)**
Member of **UltraLiteAFLib.ULColumn**

説明　カラムの型が TypeLongString または TypeString の場合、カラムに文字列を追加します。

パラメータ　**data**　テーブル内の既存の文字列に追加する文字列。

エラー・セット　**ulSQLE_CONVERSION_ERROR**　カラムのデータ型が CHAR または LONG VARCHAR でない場合、エラーが発生します。

GetByteChunk メソッド

プロトタイプ　**GetByteChunk (_**
offset As Long**, _**
data As Long**, _**
data_len As Long**, _**

```
    filled_len As Long _
) As Boolean
Member of UltraLiteAFLib.ULColumn
```

説明

TypeBinary または TypeLongBinary カラムからデータを取得します。

パラメータ

offset 基本となるバイト配列へのオフセット。ソース・オフセットは、0 以上であることが必要です。それ以外の場合は、ulSQLE_INVALID_PARAMETER エラーが発生します。

data バイトの配列へのポインタ。バイトの配列へのポインタを取得するには、Visual Basic VarPtr() 関数を使用します。

data_len バッファ (配列) の長さ。data_len は 0 以上であることが必要です。

filled_len これは OUT パラメータです。メソッドが呼び出された後で、有効なデータとともにフェッチされたバイト数を示します。BLOB データのサイズがあらかじめ知られていない場合は、BLOB データは、固定長のチャンクを使って、一度に 1 チャンクずつ、フェッチされます。最後にフェッチされるチャンクは、チャンク・サイズより小さくてもかまわないので、filled_len は、バッファ内にある有効なデータのバイト数を知らせます。

検査結果

True このカラム値にデータがまだほかにも含まれる場合

False データベース内のこのカラム値に、これ以上データがない場合

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER カラムのデータ型が BINARY でオフセットが 0 でも 1 でもない、またはデータ長が 0 より小さい場合は、エラーが発生します。

カラムのデータ型が LONG BINARY でオフセットが 1 より小さい場合も、エラーが発生します。

例

次の例で、edata はカラム名です。

```
'MobileVB
 Dim filled As Long
 Dim more_data As Boolean
 Dim data (1 to 512) As Byte
 more_data = table.Column("edata").GetByteChunk(0, _
 VarPtr(data(1)), 512, filled)

'Crossfire
 Dim filled As Long
 Dim more_data As Boolean
 Dim data (1 to 512) As Byte
 more_data = table.Column("edata").GetByteChunk(0, _
 data, 512, filled)
```

GetStringChunk メソッド

プロトタイプ

GetStringChunk(_
offset As Long**, _**
data As String**, _**
string_len As Long**, _**
filled_len As Long **_**
) As Boolean
Member of **UltraLiteAFLib.ULColumn**

説明

TypeString または TypeLongString カラムからデータを取得します。

パラメータ

offset　基本となるデータへの文字オフセット。ここから文字列の取得を開始します。

data　文字列データを受信する変数。

string_length　返す文字列の長さ。

filled_len　フェッチされる文字列の長さ。

検査結果

True データベースから取得するデータがまだある場合。

False これ以上データがない場合。

エラー

ulSQLE_CONVERSION_ERROR　カラムのデータ型が CHAR でも LONG VARCHAR でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER カラム・データ型が CHAR であり、src_offset が 64 K を超えている場合、エラーが発生します。

src_offset が 0 より小さいか、文字列の長さが 0 より小さい場合も、エラーが発生します。

SetByteChunk メソッド

プロトタイプ

SetByteChunk (_
data As Long**, _**
length As Long _
)
Member of **UltraLiteAFLib.ULColumn**

説明

TypeBinary または TypeLongBinary カラムにデータを設定します。

データを上書きしないで追加するには、『Ultra Light for MobileVB ユーザーズ・ガイド』> 「AppendByteChunk メソッド」を使用します。

パラメータ

data MobileVB では、バイトの配列へのポインタ。バイトの配列へのポインタを取得するには、Visual Basic VarPtr() 関数を使用します。Crossfire では、バイトの配列であるローカル変数です。

length 配列の長さ。

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER データ長が 0 より小さいか 64K より大きい場合は、エラーが発生します。

例

次の例では、edata はカラム名で、データ変数の最初の 232 バイトはデータベースに格納されています。

```
'MobileVB
 Dim data (1 to 512) As Byte
 ' ...
 table.Column("edata").SetByteChunk( VarPtr(data(1)),
232)
```

```
'Crossfire
 Dim data (1 to 512) As Byte
 ' ...
 table.Column("edata").SetByteChunk( data, 232)
```

SetNull メソッド

プロトタイプ　**SetNull( )**
Member of **UltraLiteAFLib.ULColumn**

説明　カラムの値を null に設定します。

SetToDefault メソッド

プロトタイプ　**SetToDefault( )**
Member of **UltraLiteAFLib.ULColumn**

説明　現在のカラムを、データベース・スキーマで定義されているデフォルト値に設定します。たとえば、オートインクリメント・カラムの場合は、次に使用可能な値が割り当てられます。

ULColumnSchema クラス

ULColumnSchema オブジェクトを使用して、テーブル内のメタデータ、つまりカラムの属性を取得できます。属性は、テーブルのデータに依存しません。

プロパティ

プロトタイプ	説明
AutoIncrement As Boolean (読み込み専用)	このカラムのデフォルトがオートインクリメント値かどうかを示す。オートインクリメントの場合は true です。
DefaultValue As String (読み込み専用)	ローの挿入時に値が指定されていない場合に使用される値を取得する。
GlobalAutoIncrement As Boolean (読み込み専用)	このカラムのデフォルトがグローバル・オートインクリメント値かどうかを示す。
ID As Integer (読み込み専用)	カラムの ID を取得する。
Name As String (読み込み専用)	カラム名を取得する。
Nullable As Boolean (読み込み専用)	カラムが NULL を許可するかどうかを示す。
OptimalIndex As ULIndexSchema (読み込み専用)	最初のカラムとしてこのカラムを持つインデックスを取得する。
Precision As Integer (読み込み専用)	型が ulTypeNumeric である場合は、カラムの精度値を取得する。
Scale As Integer (読み込み専用)	型が ulTypeNumeric である場合は、カラムの位取りの値を取得する。
Size As Long (読み込み専用)	バイナリ、数値、char データ型のカラム・サイズを取得する。

プロトタイプ	説明
SQLType As ULSQLType (読み込み専用)	作成時にカラムに割り当てられた SQL 型を取得する。

ULConnection クラス

ULConnection オブジェクトは、Ultra Light データベース接続を表します。これは、テーブルなどのデータベース・オブジェクトを取得するメソッドと同期のメソッドを提供します。

同期の進行状況を受け取るための WithEvents の使用
同期を行う場合、ULConnection オブジェクトは進行情報も受け取ることができます。この情報を受け取る場合は、接続を WithEvents で宣言します。接続を WithEvents で宣言しなくても同期を実行できますが、接続オブジェクトは同期の進行の通知を受け取りません。

例

接続を **WithEvents** で宣言するには、MobileVB フォームで次の構文を使用します。

```
Public WithEvents Connection As ULConnection
```

WithEvents の追加により、同期の進行情報を受け取れます。

プロパティ

ULConnection のプロパティは次のとおりです。

プロトタイプ	説明
AutoCommit As Boolean	AutoCommit 値を示す。true の場合、データを変更するとその直後にすべてのデータ変更がコミットされます。それ以外の場合、Commit を呼び出すまで、変更はデータベースにコミットされません。デフォルトでは、このプロパティは True です。
CollationName As String (読み込み専用)	データベース文字セットとソート順を取得する。

プロトタイプ	説明
DatabaseID As Long	グローバル・オートインクリメントのカラムの開始値を決定するデータベース ID を取得または設定する。 データベース ID が設定されていない場合、値は -1 です。
DatabaseNew As Boolean (読み込み専用)	この接続に対してデータベースが新しく作成されたかどうかを示す。
GlobalAutoIncrementUsage As Integer (読み込み専用)	利用可能なグローバル・オートインクリメントの値の使用済み比率 (%) を取得する。
IsCaseSensitive As Boolean (読み込み専用)	データベースで大文字と小文字が区別されるかどうかを示す。
LastIdentity As Long (読み込み専用)	デフォルトでオートインクリメントまたはグローバル・オートインクリメントに設定されたカラムに最後に挿入された値を取得する。
OpenParms As String (読み込み専用)	データベースへの接続を開くために使用される文字列を取得する。
Schema As ULDatabaseSchema (読み込み専用)	データベースの定義を表す ULDatabaseSchema オブジェクトを取得する。
SQLErrorOffset As Integer (読み込み専用)	PrepareStatement がエラーを発生した場合、エラーが記録された SQL 文で 1 から始まるオフセットを示す。この値が 0 以下の場合、オフセット情報はありません。
SyncParms As ULSyncParms (読み込み専用)	同期パラメータ・オブジェクトを取得する。
SyncResult As ULSyncResult (読み込み専用)	最後の同期の結果を取得する。

CancelSynchronize メソッド

プロトタイプ

CancelSynchronize( )
Member of **UltraLiteAFLib.ULConnection**

説明

同期処理中に呼び出された場合、このメソッドは同期をキャンセルします。ユーザがこのメソッドを呼び出せるのは、いずれかの同期イベントの実行時だけです。

これを可能にするには、ULConnection オブジェクトを **WithEvents** で宣言します。

ChangeEncryptionKey メソッド

プロトタイプ

ChangeEncryptionKey(*newkey*As String **)**
Member of **UltraLiteAFLib.ULConnection**

説明

指定されたキーを使用してデータベースを暗号化します。

パラメータ

newkey データベースの新しい暗号化キーの値。

例

CreateDatabaseWithParms を呼び出し parms オブジェクトを渡す場合、EncryptionKey に値が指定されていれば、データベースは暗号化されて作成されます。暗号化キーを変更する別の方法は、新しい暗号化キーを ULConnection オブジェクトで指定することです。この例では "apricot" がキーです。

```
Connection.ChangeEncryptionKey( "apricot" )
```

OpenConnectionWithParms のようなデータベースへの接続は、データベースが暗号化されてから、EncryptionKey プロパティにも *apricot* を指定します。そうしないと、接続は失敗します。

Close メソッド

プロトタイプ

Close( )
Member of **UltraLiteAFLib.ULConnection**

説明　データベースとの接続を閉じます。この接続で ULConnection オブジェクトなどのデータベース・オブジェクトのメソッドは、このメソッドを呼び出す前に呼び出します。接続が明示的に閉じられていない場合は、アプリケーションの終了時に暗黙的に閉じられます。

Commit メソッド

プロトタイプ

```
Commit( )
    Member of UltraLiteAFLib.ULConnection
```

説明　未処理の変更をデータベースにコミットします。AutoCommit が false の場合にのみ役立ちます。

詳細については、ULConnection 「プロパティ」112 ページの Autocommit を参照してください。

CountUploadRows メソッド

プロトタイプ

```
CountUploadRows(
    [ mask As Long = 0 ], _
    [ threshold As Long = -1 ] _
) As Long
    Member of UltraLiteAFLib.ULConnection
```

説明　次回の同期でアップロードする必要のあるロー数を返します。

パラメータ　**mask**　チェックするパブリケーションを示す、オプションのユニークな識別子。すべてのパブリケーションをチェックする場合は 0 を使用します。指定されていない場合は、値は 0 になります。

threshold　カウントするローの最大数を表す、オプションのパラメータ。最大数がないことを指定するには、-1 を使用します。指定されていない場合、この値は -1 です。

検査結果　次回の同期でアップロードする必要のあるロー数を返します。

GetNewUUID メソッド

プロトタイプ

GetNewUUID( ) As String
Member of **UltraLiteAFLib.ULConnection**

説明

新しいユニバーサル・ユニーク識別子を返します。値は *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* 形式の文字列であり、通常はデータ型 UNIQUEIDENTIFIER のカラムに格納されます。

検査結果

呼び出すたびに新しい UUID が返されます。

GetTable メソッド

プロトタイプ

GetTable(*name* As String **)** As ULTable
Member of **UltraLiteAFLib.ULConnection**

説明

指定されたテーブルの **ULTable** オブジェクトを返します。テーブルを開いてから、テーブルのデータを読み込みます。

パラメータ

name 調べるテーブルの名前。

検査結果

ULTable オブジェクトを返します。

GrantConnectTo メソッド

プロトタイプ

GrantConnectTo(
userid As String**,** _
password As String _
)
Member of **UltraLiteAFLib.ULConnection**

説明

指定したパスワードを使用してデータベースに接続するパーミッションを特定のユーザに付与します。

パラメータ

userid 接続する権限を付与されるユーザ ID。

password ユーザ ID が接続するのに指定するパスワード。

LastDownloadTime メソッド

プロトタイプ

LastDownloadTime([*mask* As Long = 0] **)** As Date
Member of **UltraLiteAFLib.ULConnection**

説明

パブリケーションの最終ダウンロード時間を返します。

パラメータ

mask　チェックするパブリケーションを示す、オプションのユニークな識別子。すべてのパブリケーションをチェックする場合は 0 を使用します。このパラメータが省略されている場合は、0 が使用されます。

検査結果

日付形式で表した最後のダウンロード時刻。

OnReceive イベント

プロトタイプ

```
OnReceive(
    nBytes As Long, _
    nInserts As Long, _
    nUpdates As Long, _
    nDeletes As Long _
)
```
Member of **UltraLiteAFLib.ULConnection**

説明

Mobile Link を介した、統合データベースからアプリケーションへのダウンロード情報をレポートします。このイベントは、複数回呼び出すことができます。

パラメータ

nBytes　リモート・アプリケーションで受信した、統合データベースからの累積バイト数。

nInserts　リモート・アプリケーションで受信した、統合データベースからの挿入の累積回数。

nUpdates　リモート・アプリケーションで受信した、統合データベースからの更新の累積回数。

nDeletes　リモート・アプリケーションで受信した、統合データベースからの削除の累積回数。

例

このメソッドの例については、CustDB アプリケーションを参照してください。

OnSchemaUpgradeProgress イベント

プロトタイプ

OnSchemaUpgradeProgress (
nProgress As Long, _
nFinalProgress As Long, _
nOperations As Long _
)
Member of **UltraLiteAFLib.ULConnection**

説明

ステータス・ダイアログに表示するスキーマのアップグレードの進行状況をレポートします。

パラメータ

- **nProgress** 現時点までの進行状況の近似値。この値は 0 ～ nFinalProgress の値で、処理の完了比率 (%) をダイアログ・ボックスに表示できます。
- **nFinalProgress** アップグレードが正常に完了したときの nProgress の値。
- **nOperations** アップグレードが進行している中で終了した作業量の近似値。値は 0 から開始し、アップグレードの進行に伴い増加します。nProgress よりも頻繁に更新されます。ほかのスキーマ・アップグレードと比較するための相対的な計測値として使用できます。

参照

- 『Ultra Light データベース・ユーザーズ・ガイド』> 「スキーマのアップグレードの監視」

OnSchemaUpgradeStateChange イベント

プロトタイプ

OnSchemaUpgradeStateChange (
newState As ULSchemaUpgradeState, _
oldState As ULSchemaUpgradeState _
)

説明　このイベントは、データベース・スキーマのアップグレード中にアップグレードが新しいステータスになるとトリガされます。使用可能なステータスは、ULSchemaUpgradeState 列挙に指定されます。

パラメータ

- **newState** アップグレードが開始されたときのステータス。
- **oldState** 完了したステータス。

参照

- 『Ultra Light データベース・ユーザーズ・ガイド』>「スキーマのアップグレードの監視」
- 『Ultra Light for MobileVB ユーザーズ・ガイド』>「ULSchemaUpgradeState 列挙」

OnSend イベント

プロトタイプ

```
OnSend(
    nBytes As Long, _
    nInserts As Long, _
    nUpdates As Long, _
    nDeletes As Long _
)
Member of UltraLiteAFLib.ULConnection
```

説明　Mobile Link を介した、リモート・データベースから統合データベースへのアップロード情報をレポートします。このイベントは、複数回呼び出すことができます。

パラメータ

nBytes　Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した累積バイト数。

nInserts　Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した挿入の累積回数。

nUpdates　Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した更新の累積回数。

nDeletes　Mobile Link を介して、リモート・アプリケーションが統合データベースに送信した削除の累積回数。

例　このメソッドの例については、CustDB アプリケーションを参照してください。

OnStateChange イベント

プロトタイプ

```
OnStateChange(
    newState As ULSyncState, _
    oldState As ULSyncState _
)
Member of UltraLiteAFLib.ULConnection
```

説明

このイベントは、同期ステータスが変更されるたびに呼び出されます。詳細については、『Ultra Light for MobileVB ユーザーズ・ガイド』 > 「ULSyncState 列挙」を参照してください。

パラメータ

newState 直後に開始される同期処理のステータス。

oldState 直前に完了した同期処理のステータス。

例

このメソッドの例については、CustDB アプリケーションを参照してください。

OnTableChange イベント

プロトタイプ

```
OnTableChange(
    newTableIndex As Long, _
    numTables As Long _
)
Member of UltraLiteAFLib.ULConnection
```

説明

このイベントは、同期処理が次のテーブルの同期を開始するたびに呼び出されます。

パラメータ

newTableIndex 現在同期が行われているテーブルのインデックス番号。この番号はテーブル ID とは異なるため、ULDatabaseSchema.GetTableName メソッドでは使用できません。

numTables 同期が行われるテーブル数。

例

このメソッドの例については、CustDB アプリケーションを参照してください。

PrepareStatement メソッド

プロトタイプ

PrepareStatement(
sqlStatement As String**, _**
persistent_name As String **_**
) As ULPreparedStatement
Member of **UltraLiteAFLib.ULConnection**

説明

SQL 文の実行を準備します。

パラメータ

sqlStatement　準備する SQL 文。

persistent_name　Palm アプリケーションでは、文の持続的な名前。

検査結果

ULPreparedStatement を返します。文を準備するときに問題が起こった場合は、エラーが発生します。エラーが発生した文へのオフセットは、SQLErrorOffset プロパティから判断できます。

ResetLastDownloadTime メソッド

プロトタイプ

ResetLastDownloadTime([*mask* As Long] **)**
Member of **UltraLiteAFLib.ULConnection**

説明

マスクで指定されたパブリケーションの最後のダウンロードの時刻をリセットします。

パラメータ

mask　リセットするパブリケーションのマスク。デフォルトは 0 で、すべてのパブリケーションを指定します。

RevokeConnectFrom メソッド

プロトタイプ

RevokeConnectFrom(*userID* As String **)**
Member of **UltraLiteAFLib.ULConnection**

説明

指定されたユーザがデータベースに接続できないようにします。

パラメータ

userid　接続する権限を取り消されるユーザ ID。

Rollback メソッド

プロトタイプ

Rollback( )
Member of **UltraLiteAFLib.ULConnection**

説明

未処理の変更をデータベースにロールバックします。AutoCommit が false の場合にのみ役立ちます。

RollbackPartialDownload メソッド

失敗した同期から変更をロールバックします。

プロトタイプ

RollbackPartialDownload ( )
Member of **UltraLiteAFLib.ULConnection**

説明

同期のダウンロード時に通信エラーが発生した場合、Ultra Light では、ダウンロードした変更を適用し、同期が中断した時点から同期を再開することができます。ダウンロードした変更が不要な場合 (ダウンロードが中断した時点での再開を望まない場合)、RollbackPartialDownload を使用することで、失敗したダウンロード・トランザクションをロールバックします。

参照

- 『Mobile Link 管理ガイド』> 「失敗したダウンロードの再開」
- 『Mobile Link クライアント』> 「Keep Partial Download 同期パラメータ」
- 『Mobile Link クライアント』> 「Partial Download Retained 同期パラメータ」
- 『Mobile Link クライアント』> 「Resume Partial Download 同期パラメータ」

StartSynchronizationDelete メソッド

プロトタイプ

StartSynchronizationDelete( )
Member of **UltraLiteAFLib.ULConnection**

説明

StartSynchronizationDelete が呼び出されると、すべての削除操作がもう一度同期されます。

StopSynchronizationDelete メソッド

プロトタイプ

StopSynchronizationDelete( )
Member of **UltraLiteAFLib.ULConnection**

説明

この関数が呼び出されると、削除操作が同期されなくなります。領域を節約するために、Ultra Light データベースから古い情報を削除して、統合データベースにはそれを残しておく場合に使用すると便利です。

StringToUUID メソッド

プロトタイプ

StringToUUID(
s_uuid As String**, _**
buffer_16_bytes As Long **_**
)
Member of **UltraLiteAFLib.ULConnection**

説明

形式が xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx の文字列として表されるユニバーサル・ユニーク識別子を 16 バイトのバイト配列に変換します。MobileVB アプリケーションでは、文字列形式で参照するのが便利な場合があります。したがって、ULColumn オブジェクトの UUIDValue プロパティは、文字列からバイナリ (16) へ、バイナリ (16) から文字列へ変換します。StringToUUID 関数は、MobileVB 文字列をバイト配列に簡単に変換します。この関数は、Ultra Light データベースをまったく参照しません。

バッファへのポインタにに関する注意

バッファへのポインタは、少なくとも 16 バイトとして宣言します。Visual Basic には境界チェックが用意されていないため、バッファが小さすぎるとメモリが上書きされることがあります。MobileVB では、VarPtr() 関数を使用して、バッファへのポインタを取得します。ULColumn.UUIDValue プロパティも参照してください。

新しいデータベースでは不要
バージョン 9.0.2 より前に作成されたデータベースでは、UNIQUEIDENTIFIER データ型はユーザ定義データ型として定義され、UUID 値のバイナリ表現と文字列表現の間を変換するための関数が必要です。

バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型はネイティブ・データ型であり、Ultra Light が必要に応じて変換を実行します。したがって、StringToUUID 関数は不要です。

詳細については、『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。

パラメータ

s_uuid 文字列として渡されるユニバーサル・ユニーク識別子。GetNewUUID を使用して新しい文字列 UUID を取得できます。

buffer_16_bytes 少なくとも 16 個の要素を持つ、バイト配列へのポインタ。VarPtr() 関数を使用して、ポインタ値を取得します。

例

次の例は、文字列形式の UUID 0a141e28-323c-4650-5a64-6e78828c96a0 をバイナリ配列に変換します。

```
Dim buff(1 to 16) As Byte
 conn.StringToUUID( "0a141e28-323c-4650-5a64-
6e78828c96a0", VarPtr(buff(1)) )
```

Synchronize メソッド

プロトタイプ

Synchronize( )
Member of **UltraLiteAFLib.ULConnection**

説明

Mobile Link を使用して統合データベースの同期を行います。この関数は、同期が完了するまで戻りませんが、接続が WithEvents で宣言されている場合はイベントを通知できます。

UUIDToString メソッド

プロトタイプ

UUIDToString(*buffer_16_bytes* As Long **)** As String
Member of **UltraLiteAFLib.ULConnection**

説明

UUID をバイト配列から *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* 形式の文字列に変換します。

新しいデータベースでは不要

バージョン 9.0.2 より前に作成されたデータベースでは、UNIQUEIDENTIFIER データ型はユーザ定義データ型として定義され、UUID 値のバイナリ表現と文字列表現の間を変換するための関数が必要です。

バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型はネイティブ・データ型であり、Ultra Light が必要に応じて変換を実行します。したがって、UUIDToString 関数は不要です。

詳細については、『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。

パラメータ

buffer_16_bytes　UUID を含む 16 バイトの配列。

検査結果

呼び出しごとに、*xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の形式の文字列が返されます。

ULConnectionParms クラス

ULConnectionParms オブジェクトを使用すると、ユーザ ID、パスワード、スキーマ・ファイル、デスクトップ上のファイルなど、接続を指定する多くのパラメータの設定ができます。

プロパティ

ULConnectionParms クラスは、Ultra Light データベースへの接続を開くためのパラメータを指定します。

Ultra Light for MobileVB では、フォームに ULConnectionParms オブジェクトがあり、ConnectionParms ダイアログで接続プロパティを設定したことを確認します。ULConnectionParms オブジェクトは、**ULDatabaseManager.CreateDatabaseWithParms** メソッドと **ULDatabaseManager.OpenConnectionWithParms** メソッドとともに使用します。

注意
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、ユーザ ID DBA とパスワード SQL を使用して、接続が開かれます。

これらのパラメータの意味の詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「接続パラメータ」を参照してください。

プロトタイプ	説明
AdditionalParms As String (読み込み／書き込み)	セミコロンで区切られた**名前** = *値*の組み合わせとして指定される追加のパラメータ。 『Ultra Light データベース・ユーザーズ・ガイド』> 「AdditionalParms 接続パラメータ」を参照してください。

プロトタイプ	説明
CacheSize As Integer (読み込み／書き込み)	キャッシュのサイズ。CacheSize の値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 KB なので、デフォルトのキャッシュ・サイズは 64 KB です。 『Ultra Light データベース・ユーザーズ・ガイド』>「CacheSize 接続パラメータ」を参照してください。
ConnectionName As String (読み込み／書き込み)	接続の名前。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。 『Ultra Light データベース・ユーザーズ・ガイド』>「ConnectionName 接続パラメータ」を参照してください。
DatabaseOnCE As String (読み込み／書き込み)	PocketPC に配備されるデータベースのファイル名。 『Ultra Light データベース・ユーザーズ・ガイド』>「DatabaseOnCE 接続パラメータ」を参照してください。
DatabaseOnDesktop As String (読み込み／書き込み)	開発中のデータベースのファイル名。 『Ultra Light データベース・ユーザーズ・ガイド』>「DatabaseOnDesktop 接続パラメータ」を参照してください。
DatabaseOnPalm As String (読み込み／書き込み)	Palm デバイス上の Ultra Light データベースの作成者 ID。 『Ultra Light データベース・ユーザーズ・ガイド』>「DatabaseOnPalm 接続パラメータ」を参照してください。

プロトタイプ	説明
EncryptionKey As String (読み込み／書き込み)	データベースを暗号化するためのキー。OpenConnection と OpenConnectionWithParms は、データベース作成中に指定されたキーと同じキーを使用する必要があります。キーは以下の条件を満たしていることが推奨されます。 1. 任意の長い文字列を選択します。 2. キーを見破られる可能性を減らすため、多様な数字、文字、特殊文字を使用した文字列を選択します。 『Ultra Light データベース・ユーザーズ・ガイド』>「EncryptionKey 接続パラメータ」を参照してください。
PageSize As Integer (読み込み／書き込み)	データベースのページ・サイズ。 『Ultra Light データベース・ユーザーズ・ガイド』>「PageSize 接続パラメータ」を参照してください。
ParmsUsed As String (読み込み専用)	ULDatabaseManager が使用するパラメータ。デバッグに便利です。
Password As String (読み込み／書き込み)	認証ユーザのパスワード。データベースは最初、1 つの認証されたユーザ・パスワード *SQL* を使用して、作成されます。データベースの大文字と小文字を区別しない場合は、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。デフォルト値は *SQL* です。 『Ultra Light データベース・ユーザーズ・ガイド』>「Password 接続パラメータ」を参照してください。

プロトタイプ	説明
ReserveSize As Integer (読み込み／書き込み)	Ultra Light の永続的データの保管に使用するため予約するファイル・システム領域の大きさ。 『Ultra Light データベース・ユーザーズ・ガイド』>「Reserve Size 接続パラメータ」を参照してください。
SchemaOnCE As String (読み込み／書き込み)	PocketPC に配備されるスキーマ・ファイル名。 『Ultra Light データベース・ユーザーズ・ガイド』>「SchemaOnCE 接続パラメータ」を参照してください。
SchemaOnDesktop As String (読み込み／書き込み)	開発中のスキーマ・ファイル名。 『Ultra Light データベース・ユーザーズ・ガイド』>「SchemaOnDesktop 接続パラメータ」を参照してください。
SchemaOnPalm As String (読み込み／書き込み)	Palm デバイス上のスキーマ PDB。 『Ultra Light データベース・ユーザーズ・ガイド』>「SchemaOnPalm 接続パラメータ」を参照してください。
UserID As String (読み込み／書き込み)	データベースで認証されたユーザ。データベースは最初、1 つの認証されたユーザ DBA を使用して、作成されます。データベースの大文字と小文字を区別しない場合は、UserID 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、UserID の大文字と小文字も区別します。デフォルト値は *DBA* です。 『Ultra Light データベース・ユーザーズ・ガイド』>「User ID 接続パラメータ」を参照してください。

プロトタイプ	説明
VFSOnPalm As Boolean (読み込み／書き込み)	Palm データベースが仮想ファイル・システムにあるか (true)、Palm ストアにあるか (false) を示す。 『Ultra Light データベース・ユーザーズ・ガイド』> 「VFSOnPalm パラメータ」 を参照してください。

ULDatabaseManager クラス

ULDatabaseManager クラスを使用して、接続とデータベースを管理します。アプリケーションは、このオブジェクトのインスタンスを 1 つだけ持ちます。データベースを作成し、そのデータベースへの接続を確立することは、Ultra Light の使用に必要な最初の手順です。CreateDatabaseWithParms、OpenConnectionWithParms、DropDatabaseWithParms を使用し、正しく接続してからデータ操作言語でデータベースを操作するように、コードに検査制約を含めることをおすすめします。

ULConnectionParms の使用の有無

データベースへの接続を作成、開始、削除するには、2 種類のメソッドがあります。WithParms メソッドと ULConnectionParms オブジェクトを使用しないメソッドです。WithParms メソッドは、ULConnectionParms オブジェクトを使用するメソッドで、簡単かつ正確に接続パラメータを操作できます。ULConnectionParms オブジェクトを使用しないメソッドでは、接続文字列を正しく作成し、その接続文字列を CreateDatabase、OpenConnection、または DropDatabase メソッドで使用することが要求されます。

プロパティ

ULDatabaseManager のプロパティは次のとおりです。

プロトタイプ	説明
Version As String (読み込み専用)	Ultra Light コンポーネントのバージョン文字列を取得する。

CreateDatabase メソッド

CreateDatabase は、新規データベースを作成し、そのデータベースへの接続を返します。

プロトタイプ

CreateDatabase(*parms* As String **)** As ULConnection
Member of **UltraLiteAFLib.ULDatabaseManager**

説明

新規データベースを作成し、そのデータベースへの接続を返します。指定したデータベースがすでに存在する場合は失敗します。データベースを正常に作成するには、有効なスキーマ・ファイルを指定します。既存のデータベースのスキーマを変更するには、ULDatabaseSchema ApplyFile メソッドを使用します。

警告
一度にアクティブにできるデータベースは 1 つのみです。ほかの接続が開いているときに別のデータベースを作成しようとすると、エラーが発生します。

ApplyFile の詳細については、「ULDatabaseSchema クラス」139 ページと「ApplyFile メソッド」140 ページを参照してください。

パラメータ

parms セミコロンで区切られたデータベース作成パラメータのリスト。

Palm ユーザに対する VFS カードについての注意
データベースを仮想ファイル・システムに配置する場合は、Palm_fs=vfs パラメータを CreateDatabase メソッドと OpenConnection メソッドの両方に対して指定してください。

接続パラメータの詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「接続パラメータ」を参照してください。

Palm_fs パラメータの詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「VFSOnPalm パラメータ」を参照してください。

検査結果

新しく作成された Ultra Light データベースへの接続を返します。

例

次のコードは、ULDatabaseManager オブジェクトを作成します。これは、Ultra Light for MobileVB のコードを記述するときに作成する最初のオブジェクトです。CreateDatabase では *.udb* ファイルがまだ存在していないことが必要であり、*.udb* ファイルがすでに存在している場合は OpenConnection が使用されます。

```
Dim conn_parms As String
 Dim open_parms As String
 Dim schema_parms As String

 conn_parms = "uid=DBA;pwd=SQL"
 open_parms = conn_parms & ";" & _

"PALM_DB=Syb3;file_name=c:¥tutorial¥tutCustomer.udb"
 schema_parms = open_parms & ";" & _
     "PALM_SCHEMA=tutCustomer;" & _
     "schema_file=c:¥tutorial¥tutCustomer.usm"

 On Error Resume Next

 Set Connection =
DatabaseMgr.OpenConnection(open_parms)
 If Err.Number =  _
     ULSQLCode.ulSQLE_DATABASE_NOT_FOUND _
 Then
     Err.Clear
         Set Connection = _
     DatabaseMgr.CreateDatabase(schema_parms)
     If Err.Number <> 0 Then
             MsgBox Err.Description
     End If
 End If
```

接続パラメータの詳細については、「OpenConnection メソッド」136 ページを参照してください。

CreateDatabaseWithParms メソッド

CreateDatabaseWithParms は、接続パラメータ・オブジェクトを使用して、新しいデータベースを作成し、そのデータベースへの接続を返します。

プロトタイプ

CreateDatabaseWithParms(*parms* As ULConnectionParms **)**
As ULConnection
Member of **UltraLiteAFLib.ULDatabaseManager**

説明

新規データベースを作成し、そのデータベースへの接続を返します。指定したデータベースがすでに存在する場合は失敗します。データベースを正常に作成するには、有効なスキーマ・ファイルを指定します。既存のデータベースのスキーマを変更するには、**ULDatabaseSchema.ApplyFileWithParms** メソッドを使用します。

警告
一度にアクティブにできるデータベースは 1 つのみです。ほかの接続が開いているときに別のデータベースを作成しようとすると、エラーが発生します。

パラメータ

parms 一連の接続パラメータを格納する ULConnectionParms オブジェクト。

Palm ユーザに対する VFS カードについての注意
ULConnectionParms オブジェクトで VFSOnPalm を指定します。

Palm_fs パラメータの詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「VFSOnPalm パラメータ」を参照してください。

検査結果

新しく作成された Ultra Light データベースへの接続を返します。指定したデータベースがすでに存在する場合は失敗します。

例

次の例は、フォーム上に ULConnectionParms オブジェクトを配置しており、それに **LoginParms** という名前を付け、[Connection parms] プロパティ・ウィンドウでデータベースのロケーションとスキーマのロケーションを指定していることを前提としています。

次のコードは、ULDatabaseManager オブジェクトを作成します。これは、Ultra Light for MobileVB のコードを記述するときに作成する最初のオブジェクトです。

CreateDatabaseWithParms では *.udb* ファイルがまだ存在していないことが必要であり、*.udb* ファイルがすでに存在している場合は OpenConnectionWithParms が使用されます。

```
DatabaseMgr.DropDatabaseWithParms LoginParms
 Set Connection =
DatabaseMgr.CreateDatabaseWithParms(LoginParms)
```

DropDatabase メソッド

DropDatabase メソッドは、データベース・ファイルを削除します。

プロトタイプ

DropDatabase(*parms* As String **)**
Member of **UltraLiteAFLib.ULDatabaseManager**

説明

データベース・ファイルを削除します。データベース・ファイルの情報はすべて失われます。指定されたデータベースが存在しない場合、または DropDatabase の実行時にオープン接続が存在する場合は、失敗します。

パラメータ

parms　データベースのファイル名。

例

次の例は、データベースを削除します。

```
Dim parms As String
 parms = "PALM_DB=Syb1;NT_FILE=c:¥temp¥ul_CustDB.udb"
 DropDatabase(parms)
```

DropDatabaseWithParms メソッド

DropDatabaseWithParms メソッドは、データベース・ファイルを削除します。

プロトタイプ

DropDatabaseWithParms(*parms* As ULConnectionParms **)**
Member of

説明

データベース・ファイルを削除します。データベース・ファイルの情報はすべて失われます。

パラメータ　**parms**　重要な接続パラメータを含む ULConnectionParms オブジェクト。

例　次の例は、**LoginParms** という名前の ULConnectionParms オブジェクトを宣言しインスタンス化しており、そのオブジェクトを使用してデータベースのロケーションを指定していることを前提としています。

```
DatabaseMgr.DropDatabaseWithParms LoginParms
```

OpenConnection メソッド

プロトタイプ　**OpenConnection(** *connparms* As string **)** As ULConnection
Member of **UltraLiteAFLib.ULDatabaseManager**

説明　データベースが存在する場合は、このメソッドを使用して接続を受信します。データベースが存在しない場合、または接続パラメータが無効な場合は、この呼び出しは失敗します。エラー・オブジェクトを使用して、呼び出しが失敗した理由を判別します。

この関数は、指定した Ultra Light データベースとのオープン接続を提供する ULConnection オブジェクトを返します。データベース・ファイル名は、connparms 文字列を使用して指定します。パラメータは、一連の**名前** = *値*の組み合わせを使用して指定します。ユーザ ID またはパスワードを指定しない場合は、デフォルトが使用されます。

次の形式で値を指定してください。

```
file_name=UDBFILE
 DBF=UDBFILE
 palm_db=CreatorID.
```

パラメータ　**connparms**　データベースへの接続を確立するために使用されるパラメータ。パラメータは、一連の**キーワード =** *値*の組み合わせを使用して指定します。ユーザ ID またはパスワードを指定しない場合は、デフォルトが使用されます。

Palm ユーザに対する注意
Palm 仮想ファイル・システムのデータベースを使用する場合は、Palm_fs=vfs パラメータを CreateDatabase メソッドと OpenConnection メソッドの両方に対して指定する必要があります。

検査結果

Palm_fs パラメータの詳細については、『Ultra Light データベース・ユーザーズ・ガイド』> 「VFSOnPalm パラメータ」を参照してください。

接続に成功した場合は、ULConnection オブジェクトが返されます。

例

次の例は、CustDB サンプル・アプリケーションからのデータベース接続を新しく作成します。

```
Set Connection = DatabaseMgr.OpenConnection(

"file_name=d:¥Dbfile.udb;palm_db=Syb3;CE_file=¥myapp¥My
DB.udb")
```

OpenConnectionWithParms メソッド

プロトタイプ

OpenConnectionWithParms(*connparms* As ULConnectionParms**)**
As ULConnection
Member of **UltraLiteAFLib.ULDatabaseManager**

説明

データベースが存在する場合は、このメソッドを使用して接続を受信します。データベースが存在しない場合、または接続パラメータが無効な場合は、この呼び出しは失敗します。エラー・オブジェクトを使用して、呼び出しが失敗した理由を判別します。

この関数は、指定した Ultra Light データベースとのオープン接続を提供する ULConnection オブジェクトを返します。データベース・ファイル名は、connparms オブジェクトを使用して指定します。パラメータは、一連の**名前** = *値*の組み合わせを使用して指定します。ユーザ ID またはパスワードを指定しない場合は、デフォルトが使用されます。

パラメータ

connparms　この接続を定義するパラメータ。

検査結果

接続に成功した場合は、ULConnection オブジェクトが返されます。

例

次の例は、フォーム上に ULConnectionParms オブジェクトを配置しており、それに **LoginParms** という名前を付け、[ULConnection parms] プロパティ・ウィンドウでデータベースのロケーションとスキーマのロケーションを指定していることを前提としています。

```
Set Connection = DatabaseMgr.OpenConnection(LoginParms)
```

ULDatabaseSchema クラス

ULDatabaseSchema オブジェクトを使用すると、接続先データベースの属性を取得できます。

プロパティ

ULDatabaseSchema のプロパティは次のとおりです。

プロトタイプ	説明
DateFormat As String (読み込み専用)	データベースから取り出した日付のフォーマットを取得する。デフォルトは、'YYYY-MM-DD' です。取得される日付のフォーマットは、スキーマ・ファイルの作成時に使用したフォーマットに対応しています。
DateOrder As String (読み込み専用)	日付フォーマットの解釈を示す。有効な値は 'MDY'、'YMD'、または 'DMY' です。
NearestCentury As String (読み込み専用)	文字列から日付への変換で、2 桁の年の解釈を示す。これは、ロールオーバ・ポイントとして動作する数値です。この値より小さい 2 桁の年は 20yy に変換され、この値以上の年は 19yy に変換されます。デフォルトは 50 です。
Precision As String (読み込み専用)	10 進法計算での結果の最大桁数を取得する。
PublicationCount As Integer (読み込み専用)	接続したデータベース内のパブリケーション数を取得する。
Signature As String (読み込み専用)	データベースのシグニチャを取得する。これは、データベース・スキーマを表す内部識別子です。
TableCount As Integer (読み込み専用)	接続したデータベース内のテーブル数を取得する。

プロトタイプ	説明
TimeFormat As String (読み込み専用)	データベースから取り出した時刻のフォーマットを取得する。
TimestampFormat As String (読み込み専用)	データベースから取り出したタイムスタンプのフォーマットを取得する。

ApplyFile メソッド

プロトタイプ

ApplyFile(*parms* As String **)**
Member of **UltraLiteAFLib.ULDatabaseSchema**

説明

このデータベースのスキーマを変更します。*Parms* は、データベースに適用しているスキーマ・ファイルを指します。このメソッドは、既存のデータベース構造を変更する場合にのみ役立ちます。

警告
ApplyFile は非常に安全ですが、次のような場合にデータの消失が起こることがあります。(1) カラムが削除された場合、(2) カラムのデータ型が互換性のない型に変更された場合、(3) Ultra Light 9.0 で ApplyFile を使用して 8.0.2 のデータベースを更新した場合です。

パラメータ

parms 変更するデータベース・スキーマを含むファイル。

例

```
DatabaseSchema.ApplyFile( _
 "schema_file=MySchemaFile.usm;palm_schema=MySchema" )
```

ApplyFileWithParms メソッド

プロトタイプ

ApplyFileWithParms(*parms* As ULConnectionParms)
Member of **UltraLiteAFLib.ULDatabaseSchema**

説明

パラメータ・オブジェクト *Parms* を使用して、このデータベースのスキーマを更新します。オブジェクトは、データベースに適用するスキーマ・ファイルを指しています。このメソッドは、既存のデータベース構造を変更する場合にのみ役立ちます。

警告
ApplyFile は非常に安全ですが、ApplyFileWithParms を使用すると、次のような場合にデータの消失が起こることがあります。(1) カラムが削除された場合、(2) カラムのデータ型が互換性のない型に変更された場合、(3) Ultra Light 9.0 で ApplyFile を使用して 8.0.2 のデータベースを更新した場合です。

パラメータ

parms　適用するスキーマ・ファイルを示すオブジェクト。

GetPublicationName メソッド

プロトタイプ

GetPublicationName(*id* As Integer **)** As String
Member of **UltraLiteAFLib.ULDatabaseSchema**

説明

指定したパブリケーションの名前を返します。パブリケーション *ID* の範囲は、1 ～ PublicationCount です。

パラメータ

id　*id* は、名前が返されるパブリケーションの識別子。

検査結果

接続しているデータベース内のパブリケーションの名前を返します。

ULPublicationSchema オブジェクトについては、「ULPublicationSchema クラス」152 ページを参照してください。

詳細については、ULDatabaseSchema 「プロパティ」139 ページを参照してください。

GetPublicationSchema メソッド

プロトタイプ

GetPublicationSchema(*Name* As String **)** As ULPublicationSchema
Member of **UltraLiteAFLib.ULDatabaseSchema**

説明　パブリケーション名を使用して ULPublicationSchema オブジェクトを取得します。

パラメータ　**name**　パブリケーションの *name*。

検査結果　ULPublicationSchema オブジェクトを返します。

GetTableName メソッド

プロトタイプ　**GetTableName(** *id* As Integer **)** As String
Member of **UltraLiteAFLib.ULDatabaseSchema**

説明　指定した *id* 値に対応する、接続しているデータベース内のテーブル名を返します。TableCount プロパティは、接続しているデータベース内のテーブル数を返します。各テーブルには、1 ～ TableCount 値までのユニークな番号があります。1 はデータベース内の最初のテーブルであり、2 はデータベース内の 2 番目のテーブルです。以下も同様に続きます。データベースのスキーマを変更した場合は、テーブルの id が変更されていることがあります。

パラメータ　**id**　テーブルの *id*。

検査結果　指定した *id* のテーブル名を返します。

ULIndexSchema クラス

ULIndexSchema オブジェクトを使用すると、インデックスの属性を取得できます。インデックスは順序付きカラムのセットであり、これを使用してテーブル内のデータをソートします。インデックスはおもに、1 つ以上のカラムでテーブルのデータを並べ替えるために使用されます。

インデックスに外部キーを使用することができ、この場合、データベースの参照整合性が維持されます。

プロパティ

プロトタイプ	説明
ColumnCount As Integer (読み込み専用)	インデックス内のカラム数を取得する。
ForeignKey As Boolean (読み込み専用)	外部キーかどうかを示す。
ForeignKeyCheckOnCommit (読み込み専用)	コミットが行われたときだけ (TRUE) または即座に (FALSE)、参照整合性をチェックするかどうかを示す。
ForeignKeyNullable (読み込み専用)	外部キーのカラムが NULL を許容するかどうかを示す。
Name As String (読み込み専用)	インデックスの名前を取得する。
PrimaryKey As Boolean (読み込み専用)	このテーブルのプライマリ・キーかどうかを取得する。
ReferencedIndexName As String (読み込み専用)	インデックスが外部キーの場合、このインデックスが参照するインデックスの名前を取得する。
ReferencedTableName As String (読み込み専用)	インデックスが外部キーの場合、このインデックスが参照するテーブルの名前を取得する。
UniqueIndex As Boolean (読み込み専用)	インデックスの値がユニークである必要があるかどうかを示す。

プロトタイプ	説明
UniqueKey As Boolean (読み込み専用)	インデックスがテーブルの一意性制約かどうかを示す。True の場合は、インデックス内のカラムはユニークであり、NULL 値を使用できません。

GetColumnName メソッド

プロトタイプ **GetColumnName(** *col_pos_in_index* As Integer **)** As String
Member of **UltraLiteAFLib.ULIndexSchema**

説明 インデックス内のカラム名を返すために使用します。パラメータ *col_pos_in_index* には 1 ～ ColumnCount までの値を設定します。

パラメータ **col_pos_in_index** インデックス内のカラム位置。

検査結果 インデックス内のカラム名を返します。

IsColumnDescending メソッド

プロトタイプ **IsColumnDescending(** *col_name* As String **)** As Boolean
Member of **UltraLiteAFLib.ULIndexSchema**

説明 インデックス内の指定したカラムが降順であるかどうかを示します。

パラメータ **col_name** インデックス・カラム名。

検査結果 **True** カラムが降順の場合

False カラムが昇順の場合

ULPreparedStatement クラス

ULPreparedStatement は、実行の準備ができたコンパイル済みの SQL 文を表します。準備文を使用して、SQL クエリを実行できます。ULPreparedStatement を使用すれば、多くの入力パラメータを使用して、同じ文を複数回実行することもできます。準備文はコンパイル済みなので、最初の実行以後の追加については、わずかな処理で済みます。複数のローで比較的速いデータ操作言語が必要な場合は、ULPreparedStatement と Dynamic SQL を使用します。

プロパティ

プロトタイプ	説明
HasResultSet As Boolean (読み込み専用)	準備文が結果セットを生成するかどうかを示す。 文が結果セットを持つ場合は true、持たない場合は false です。 true の場合、ExecuteStatement ではなく ExecuteQuery を呼び出します。
Plan (読み込み専用) As String	クエリを実行するのに Ultra Light が使用するアクセス・プランを取得する。このプロパティは、主に開発中の使用を目的とします。
ResultSetSchema As ULResultSetSchema (読み込み専用)	その文が結果セット用の場合、結果セットのスキーマの説明を取得する。

AppendByteChunkParameter メソッド

プロトタイプ

AppendByteChunkParameter (
param_id As Integer**,**
data As Long**,**
data_len As Long**)**
Member of **UltraLiteAFLib.ULPreparedStatement**

説明

カラムの型が ulTypeLongBinary の場合、バイトのバッファをローのカラムに追加します。

パラメータ

parameter_id 1 から始まるパラメータ番号を設定します。

data 追加するバイトの配列。

data_len 追加する配列からのバイトの数。

エラー・セット

ulSQLE_INVALID_PARAMETER データ長が 0 より小さい場合は、エラーが発生します。

ulSQLE_CONVERSION_ERROR カラムのデータ型が LONG BINARY でない場合、エラーが発生します。

AppendStringChunkParameter メソッド

プロトタイプ

AppendStringChunkParameter(
param_id As Integer **,**
chunk As String **)**
Member of **UltraLiteAFLib.ULPreparedStatement**

説明

カラムの型が ulTypeLongString の場合、カラムに文字列を追加します。

パラメータ

parameter_id 1 から始まるパラメータ番号を設定します。

chunk テーブル内の既存の文字列に追加する文字列。

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が LONG VARCHAR でない場合、エラーが発生します。

Close メソッド

プロトタイプ　**Close( )**
Member of **UltraLiteAFLib.ULPreparedStatement**

説明　ULPreparedStatement に関連付けられているリソースを解放します。

ExecuteQuery メソッド

プロトタイプ　**ExecuteQuery( )** As ULResultSet
Member of **UltraLiteAFLib.ULPreparedStatement**

説明　クエリを実行し結果セットを返します。

検査結果　ULResultSet オブジェクト。ULResultSet は、SELECT 文で要求したデータです。クエリに関する詳細については、「ULResultSetSchema クラス」161 ページを参照してください。

ExecuteStatement メソッド

プロトタイプ　**ExecuteStatement( )** As Long
Member of **UltraLiteAFLib.ULPreparedStatement**

説明　文を実行します。

検査結果　更新されたローの数。

SetBooleanParameter メソッド

プロトタイプ　**SetBooleanParameter(**
param_number As Integer
param_value As Boolean
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明　パラメータを、渡された Boolean 値に設定します。

パラメータ

param_number 1 から始まるパラメータ番号を設定します。

param_value パラメータが受け取る値。

SetByteChunkParameter メソッド

プロトタイプ

SetByteChunkParameter(
param_number As Integer**,**
data As Long**,**
data_len As Long
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明

データを binary または long binary のカラムに設定します。

パラメータ

param_number 1 から始まるパラメータ番号を設定します。

data バイトの配列。

data_len 設定する配列からのバイトの数。SetByteChunk は、現在の内容を上書きします。既存の値を追加する方法については、「AppendByteChunkParameter メソッド」146 ページを参照してください。

SetByteParameter メソッド

プロトタイプ

SetByteParameter(
param_number As Integer
param_value As Byte
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明

パラメータを、渡された Byte 値に設定します。

パラメータ

param_number 1 から始まるパラメータ番号を設定します。

param_value パラメータが受け取る値。

SetDatetimeParameter メソッド

プロトタイプ

SetDatetimeParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明　パラメータを、渡された Datetime 値に設定します。

パラメータ　**param_number**　1 から始まるパラメータ番号を設定します。

param_value　パラメータが受け取る値。

SetDoubleParameter メソッド

プロトタイプ

SetDoubleParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明　パラメータを、渡された Double 値に設定します。

パラメータ　**param_number**　1 から始まるパラメータ番号を設定します。

param_value　パラメータが受け取る値。

SetIntegerParameter メソッド

プロトタイプ

SetIntegerParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明　パラメータを、渡された Integer 値に設定します。

パラメータ　**param_number**　1 から始まるパラメータ番号を設定します。

param_value　パラメータが受け取る値。

SetLongParameter メソッド

プロトタイプ

SetLongParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明

パラメータを、渡された Long 値に設定します。

パラメータ

param_number　1 から始まるパラメータ番号を設定します。

param_value　パラメータが受け取る値。

SetNullParameter メソッド

プロトタイプ

SetNullParameter(*param_id* As Integer **)**
Member of **UltraLiteAFLib.ULPreparedStatement**

説明

パラメータを NL に設定します。

パラメータ

parameter_id　1 から始まるパラメータ番号を設定します。

SetRealParameter メソッド

プロトタイプ

SetRealParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明

パラメータを、渡された Long 値に設定します。

パラメータ

param_number　1 から始まるパラメータ番号を設定します。

param_value　パラメータが受け取る値。

SetStringParameter メソッド

プロトタイプ

SetStringParameter(
param_number As Integer
param_value As String
)
Member of **UltraLiteAFLib.ULPreparedStatement**

説明

パラメータを、渡された文字列に設定します。

パラメータ

param_number　1 から始まるパラメータ番号を設定します。

param_value　パラメータが受け取る値。

ULPublicationSchema クラス

ULPublicationSchema オブジェクトを使用すると、パブリケーションの属性を取得できます。

プロパティ

プロトタイプ	説明
Mask As Long (読み込み専用)	パブリケーションのマスクを取得する。
Name As String (読み込み専用)	パブリケーションの名前を取得する。

ContainsTable メソッド

プロトタイプ

ContainsTable(*name* As String **)** As Boolean
Member of **UltraLiteAFLib.ULPublicationSchema**

説明

指定したテーブルがこのパブリケーションに含まれるかどうかを示します。

パラメータ

name ターゲット・テーブル名。

検査結果

True テーブルがパブリケーションに含まれる場合

False テーブルがパブリケーションに含まれない場合

ULResultSet クラス

ULResultSet オブジェクトは、SQL クエリが返したローを移動します。ULResultSet オブジェクトには、クエリが返したデータが含まれているので、INSERT、UPDATE、または DELETE のようなデータ操作言語の操作を行った後では、クエリの結果セットをリフレッシュする必要があります。このためには、ExecuteStatement を実行してから、ExecuteQuery を実行します。

プロパティ

プロトタイプ	説明
BOF As Boolean (読み込み専用)	現在のローの位置が最初のローの前かどうかを示す。現在のローの位置が最初のローの前なら True を返し、そうでなければ false を返す。
EOF As Boolean (読み込み専用)	現在のローの位置が最後のローの後かどうかを示す。最後のローの後であれば、EOF は true であり、そうでなければ false となる。
RowCount As Long (読み込み専用)	結果セット内のロー数。
Schema As ULResultSetSchema (読み込み専用)	この結果セットのスキーマの説明。

Close メソッド

プロトタイプ

Close()
Member of **UltraLiteAFLib.ULResultSet**

説明

このオブジェクトに関連付けられているリソースを解放します。

GetByteChunk メソッド

プロトタイプ

```
GetByteChunk ( _
    index As Integer, _
    src_offset As Long, _
    data As Long, _
    data_len As Long, _
    filled_len As Long _
) As Boolean
Member of UltraLiteAFLib.ULResultSet
```

説明

渡されたバッファ (配列) に、カラム内のバイナリ・データを入れます。BLOB に適しています。

パラメータ

index バイナリ・データを含むカラムの 1 から始まる序数。

offset 基本となるバイト配列へのオフセット。ソース・オフセットは、0 以上であることが必要です。それ以外の場合は、SQLE_INVALID_PARAMETER エラーが発生します。64K を超えるバッファも許されます。

data バイトの配列へのポインタ。バイトの配列へのポインタを取得するには、Visual Basic VarPtr() 関数を使用します。

data_len バッファ (配列) の長さ。data_len は 0 以上であることが必要です。

filled_len フェッチされたバイト数。BLOB データの長さは予め分からないので、BLOB データは通常、固定長のチャンクを使って、一度に 1 チャンクずつ、フェッチします。最後のチャンクは、チャンク・サイズよりも小さい場合があります。filled_len は、実際にフェッチされたバイト数をレポートします。

検査結果

読み込まれたバイト数。

エラー・セット

ulSQLE_CONVERSION_ERROR カラムのデータ型が BINARY でも LONG BINARY でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER カラムのデータ型が BINARY でオフセットが 0 でも 1 でもない、またはデータ長が 0 より小さい場合は、エラーが発生します。

カラムのデータ型が LONG BINARY で、オフセットが 1 より小さい場合も、エラーが発生します。

例

次の例で、edata はカラム名です。渡される *data_len* パラメータの長さが十分でない場合は、アプリケーション全体が終了します。

```
Dim data (512) As Byte
 ...
 table.Column("edata").GetByteChunk(0,data)
```

GetStringChunk メソッド

プロトタイプ

GetStringChunk(_
index As Integer**, _**
offset As Long**, _**
data As String**, _**
string_len As Long**, _**
filled_len As Long **_**
) As Boolean
Member of **UltraLiteAFLib.ULResultSet**

説明

渡された文字列に、カラム内のバイナリ・データを挿入します。Long Varchar に適しています。

パラメータ

index　ターゲット・カラムの 1 から始まるカラム ID。

offset　基本となるデータへの文字オフセット。ここから文字列の取得を開始します。

data　データ文字列。

string_len　返す文字列の長さ。

filled_len　挿入される文字列の長さ。

検査結果

BLOB データを binary または long binary のカラムから取得します。

エラー・セット

ulSQLE_CONVERSION_ERROR　カラムのデータ型が CHAR でも LONG VARCHAR でもない場合、エラーが発生します。

ulSQLE_INVALID_PARAMETER カラム・データ型が CHAR であり、src_offset が 64 K を超えている場合、エラーが発生します。

オフセットが 0 より小さいか、文字列の長さが 0 より小さい場合も、エラーが発生します。

MoveAfterLast メソッド

プロトタイプ **MoveAfterLast( )**
Member of **UltraLiteAFLib.ULResultSet**

説明 ULResultSet の最後のローの後に移動します。

MoveBeforeFirst メソッド

プロトタイプ **MoveBeforeFirst( )**
Member of **UltraLiteAFLib.ULResultSet**

説明 最初のローの前に移動します。

MoveFirst メソッド

プロトタイプ **MoveFirst( )** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

説明 最初のローに移動します。

検査結果 成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MoveLast メソッド

プロトタイプ **MoveLast( )** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

説明　最後のローに移動します。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MoveNext メソッド

プロトタイプ　**MoveNext( )** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

説明　次のローに移動します。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MovePrevious メソッド

プロトタイプ　**MovePrevious( )** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

説明　前のローに移動します。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

MoveRelative メソッド

プロトタイプ　**MoveRelative(** *index* As Long **)** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

説明 いくつかのローを、現在のローを基準にして相対的に移動します。結果セットでのカーソルの現在位置を基準にして、正のインデックス値は結果セット内を前に移動し、負のインデックス値は結果セット内を後ろに移動し、0 はカーソルを移動しません。

パラメータ **index** 移動するローの数。値は、正、負、または 0 です。

検査結果 成功の場合は **True** です。

失敗の場合は **False** です。たとえば、ローがない場合、メソッドは失敗します。

IsNull メソッド

プロトタイプ **IsNull(** *index* As Integer **)** As Boolean
Member of **UltraLiteAFLib.ULResultSet**

説明 このカラムに null 値が含まれているかどうかを示します。

パラメータ **index** カラムのインデックス値。

検査結果 値が Null の場合は true。

GetDatetime メソッド

プロトタイプ **GetDatetime(** *index* As Integer **)** As Date
Member of **UltraLiteAFLib.ULResultSet**

説明 カラム値を Date として取得します。

パラメータ **index** 結果セットで 1 から始まる序数を取得します。

検査結果 Date としての値。

GetDouble メソッド

プロトタイプ　**GetDouble(** *index* As Integer **)** As Double
Member of **UltraLiteAFLib.ULResultSet**

説明　カラム値を Double として取得します。

パラメータ　**index**　結果セットで 1 から始まる序数を取得します。

検査結果　Double としての値。

GetInteger メソッド

プロトタイプ　**GetInteger(** *index* As Integer **)** As Integer
Member of **UltraLiteAFLib.ULResultSet**

説明　カラム値を Integer として取得します。

パラメータ　**index**　結果セットで 1 から始まる序数を取得します。

検査結果　整数としての値。

GetLong メソッド

プロトタイプ　**GetLong(** *index* As Integer **)** As Long
Member of **UltraLiteAFLib.ULResultSet**

説明　カラム値を Long として取得します。

パラメータ　**index**　結果セットで 1 から始まる序数を取得します。

検査結果　Long としての値。

GetReal メソッド

プロトタイプ　**GetReal(** *index* As Integer **)** As Single
Member of **UltraLiteAFLib.ULResultSet**

説明　カラム値を Real として取得します。

パラメータ　**index**　結果セットで 1 から始まる序数を取得します。

検査結果　Real としての値。

GetString メソッド

プロトタイプ　**GetString(** *index* As Integer **)** As String
Member of **UltraLiteAFLib.ULResultSet**

説明　カラム値を String として取得します。

パラメータ　**index**　結果セットで 1 から始まる序数を取得します。

検査結果　String としての値。

ULResultSetSchema クラス

ULResultSetSchema は、結果セットのスキーマに関する情報を提供します。

プロパティ

プロトタイプ	説明
ColumnCount As Integer (読み込み専用)	結果セット内のカラム数を取得する。
ColumnName As String (読み込み専用)	結果セットに含まれるカラムの名前を取得する。
ColumnPrecision As Integer (読み込み専用)	カラムが数値の場合は、カラムのデータ型の精度を取得する。
ColumnScale As Integer (読み込み専用)	カラムが数値の場合は、カラムのデータ型の位取りを取得する。
ColumnSize As Integer (読み込み専用)	カラムのデータのサイズを取得する。
ColumnSQLType As ULSQLType (読み込み専用)	カラムの ULSQLType を取得する。

ULSchemaUpgradeState 列挙

ULSchemaUpgradeState 定数は、データベース・スキーマのアップグレード時のステータスを識別します。

定数	説明
ulUpgradeStateStarting	スキーマのアップグレードが開始された。 アップグレードをキャンセルできる唯一のステータスです。アップグレードがキャンセルされると、ステータス ulUpgradeStateAbort を持つ 2 番目のイベントを受け取ります。
ulUpgradeStateUpgrading	スキーマのアップグレードが進行中。
ulUpgradeStateAbort	スキーマのアップグレードはキャンセルされ、古いデータベースが保持されている。 回復可能なエラーまたはユーザ・アクションの結果、このステータスになることがあります。
ulUpgradeStateDone	スキーマのアップグレードが正常に完了した。
ulUpgradeStateError	重大なエラーが発生したため、データベースは使用できない。

参照

- 『Ultra Light データベース・ユーザーズ・ガイド』> 「Ultra Light データベース・スキーマのアップグレード」
- 『Ultra Light for MobileVB ユーザーズ・ガイド』> 「OnSchemaUpgradeProgress イベント」
- 『Ultra Light for MobileVB ユーザーズ・ガイド』> 「OnSchemaUpgradeStateChange イベント」

ULSQLCode 列挙

ULSQLCode 定数は、Ultra Light によってレポートされる SQL コードを示します。

エラーの説明については、『*Adaptive Server Anywhere エラー・メッセージ*』マニュアルを参照してください。

定数	値
ulSQLE_AGGREGATES_NOT_ALLOWED	-150
ulSQLE_ALIAS_NOT_UNIQUE	-830
ulSQLE_ALIAS_NOT_YET_DEFINED	-831
ulSQLE_BAD_ENCRYPTION_KEY	-840
ulSQLE_BAD_PARAM_INDEX	-689
ulSQLE_CANNOT_ACCESS_FILE	-602
ulSQLE_CANNOT_CHANGE_USER_NAME	-867
ulSQLE_CANNOT_MODIFY	-191
ulSQLE_CANNOT_EXECUTE_STMT	-111
ulSQLE_COLUMN_AMBIGUOUS	-144
ulSQLE_COLUMN_CANNOT_BE_NL	-195
ulSQLE_COLUMN_IN_INDEX	-127
ulSQLE_COLUMN_NOT_FOUND	-143
ulSQLE_COMMUNICATIONS_ERROR	-85
ulSQLE_CONNECTION_NOT_FOUND	-108
ulSQLE_CONVERSION_ERROR	-157
ulSQLE_CURSOROP_NOT_ALLOWED	-187
ulSQLE_CURSOR_ALREADY_OPEN	-172
ulSQLE_CURSOR_NOT_OPEN	-180

定数	値
ulSQLE_DATABASE_ERROR	-301
ulSQLE_DATABASE_NEW	123
ulSQLE_DATABASE_NOT_CREATED	-645
ulSQLE_DATABASE_NOT_FOUND	-83
ulSQLE_DATABASE_UPGRADE_FAILED	-672
ulSQLE_DATABASE_UPGRADE_NOT_POSSIBLE	-673
ulSQLE_DATATYPE_NOT_ALLOWED	-624
ulSQLE_DBSPACE_FL	-604
ulSQLE_DIV_ZERO_ERROR	-628
ulSQLE_DOWNLOAD_CONFLICT	-839
ulSQLE_DROP_DATABASE_FAILED	-651
ulSQLE_DYNAMIC_MEMORY_EXHAUSTED	-78
ulSQLE_ENGINE_ALREADY_RUNNING	-96
ulSQLE_ENGINE_NOT_MTIUSER	-89
ulSQLE_ERROR	-300
ulSQLE_ERROR_CALLING_FUNCTION	-622
ulSQLE_EXPRESSION_ERROR	-156
ulSQLE_IDENTIFIER_TOO_LONG	-250
ulSQLE_INDEX_NOT_FOUND	-183
ulSQLE_INDEX_NOT_UNIQUE	-196
ulSQLE_INTERRUPTED	-299
ulSQLE_INVALID_AGGREGATE_PLACEMENT	-862
ulSQLE_INVALID_FOREIGN_KEY	-194
ulSQLE_INVALID_FOREIGN_KEY_DEF	-113
ulSQLE_INVALID_GROUP_SELECT	-149

定数	値
ulSQLE_INVALID_LOGON	-103
ulSQLE_INVALID_OPTION_SETTING	-201
ulSQLE_INVALID_ORDER	-152
ulSQLE_INVALID_ORDERBY_COLUMN	-854
ulSQLE_INVALID_PARAMETER	-735
ulSQLE_INVALID_SQL_IDENTIFIER	-760
ulSQLE_INVALID_STATEMENT	-130
ulSQLE_LOCKED	-210,
ulSQLE_MEMORY_ERROR	-309
ulSQLE_METHOD_CANNOT_BE_CALLED	-669
ulSQLE_NAME_NOT_UNIQUE	-110
ulSQLE_NOERR	0
ulSQLE_NOTFOUND	100
ulSQLE_NOT_IMPLEMENTED	-134
ulSQLE_NO_CURRENT_ROW	-197
ulSQLE_NO_INDICATOR	-181
ulSQLE_OVERFLOW_ERROR	-158
ulSQLE_PERMISSION_DENIED	-121
ulSQLE_PRIMARY_KEY_NOT_UNIQUE	-193
ulSQLE_PRIMARY_KEY_VALUE_REF	-198
ulSQLE_PUBLICATION_NOT_FOUND	-280
ulSQLE_RESOURCE_GOVERNOR_EXCEEDED	-685
ulSQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE	130
ulSQLE_SERVER_SYNCHRONIZATION_ERROR	-857
ulSQLE_START_STOP_DATABASE_DENIED	-75

定数	値
ulSQLE_STATEMENT_ERROR	-132
ulSQLE_SYNTAX_ERROR	-131
ulSQLE_STRING_RIGHT_TRUNCATION	-638
ulSQLE_TABLE_HAS_PUBLICATIONS	-281
ulSQLE_TABLE_IN_USE	-214
ulSQLE_TABLE_NOT_FOUND	-141
ulSQLE_TOO_MANY_CONNECTIONS	-102
ulSQLE_TRALITE_OBJ_CLOSED	-908
ulSQLE_UNABLE_TO_CONNECT_OR_START	-764
ulSQLE_UNABLE_TO_START_DATABASE	-82
ulSQLE_UNCOMMITTED_TRANSACTIONS	-755
ulSQLE_UNKNOWN_FUNC	-148
ulSQLE_UNKNOWN_USERID	-140
ulSQLE_UNSUPPORTED_CHARACTER_SET_ERROR	-869
ulSQLE_UPLOAD_FAILED_AT_SERVER	-794
ulSQLE_WRONG_PARAMETER_COUNT	-154

ULSQLType 列挙

ULSQLType は、テーブル・カラムの型として使用されている、Ultra Light SQL データベースの型をリストします。

定数	Ultra Light データベースの型	値
ulTypeLong	Integer	0
ulTypeUnsignedLong	SmallInt	2
ulTypeShort	UnsignedInteger	1
ulTypeUnsignedShort	UnsignedSmallInt	3
ulTypeBig	Big	4
ulTypeUnsignedBig	UnsignedBig	5
ulTypeByte	Byte	6
ulTypeBit	Bit	7
ulTypeDateTime	Time	8
ulTypeDate	Date	9
ulTypeTime	Timestamp	10
ulTypeDouble	Double	11
ulTypeReal	Real	12
ulTypeNumeric	(Var)Binary	17
ulTypeBinary	LongBinary	13
ulTypeString	(Var)Char	15
ulTypeLongString	LongVarchar	16
ulTypeLongBinary	Numeric	14
ulTypeUUID	UniqueIdentifier	18

ULStreamErrorCode 列挙

ULStreamErrorCode 定数は、同期時の通信エラーを識別します。

これらのエラーの詳細については、『ASA エラー・メッセージ』>「Mobile Link 通信エラー・メッセージ」を参照してください。

定数	値
ulStreamErrorCodeNone	0
ulStreamErrorCodeParameter	1
ulStreamErrorCodeParameterNotUint32	2
ulStreamErrorCodeParameterNotUint32Range	3
ulStreamErrorCodeParameterNotBoolean	4
ulStreamErrorCodeParameterNotHex	5
ulStreamErrorCodeMemoryAllocation	6
ulStreamErrorCodeParse	7
ulStreamErrorCodeRead	8
ulStreamErrorCodeWrite	9
ulStreamErrorCodeEndWrite	10
ulStreamErrorCodeEndRead	11
ulStreamErrorCodeNotImplemented	12
ulStreamErrorCodeWouldBlock	13
ulStreamErrorCodeGenerateRandom	14
ulStreamErrorCodeInitRandom	15
ulStreamErrorCodeSeedRandom	16
ulStreamErrorCodeCreateRandomObject	17
ulStreamErrorCodeShuttingDown	18

定数	値
ulStreamErrorCodeDequeuingConnection	19
ulStreamErrorCodeSecureCertificateRoot	20
ulStreamErrorCodeSecureCertificateCompanyName	21
ulStreamErrorCodeSecureCertificateChainLength	22
ulStreamErrorCodeSecureCertificateRef	23
ulStreamErrorCodeSecureCertificateNotTrusted	24
ulStreamErrorCodeSecureDuplicateContext	25
ulStreamErrorCodeSecureSetIo	26
ulStreamErrorCodeSecureSetIoSemantics	27
ulStreamErrorCodeSecureCertificateChainFunc	28
ulStreamErrorCodeSecureCertificateChainRef	29
ulStreamErrorCodeSecureEnableNonBlocking	30
ulStreamErrorCodeSecureSetCipherSuites	31
ulStreamErrorCodeSecureSetChainNumber	32
ulStreamErrorCodeSecureCertificateFileNotFound	33
ulStreamErrorCodeSecureReadCertificate	34
ulStreamErrorCodeSecureReadPrivateKey	35
ulStreamErrorCodeSecureSetPrivateKey	36
ulStreamErrorCodeSecureCertificateExpiryDate	37
ulStreamErrorCodeSecureExportCertificate	38
ulStreamErrorCodeSecureAddCertificate	39
ulStreamErrorCodeSecureTrustedCertificateFileNotFound	40
ulStreamErrorCodeSecureTrustedCertificateRead	41
ulStreamErrorCodeSecureCertificateCount	42

定数	値
ulStreamErrorCodeSecureCreateCertificate	43
ulStreamErrorCodeSecureImportCertificate	44
ulStreamErrorCodeSecureSetRandomRef	45
ulStreamErrorCodeSecureSetRandomFunc	46
ulStreamErrorCodeSecureSetProtocolSide	47
ulStreamErrorCodeSecureAddTrustedCertificate	48
ulStreamErrorCodeSecureCreatePrivateKeyObject	49
ulStreamErrorCodeSecureCertificateExpired	50
ulStreamErrorCodeSecureCertificateCompanyUnit	51
ulStreamErrorCodeSecureCertificateCommonName	52
ulStreamErrorCodeSecureHandshake	53
ulStreamErrorCodeHttpVersion	54
ulStreamErrorCodeSecureSetReadFunc	55
ulStreamErrorCodeSecureSetWriteFunc	56
ulStreamErrorCodeSocketHostNameNotFound	57
ulStreamErrorCodeSocketGetHostByAddr	58
ulStreamErrorCodeSocketLocalhostNameNotFound	59
ulStreamErrorCodeSocketCreateTcpip	60
ulStreamErrorCodeSocketCreateUdp	61
ulStreamErrorCodeSocketBind	62
ulStreamErrorCodeSocketCleanup	63
ulStreamErrorCodeSocketClose	64
ulStreamErrorCodeSocketConnect	65
ulStreamErrorCodeSocketGetName	66

定数	値
ulStreamErrorCodeSocketGetOption	67
ulStreamErrorCodeSocketSetOption	68
ulStreamErrorCodeSocketListen	69
ulStreamErrorCodeSocketShutdown	70
ulStreamErrorCodeSocketSelect	71
ulStreamErrorCodeSocketStartup	72
ulStreamErrorCodeSocketPortOutOfRange	73
ulStreamErrorCodeLoadNetworkLibrary	74
ulStreamErrorCodeActsyncNoPort	75
ulStreamErrorCodeHttpExpectedPost	89

ULStreamErrorContext 列挙

ULStreamErrorContext 定数は、ULStreamErrorContext の指定に使用できる定数を識別します。ULStreamErrorContext は、ストリーム・エラーが発生したときに実行されるネットワーク・オペレーションです。

定数	値
ulStreamErrorContextUnknown	0
ulStreamErrorContextRegister	1
ulStreamErrorContextUnregister	2
ulStreamErrorContextCreate	3
ulStreamErrorContextDestroy	4
ulStreamErrorContextOpen	5
ulStreamErrorContextClose	6
ulStreamErrorContextRead	7
ulStreamErrorContextWrite	8
ulStreamErrorContextWriteFlush	9
ulStreamErrorContextEndWrite	10
ulStreamErrorContextEndRead	11
ulStreamErrorContextYield	12
ulStreamErrorContextSoftshutdown	13

ULStreamErrorID 列挙

ULStreamErrorID は、同期が失敗したときにおそらくエラーの原因となった、ネットワーク・レイヤの列挙です。

定数	値
ulStreamErrorIDTcpip	0
ulStreamErrorIDSerial	1
ulStreamErrorIDFake	2
ulStreamErrorIDPalmConduit	3
ulStreamErrorIDPalmSs	4
ulStreamErrorIDNettech	5
ulStreamErrorIDRimbb	6
ulStreamErrorIDHttp	7
ulStreamErrorIDHttps	8
ulStreamErrorIDDhCast	9
ulStreamErrorIDSecure	10
ulStreamErrorIDCerticom	11
ulStreamErrorIDJavaCerticom	12
ulStreamErrorIDCerticomSsl	13
ulStreamErrorIDCerticomTls	14
ulStreamErrorIDWirestrm	15
ulStreamErrorIDWireless	16
ulStreamErrorIDReplay	17
ulStreamErrorIDStrm	18
ulStreamErrorIDUdp	19

定数	値
ulStreamErrorIDEmail	20
ulStreamErrorIDFile	21
ulStreamErrorIDActivesync	22
ulStreamErrorIDRsaTls	23
ulStreamErrorIDJavaRsa	24

ULStreamType 列挙

ULStreamType 定数は、ストリーム・タイプの指定に使用できる定数を識別します。これらの定数は、同期に使用できる、Mobile Link 同期ストリームのタイプを表します。

定数	値	説明
ulUnknown	0	ストリーム・タイプは設定されていない。ストリーム・タイプを設定してから同期を行ってください。
ulTCPIP	1	TCP/IP ストリーム
ulHTTP	2	HTTP ストリーム
ulHTTPS	3	HTTPS 同期
ulPalmConduit	4	HotSync 同期用

ULSyncParms クラス

ULSyncParms オブジェクトの属性セットは、データベースと統合データベースまたはデスクトップ・データベースとの同期方法を決定します。読み込み専用の属性は、最後の同期ステータスを反映します。

プロパティ

ULSyncParms のプロパティは次のとおりです。

プロトタイプ	説明
CheckpointStore As Boolean	true の場合は、同期中にデータベースのチェックポイントを追加して、同期処理中にデータベースが大きくなりすぎないように制限する。これは、多くの更新を伴う大量のダウンロードに最適です。 『Mobile Link クライアント』>「Checkpoint Store 同期パラメータ」を参照してください。
DownloadOnly As Boolean	同期はデータのダウンロードのみを行うかどうかを示す。 『Mobile Link クライアント』>「Download Only 同期パラメータ」を参照してください。
KeepPartialDownload As Boolean	ダウンロード時の通信エラーが原因で同期が失敗した場合、すべての変更をロールバックしないで、正常にダウンロードされた変更を適用する。 『Mobile Link クライアント』>「Keep Partial Download 同期パラメータ」を参照してください。

プロトタイプ	説明
NewPassword As String	次の同期で、この新しいパスワードにユーザ・パスワードを変更する。 『Mobile Link クライアント』>「New Password 同期パラメータ」を参照してください。
Password As String	特定のユーザ名に対応したパスワード。 『Mobile Link クライアント』>「Password 同期パラメータ」を参照してください。
PingOnly As Boolean	true の場合は、サーバの活性をチェックするが、データを同期しない。 『Mobile Link クライアント』>「Ping 同期パラメータ」を参照してください。
PublicationMask As Long	同期させるパブリケーションを指定する。デフォルトでは、すべてのデータを同期する。 『Mobile Link クライアント』>「publication 同期パラメータ」を参照してください。
ResumePartialDownload As Boolean	同期が失敗したときにダウンロードされる予定だった変更のみを適用し、ダウンロード時の通信エラーが原因で失敗した同期を再開する。 『Mobile Link クライアント』>「Resume Partial Download 同期パラメータ」を参照してください。

プロトタイプ	説明
SendColumnNames As Boolean	SendColumnNames が true の場合、Mobile Link 同期サーバにカラム名を送信する。自動スクリプト生成を行うためには、Mobile Link 同期サーバにカラム名を送信する必要がある。 『Mobile Link クライアント』>「Send Column Names 同期パラメータ」を参照してください。
SendDownloadAck As Boolean	SendDownloadAck が true の場合、同期中にダウンロード確認を送信する。 『Mobile Link クライアント』>「Send Download Acknowledgement 同期パラメータ」を参照してください。
Stream As ULStreamType constants	同期中に使用するストリームのタイプを設定する。 『Mobile Link クライアント』>「Stream Type 同期パラメータ」を参照してください。
StreamParms As String	特定のストリーム・タイプのネットワーク・プロトコル・オプションを設定する。 『Mobile Link クライアント』>「Stream Parameters 同期パラメータ」と『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。
UploadOnly As Boolean	同期はデータのアップロードのみを行うかどうかを示す。 『Mobile Link クライアント』>「Upload Only 同期パラメータ」を参照してください。

プロトタイプ	説明
UserName As String	同期用の Mobile Link ユーザ名。 『Mobile Link クライアント』> 「User Name 同期パラメータ」を参照してください。
Version As String	実行する同期スクリプト・バージョン。 『Mobile Link クライアント』> 「Version 同期パラメータ」を参照してください。

例　次の例は、Ultra Light for MobileVB アプリケーションの同期パラメータを設定します。

```
With Connection.SyncParms
     .UserName = "afsample"
     .Stream = ULStreamType.ulTCPIP
     .Version = "ul_default"
     .SendColumnNames = True
 End With
 Connection.Synchronize
```

AddAuthenticationParm メソッド

プロトタイプ　**AddAuthenticationParm(** BSTR parm **)**
Member of **UltraLiteAFLib.ULSyncParms**

説明　authenticate_parms Mobile Link 同期スクリプトに渡すパラメータを追加します。

パラメータ　**parm**　追加するパラメータ。

検査結果　戻り値なし。

参照　『Mobile Link クライアント』> 「Authentication Parameters 同期パラメータ」

『Mobile Link 管理ガイド』> 「authenticate_parameters 接続イベント」

ClearAuthenticationParms メソッド

プロトタイプ

ClearAuthenticationParms( )
Member of **UltraLiteAFLib.ULSyncParms**

説明

authenticate_parms Mobile Link 同期スクリプトに渡すパラメータをすべてクリアします。

検査結果

戻り値なし。

参照

『Mobile Link クライアント』> 「Authentication Parameters 同期パラメータ」

『Mobile Link 管理ガイド』> 「authenticate_parameters 接続イベント」

ULSyncResult クラス

ULSyncResult オブジェクトの属性は、最後の同期の結果を格納します。

プロパティ

次は、ULSyncResult のプロパティです。

プロトタイプ	説明
AuthStatus As ULAuthStatusCode (読み込み専用)	前回行われた同期の認証ステータス・コードを取得する。 『Mobile Link クライアント』> 「Authentication Status 同期パラメータ」を参照してください。
PartialDownloadRetained (読み込み専用)	ダウンロード時に同期が失敗し、部分的にダウンロードが保持されていることを示す。 『Mobile Link クライアント』> 「Partial Download Retained 同期パラメータ」を参照してください。
IgnoredRows As Boolean (読み込み専用)	前回行われた同期でローが無視されたかどうかを示す。 『Mobile Link クライアント』> 「Ignored Rows 同期パラメータ」を参照してください。
StreamErrorCode As ULStreamErrorCode (読み込み専用)	同期ストリームによってレポートされるエラー・コードを取得する。
StreamErrorContext As ULStreamErrorContext (読み込み専用)	実行される基本的なネットワーク・オペレーションを取得する。
StreamErrorID As ULStreamErrorID (読み込み専用)	エラーをレポートするネットワーク・レイヤを取得する。

プロトタイプ	説明
StreamErrorSystem As Long (読み込み専用)	ストリーム・エラー・システム固有のコードを取得する。
UploadOK As Boolean (読み込み専用)	前回行われた同期でデータが正常にアップロードされたかどうかを示す。 『Mobile Link クライアント』> 「Version 同期パラメータ」を参照してください。

ULSyncState 列挙

定数	説明
ulSyncStateStarting	同期処理はまだ行われていない。
ulSyncStateConnecting	同期ストリームは構築されたが、まだ開かれていない。
ulSyncStateSendingHeader	同期ストリームが開かれ、ヘッダが送信されようとしている。
ulSyncStateSendingTable	テーブルの送信中。
ulSyncStateSendingData	現在のテーブルのデータの送信中。
ulSyncStateFinishingUpload	アップロードが完了。送信された最終的なロー数が、このイベントに含まれます。
ulSyncStateReceivingUploadAck	アップロード完了の確認の受信中。
ulSyncStateReceivingTable	テーブルの受信中。
ulSyncStateReceivingData	現在のテーブルのデータの受信中。
ulSyncStateCommittingDownload	ダウンロードのコミット中。受信された最終的なロー数が、このイベントに含まれます。
ulSyncStateSendingDownloadAck	ダウンロード完了の確認の送信中。
ulSyncStateDisconnecting	同期ストリームが閉じられようとしている。
ulSyncStateDone	同期が正常に完了した。SyncResult オブジェクトが更新されました。
ulSyncStateError	同期は完了したが、エラーが発生した。SyncResult と QLCode の詳細を確認してください。

定数	説明
ulSyncStateRollingBackDownload	ダウンロード中にエラーが発生したため、同期によってダウンロードがロールバックされている。後続の ulSyncStateError 進行状況レポートにエラーがレポートされます。
ulSyncStateCancelled	同期がキャンセルされた。

ULTable クラス

ULTable クラスは、テーブルのデータの格納、削除、更新、読み込みに使用します。

Open メソッドを呼び出さないと、テーブルのデータを操作できません。ULTable ではテーブル・モードを使用してテーブルを操作します。

モード	説明
FindBegin	検索モードを開始
InsertBegin	挿入モードを開始
LookupBegin	ルックアップ・モードを開始
UpdateBegin	更新モードを開始

プロパティ

プロトタイプ	説明
BOF As Boolean (読み込み専用)	現在のローの位置が最初のローの前かどうかを示す。現在のローの位置が最初のローの前なら True を返し、そうでなければ false を返す。
EOF As Boolean (読み込み専用)	現在のローの位置が最後のローの後かどうかを示す。現在のローの位置が最初のローの前なら True を返し、そうでなければ false を返す。
IsOpen As Boolean (読み込み専用)	テーブルが現在開いているかどうかを示す。
RowCount As Long (読み込み専用)	テーブルのローの数を取得する。
Schema As ULTableSchema (読み込み専用)	テーブル・スキーマに関する情報を取得する。

Close メソッド

プロトタイプ

Close( )
Member of **UltraLiteAFLib.ULTable**

説明

テーブルに関連付けられているリソースを解放します。このメソッドは、テーブルに関するすべての処理が完了した後で呼び出してください。

Palm OS では、テーブルを閉じていない場合は、現在位置で再び開くことができます。

Column メソッド

Column(*name* As String **)** As ULColumn
Member of **UltraLiteAFLib.ULTable**

説明

指定したカラム名のオブジェクトを返します。

ULColumn オブジェクトについては、「ULColumn クラス」103 ページを参照してください。

パラメータ

name 返されるカラムの名前。

検査結果

Columns オブジェクトを返します。

Delete メソッド

プロトタイプ

Delete( )
Member of **UltraLiteAFLib.ULTable**

説明

現在のローをテーブルから削除します。

DeleteAllRows メソッド

プロトタイプ

DeleteAllRows( )
Member of **UltraLiteAFLib.ULTable**

説明

テーブルのすべてのローを削除します。

アプリケーションによっては、テーブル内のローをすべて削除してから、新しいデータ・セットをテーブルにダウンロードするほうが便利なことがあります。**ULConnection.StopSynchronizationDelete** メソッドを使用するか、**DeleteAllRows** の代わりに **Truncate** を呼び出すかすれば、ローは、統合データベースから削除しなくても、Ultra Light データベースから削除できます。

FindBegin メソッド

プロトタイプ

FindBegin( )
Member of **UltraLiteAFLib.ULTable**

説明

検索のためにテーブルを準備します。

FindFirst メソッド

プロトタイプ

FindFirst([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明

テーブルを先頭から順方向に移動しながら、現在のインデックスの値かそのセットに完全に一致するローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。このソート順は、アプリケーションが Open メソッドを呼び出したときに指定されます。デフォルトのインデックスはプライマリ・キーです。

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最後のローの後ろ (**EOF**) になります。

注意 : FindBegin を呼び出してから、このメソッドを使用してください。

パラメータ

num_columns　FindFirst で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindFirst に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindFirst で使用されます。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。

FindLast メソッド

プロトタイプ

FindLast([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明

テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセットに一致する最初のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されます。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」195 ページを参照してください。

検索する値を指定するには、値を検索するインデックスの各カラムに値を設定します。カーソルは、インデックスの値と完全に一致した最後のローで停止します。失敗すると、カーソル位置は最初のローの前 (**BOF**) になります。

注意
FindBegin を呼び出してから、このメソッドを使用してください。

パラメータ

num_columns　FindLast で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindLast に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindLast で使用されます。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。

FindNext メソッド

プロトタイプ

FindNext([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明

現在の位置からテーブルを順方向に移動しながら、現在のインデックスの値かそのセットに完全に一致する次のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」195 ページを参照してください。

カーソルは、インデックスの値と完全に一致した最初のローで停止します。失敗すると、カーソル位置は最後のローの後ろ **(EOF)** になります。

注意 : FindFirst または FindLast を呼び出してから、このメソッドを使用してください。

パラメータ

num_columns　FindNext で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindNext に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindNext で使用されます。

検査結果

成功の場合は **True** です。

失敗の場合 (EOF) は **False** です。

FindPrevious メソッド

プロトタイプ

FindPrevious([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明

現在の位置からテーブルを逆方向に移動しながら、現在のインデックスの値かそのセットに完全に一致する前のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」195 ページを参照してください。

失敗すると、カーソル位置は最初のローの前 (**BOF**) になります。

パラメータ

num_columns　FindPrevious で使用するカラム数を参照するオプションのパラメータ。たとえば、2 が渡されると、最初の 2 つのカラムが FindPrevious に使用されます。num_columns が、インデックスされたカラムの数を超えると、すべてのカラムが FindPrevious で使用されます。

検査結果

成功の場合は **True** です。

失敗の場合 (BOF) は **False** です。

Insert メソッド

プロトタイプ

Insert( ) As Boolean
Member of **UltraLiteAFLib.ULTable**

説明

直前に実行された **Set** メソッドで指定された値を使って、テーブルにローを挿入します。**InsertBegin** を呼び出してから、このメソッドを使用してください。各 ULColumn オブジェクトに対して設定します。

検査結果

成功の場合は **True** です。

失敗の場合 (BOF) は **False** です。

InsertBegin メソッド

プロトタイプ

InsertBegin( )
Member of **UltraLiteAFLib.ULTable**

説明　カラム値をデフォルトに設定し、新しいローを挿入できるようにテーブルを準備します。

例　次の例では、InsertBegin は挿入モードに設定して、CustomerTable カラムへのデータ値の代入を開始できるようにします。

```
CustomerTable.InsertBegin
 CustomerTable.Column("Fname").StringValue = fname
 CustomerTable.Column("Lname").StringValue = lname
 CustomerTable.Insert
```

参照　「UpdateBegin メソッド」196 ページ

LookupBackward メソッド

プロトタイプ　**LookupBackward(** [*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明　テーブルを最後から逆方向に移動しながら、現在のインデックスの値またはそのセットに一致する最初のローか、それより少ない値の最初のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」195 ページを参照してください。

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致するか、それより少ない値の最後のローで停止します。検索が失敗した場合 (つまり、検索する値より小さい値のローがない場合)、カーソル位置は最初のローの前 **(BOF)** になります。

パラメータ　**num_columns**　複合インデックスのための、ルックアップで使用するカラムの数。

検査結果　成功の場合は **True** です。

失敗の場合は **False** です。

LookupBegin メソッド

プロトタイプ

LookupBegin( )
Member of **UltraLiteAFLib.ULTable**

説明

ルックアップのためにテーブルを準備します。

LookupForward メソッド

プロトタイプ

LookupForward([*num_columns* As Long = 32767] **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明

テーブルを最初から順方向に移動しながら、現在のインデックスの値またはそのセットに一致するか、その値より大きい値を持つ最初のローを検索します。

現在のインデックスは、テーブルのソート順の指定に使用されているインデックスです。アプリケーションによって **Open** メソッドを呼び出すときに指定されます。デフォルトのインデックスはプライマリ・キーです。

詳細については、「Open メソッド」195 ページを参照してください。

検索する値を指定するには、インデックスの各カラムに値を設定します。カーソルは、インデックスの値に一致するか、それより大きい値の最初のローで停止します。検索が失敗した場合 (つまり、検索する値より大きい値のローがない場合)、カーソル位置は最後のローの後ろ **(EOF)** になります。

パラメータ

num_columns　複合インデックスのための、ルックアップで使用するカラムの数。

検査結果

成功の場合は **True** です。

失敗の場合は **False** です。

MoveAfterLast メソッド

プロトタイプ　**MoveAfterLast( )** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明　最後のローの後に移動します。

検査結果　成功の場合は **True** です。

処理が失敗した場合は **False** です。

MoveBeforeFirst メソッド

プロトタイプ　**MoveBeforeFirst( )** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明　最初のローの前に移動します。

検査結果　成功の場合は **True** です。

処理が失敗した場合は **False** です。

MoveFirst メソッド

プロトタイプ　**MoveFirst( )** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明　最初のローに移動します。

検査結果　成功の場合は **True** です。

テーブル内にデータがない場合は **False** です。

MoveLast メソッド

プロトタイプ　**MoveLast( )** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明　最後のローに移動します。

検査結果　成功の場合は **True** です。

テーブル内にデータがない場合は **False** です。

MoveNext メソッド

プロトタイプ　**MoveNext( )** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明　次のローに移動します。

検査結果　成功の場合は **True** です。

テーブル内にデータがない場合は **False** です。たとえば、ローがこれ以上ない場合 MoveNext は失敗します。

MovePrevious メソッド

プロトタイプ　**MovePrevious( )** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明　前のローに移動します。

検査結果　成功の場合は **True** です。

テーブル内にデータがない場合は **False** です。たとえば、ローがこれ以上ない場合 MovePrevious は失敗します。

MoveRelative メソッド

プロトタイプ　**MoveRelative(** *index* As Long **)** As Boolean
Member of **UltraLiteAFLib.ULTable**

説明　いくつかのローを、現在のローを基準にして相対的に移動します。

パラメータ　**index**　移動するローの数。値は、正、負、または 0 です。ロー・バッファを再投入する場合は、0 が便利です。

検査結果　成功の場合は **True** です。

処理が失敗した場合、たとえばカーソルが最初のローの前、または最後のローの後ろにある場合は **False** です。

Open メソッド

プロトタイプ

```
Open(
    [ index_name As String ], _
    [ persistent_name As String ] _
)
Member of UltraLiteAFLib.ULTable
```

説明　テーブルを開き、読み込みまたは操作ができるようにします。デフォルトでは、ローはプライマリ・キーによって順序付けられます。インデックスの名前を指定すると、ローを別の方法で並べ替えることができます。

カーソルは、テーブル内にある最初のローの前に置かれます。

パラメータ　**index_name**　インデックスの名前を参照するオプションのパラメータ。

persistent_name　Palm Computing Platform アプリケーションの場合、格納されたテーブル名を参照するオプションのパラメータ。

Truncate メソッド

プロトタイプ

```
Truncate( )
    Member of UltraLiteAFLib.ULTable
```

説明　このテーブルからデータをすべて削除します。この変更の同期は行われないため、同期時に統合データベース内のデータには影響しません。

詳細については、「StartSynchronizationDelete メソッド」122 ページを参照してください。

Update メソッド

プロトタイプ

Update( )
Member of **UltraLiteAFLib.ULTable**

説明

ULColumn メソッドで指定された値を使って、テーブルのローを更新します。

注意 : 先に UpdateBegin を呼び出してください。

UpdateBegin メソッド

プロトタイプ

UpdateBegin( )
Member of **UltraLiteAFLib.ULTable**

説明

現在のローの内容を変更するためにテーブルを準備します。

例

```
CustomerTable.UpdateBegin
 CustomerTable.Column("Fname").StringValue = fname
 '...
 CustomerTable.Update
```

ULTableSchema クラス

ULTableSchema オブジェクトを使用すると、テーブルの属性を取得できます。

プロパティ

ULTableSchema は、テーブルに関するメタデータを表します。ULTableSchema クラスのプロパティは次のとおりです。

プロトタイプ	説明
ColumnCount As Integer (読み込み専用)	このテーブル内のカラム数。
IndexCount As Integer (読み込み専用)	このテーブル内のインデックス数。
Name As String (読み込み専用)	このテーブルの名前。
NeverSynchronized As Boolean (読み込み専用)	テーブルが同期から常に除外されるかどうかを示す。
PrimaryKey As ULIndexSchema (読み込み専用)	このテーブルのプライマリ・キー。
UploadUnchangedRows As Boolean (読み込み専用)	前回の同期以降に変更されたローだけでなく、同期に対してテーブルのすべてのローをアップロードするかどうかを示す。

GetColumnName メソッド

プロトタイプ　**GetColumnName(** *id* As Integer **)** As String
Member of **UltraLiteAFLib.ULTableSchema**

説明　指定した *id* 値に対応するカラムの名前を返します。ColumnCount プロパティは、テーブルのカラム数を返します。各カラムは、1 ～ ColumnCount 値までのユニークな番号を持ちます。1 はテーブル内の最初のカラムであり、2 はテーブル内の 2 番目のカラムです。以下も同様に続きます。

パラメータ　**id**　カラムの ID。

検査結果　カラムの名前。

GetIndex メソッド

プロトタイプ

GetIndex(*name* As String **)** As ULIndexSchema
Member of **UltraLiteAFLib.ULTableSchema**

説明　指定のインデックスの ULIndexSchema オブジェクトを返します。

ULIndexSchema オブジェクトについては、「ULIndexSchema クラス」143 ページを参照してください。

パラメータ　**name**　インデックス名。

検査結果　テーブルの指定されたインデックスのスキーマ・オブジェクトを返します。

GetIndexName メソッド

プロトタイプ

GetIndexName(*id* As Integer **)** As String
Member of **UltraLiteAFLib.ULTableSchema**

説明　指定した *id* 値に対応する、テーブル内のインデックス名を返します。IndexCount プロパティは、テーブル内のインデックス数を返します。各インデックスは、1 ～ IndexCount 値までのユニークな番号を持ちます。1 はテーブル内の最初のインデックスであり、2 はテーブル内の 2 番目のインデックスです。以下も同様に続きます。

パラメータ　**name**　インデックスの id。

検査結果　インデックスの名前を返します。

InPublication メソッド

プロトタイプ　**InPublication(** *publicationName* As String **)** As Boolean
Member of **UltraLiteAFLib.ULTableSchema**

説明　このテーブルが、指定されたパブリケーションの一部であるかどうかを示します。

パラメータ　**publicationName**　確認するパブリケーションの名前。

検査結果　**True** テーブルがパブリケーションの一部である場合

False テーブルがパブリケーションの一部でない場合

索引

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V

W

あ

い

え

そ

た

ち

て

と

な

に

ね

は

ひ

ふ

ま

も

ゆ

ら

る

ろ