Ultra Light® データベース管理とリファレンス

バージョン 16.0

2013 年 2 月

Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

バージョン 16.0
2013 年 2 月

© 2013 SAP AG or an SAP affiliate company.All rights reserved.

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

SAP AG およびディストリビュータが販売しているソフトウェア製品には、他のソフトウェアベンダー独自のソフトウェアコンポーネントが含まれているものがあります。国内製品の仕様は変わることがあります。

これらの資料は SAP AG および関連会社 (SAP グループ) が情報のみを目的として提供するものであり、いかなる種類の表明または保証も行うものではなく、SAP グループはこの資料に関する誤りまたは脱落について責任を負わないものとします。SAP グループの製品およびサービスに関する保証は、かかる製品およびサービスに付属している明確な保証文書がある場合、そこで明記されている保証に限定されます。ここに記載されているいかなる内容も、追加保証を構成するものとして解釈されるものではありません。

ここに記載された SAP および他の SAP 製品とサービス、ならびに対応するロゴは、ドイツおよび他の国における SAP AG の商標または登録商標です。http://www.sap.com/corporate-en/legal/copyright/index.epx#trademark を参照してください。

目次

はじめに vii

Ultra Light 概要 1

Ultra Light のアーキテクチャ 1
Ultra Light 同期クライアント 2
Ultra Light のサポート対象プラットフォーム 3
Ultra Light、Ultra Light Java Edition、SQL Anywhere の機能比較 3
Ultra Light および Ultra Light Java Edition データベースの制限事項 9
CustDB サンプルアプリケーションの概要 14
Windows Mobile 用の Ultra Light ソリューションの考慮事項 17

Ultra Light データベースの作成方法 21

データベース作成ウィザードでの Ultra Light データベースの作成 21
コマンドプロンプトでの Ultra Light データベースの作成 22
Mobile Link 同期モデルでの Ultra Light データベースの作成 23
リモートデータベースの集中管理による Ultra Light データベースの作成 23
XML ファイルからの Ultra Light データベースの作成 23
初回接続時の Ultra Light データベースの作成 24
Ultra Light データベース作成パラメータ 25
Ultra Light 文字セット 26
データベースセキュリティ 29

SQL Anywhere データベースから Ultra Light データベースへの変換 33

Ultra Light データベースへの接続 35

Ultra Light 接続文字列とパラメータ 35
Ultra Light 接続パラメータと ULSQLCONNECT 環境変数 37
接続パラメータでの Ultra Light ファイルパスの指定 37

Ultra Light データベースのタスクと機能 39

データベースプロパティの読み取り 39
データベースオプションへのアクセス 39
Ultra Light のイベント通知 40
独立性レベル 43
Ultra Light データベースの検証 45
Ultra Light および Ultra Light Java Edition データベースのバックアップとリカバリ 46

Ultra Light データベースのスキーマ 49

Ultra Light のテーブルとカラム 50
Ultra Light インデックス 56
Ultra Light ユーザ 61

Mobile Link クライアントとしての Ultra Light 69

Ultra Light クライアント 69
Windows Mobile での Ultra Light と ActiveSync 87
Ultra Light の同期パラメータ 89
Ultra Light のネットワークプロトコルのオプション 111

Ultra Light の配備 113

Ultra Light アプリケーションのビルドと配備の仕様 113
Ultra Light および Ultra Light Java Edition データベースの配備に関する制限事項 121
Ultra Light データベーススキーマのアップグレードの配備 122
Ultra Light エンジンの起動 124
Ultra Light の ActiveSync プロバイダの配備 125
ActiveSync Manager を使用したアプリケーションの登録 127

チュートリアル：Ultra Light の CustDB サンプルアプリケーションの構築 129

レッスン 1：CustDB アプリケーションの構築と実行 129
レッスン 2：Mobile Link サーバの起動と初期同期の実行 130

レッスン 3 : Ultra Light データベース内のデータの更新 131
レッスン 4 : Ultra Light データベースと統合データベースの同期 132
レッスン 5 : Mobile Link 同期スクリプトのブラウズ 134

Ultra Light データベースのリファレンス 137

Ultra Light 作成パラメータ 137
Ultra Light 接続パラメータ 164
Ultra Light データベースプロパティ 187
Ultra Light データベースオプション 191
Ultra Light ユーティリティ 196
Ultra Light のシステムテーブル 237
Ultra Light Java Edition データベースのプロパティ 243
Ultra Light Java Edition データベースのオプション 245
Ultra Light Java Edition のユーティリティ 251
Ultra Light Java Edition システムテーブル 262

Ultra Light SQL リファレンス 269

Ultra Light SQL 言語の要素 269
Ultra Light SQL データ型 293
Ultra Light SQL 関数 313
Ultra Light SQL 文 416

Ultra Light のパフォーマンスに関するヒント 461

Ultra Light データベースのキャッシュサイズの調整 461
Ultra Light Java Edition データベースのキャッシュサイズ 462
クエリのパフォーマンスに関するヒント 463
挿入と更新のパフォーマンスのヒント 475
Ultra Light のベンチマークに関するヒント 479

Ultra Light トラブルシューティング 485

Ultra Light エンジンを起動できない 485
アップグレード後にデータベースに接続できない 485
Ultra Light データベースの破損 486

データベースのサイズが安定しない 487
新しい Ultra Light データベースへの ASCII データのインポート 488
以前のバージョンのユーティリティが動作する 488
結果セットに予測できない変更が生じる 489
Ultra Light エンジンクライアントでエラー -764 が発生する 489

索引 491

はじめに

このマニュアルでは、小型デバイス用 Ultra Light データベースシステムの概要を説明します。

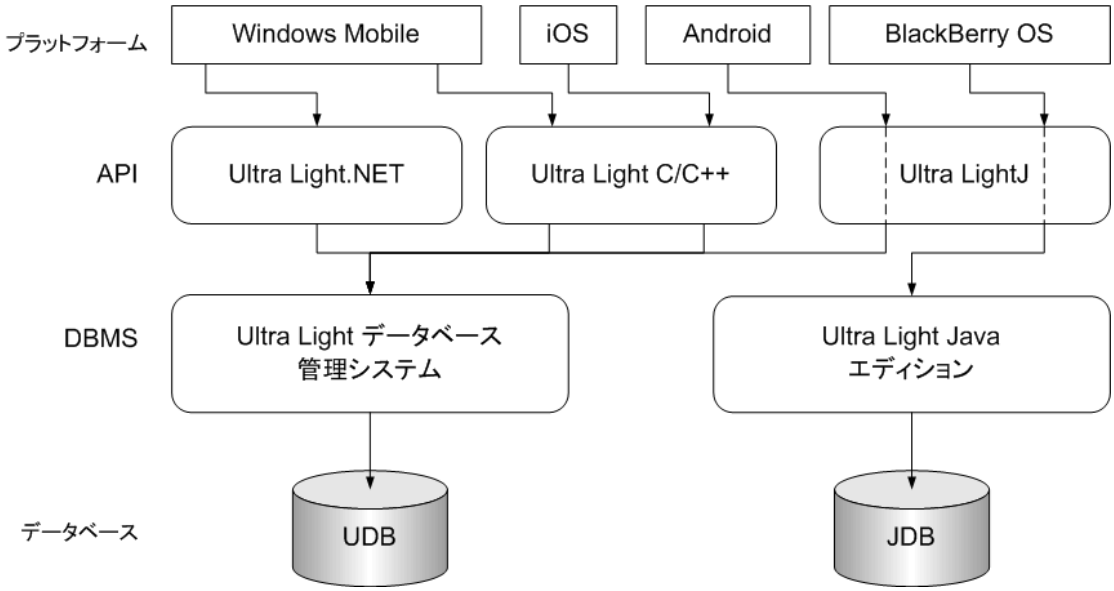
Ultra Light 概要

Ultra Light は、SQL Anywhere と同じ機能を数多く備えた、コンパクトなリレーショナルデータベース管理システムです。このシステムを使用して、スマートフォン、ハンドヘルドコンピュータ、タブレット PC などの小型デバイス用のモバイルデータベースを作成できます。

Ultra Light には、Ultra Light データベースおよび Ultra Light Java Edition データベース内で行われた変更を追跡し、ネットワーク上で Mobile Link サーバと更新を交換する同期クライアントが組み込まれています。Ultra Light が Mobile Link クライアントとして動作することで、モバイルアプリケーションと中央のデータベースや他の Ultra Light データベースとの同期を保持することができます。

Ultra Light のアーキテクチャ

次の図に示すように、多数のモバイルプラットフォームをサポートする Ultra Light は、API 開発、データベース管理、およびデータベースレイヤから構成されています。



- **サポートされるモバイルプラットフォーム**　モバイルプラットフォームによって、アプリケーション開発に使用できる Ultra Light API が異なります。 サポートされるプラットフォームとデバイスの詳細については、http://www.sybase.com/detail?id=1061806 を参照してください。

- **API 開発レイヤ**　目的のモバイルプラットフォームにどの API を使用できるかは、上の図を参照してください。 API 開発の詳細については、次の項を参照してください。

  - 「Ultra Light C++ アプリケーション開発」『Ultra Light C/C++ プログラミング』
  - 「Ultra Light.NET アプリケーション開発」『Ultra Light .NET プログラミング』
  - 「Ultra Light J アプリケーションの開発」『Ultra Light Java プログラミング』

- **データベース管理レイヤと同期クライアント**　Ultra Light には 2 つの管理システム、すなわちネイティブの「Ultra Light データベース管理システム」と「Ultra Light Java Edition」があります。

  - **Ultra Light データベース管理システム**　ほとんどのモバイルプラットフォームでは、Ultra Light API を使用して、Ultra Light データベース管理システムとインタフェースを取ります。このシステムにより、「Ultra Light データベース」を作成して、接続できます。

    Ultra Light プロジェクトを維持するために、包括的な管理ツールセットが用意されています。これらのツールは、コマンドラインユーティリティとして、または Sybase Central の Ultra Light プラグインのウィザードとして実行できます。 詳細については、「Ultra Light ユーティリティ」196 ページを参照してください。

  - **Ultra Light Java Edition**　BlackBerry OS プラットフォームでは、Ultra Light J API を使用して Ultra Light Java Edition とインタフェースを取り、「Ultra Light Java Edition データベース」を作成して、接続する必要があります。これらのデータベースには、Ultra Light データベースとの互換性はなく、いくつかの制限事項があります。 詳細については、「Ultra Light および Ultra Light Java Edition データベースの制限事項」9 ページを参照してください。

    Ultra Light には、Ultra Light Java Edition データベース上で追加のタスクを実行できる Java Edition ユーティリティのセットがあります。 詳細については、「Ultra Light Java Edition のユーティリティ」251 ページを参照してください。

  Ultra Light の同期クライアントの詳細については、「Ultra Light 同期クライアント」2 ページを参照してください。

- **データベースレイヤ**　このレイヤは、ファイルとして格納されるローカルデータレポジトリです。Ultra Light データベースは UDB ファイルとして、Ultra Light Java Edition データベースは JDB ファイルとして格納されます。UDB ファイルは、BlackBerry OS 以外のすべてのモバイルプラットフォームに移植できます。

Ultra Light 同期クライアント

Ultra Light には、Ultra Light データベース内のすべてのデータがデフォルトで同期される双方向同期クライアントが組み込まれています。Mobile Link 同期を初めて行う場合は、このデフォルトの動作を使用できます。業務上、必要であれば、カスタム同期設計により、統合データベースと同期させる Ultra Light データを変更できます。SQL Anywhere リモートデータベースとは異なり、同期機能を追加するために Ultra Light の占有容量を増やす必要がありません。

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

ローのステータスを追跡するメカニズム

テーブルとローのステータスの追跡は、データ同期時に特に重要です。Ultra Light データベースの各ローには、ローのステータス構造が関連付けられています。Ultra Light では、ローのステータスは、同期だけでなく、トランザクション処理とデータリカバリの制御にも使用されます。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

同期のステータスの追跡

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

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

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

参照

- 「Ultra Light データベースのローのステータス管理」476 ページ
- 「Mobile Link 同期」『Mobile Link クイックスタート』
- 「Mobile Link クライアントとしての Ultra Light」69 ページ
- 「チュートリアル : SQL Anywhere 統合データベースでの Mobile Link の使用」『Mobile Link クイックスタート』

Ultra Light のサポート対象プラットフォーム

Ultra Light のサポート対象のデバイス、プラットフォーム、ネットワークプロトコルの詳細については、http://www.sybase.com/detail?id=1061806 を参照してください。

参照

- 「Mobile Link クライアントとしての Ultra Light」69 ページ
- 「Ultra Light のネットワークプロトコルのオプション」111 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ

Ultra Light、Ultra Light Java Edition、SQL Anywhere の機能比較

次の表に、Ultra Light と SQL Anywhere の機能の比較を示します。Ultra Light の機能は、特に記載がなければ、Ultra Light Java Edition でもサポートされます。

注意
Ultra Light データベース管理システムを使用すると、750 - 1500 KB のアプリケーション容量が追加されます。Ultra Light Java Edition の場合は、500 KB が追加されます。一方、SQL Anywhere のデータベース、データベースサーバ、および同期クライアントによる追加分は 6 MB ほどになります。

機能	SQL Anywhere	Ultra Light	考慮事項
トランザクション処理、マルチテーブルジョイン	X	X	
トリガ、ストアドプロシージャ、ビュー	X		
外部ストアドプロシージャ (呼び出し可能な外部 DLL)	X		
組み込みの参照整合性とエンティティ整合性	X	X[1]	Ultra Light Java Edition には外部キー制約は適用されません。 「外部キー循環に関連する同期の問題の防止」79 ページを参照してください。
カスケード更新および削除	X	制限あり [1]	削除と更新がカスケードされる宣言参照整合性は、Ultra Light データベースではサポートされていない機能です。ただし、同期中はこの目的で削除が自動的にカスケードされます。
動的な複数データベースのサポート	X	X	
マルチスレッドアプリケーションのサポート	X	X	
ローレベルのロック	X	X	

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

機能	SQL Anywhere	Ultra Light	考慮事項
XML のアンロードとロードのユーティリティ		X	Ultra Light では、XML のロードとアンロードを行うために別々の管理ツールを使用します。これらの機能は、ランタイムには組み込まれていません。 Ultra Light については、次の項を参照してください。 ● 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ ● 「Ultra Light データベースのアンロードユーティリティ (ulunload)」232 ページ Ultra Light Java Edition については、次の項を参照してください。 ● 「Ultra Light Java Edition データベースロードユーティリティ (uljload)」252 ページ ● 「Ultra Light Java Edition データベースアンロードユーティリティ (uljunload)」254 ページ
SQLX 機能	X		
SQL 関数	X	X	Ultra Light アプリケーションでは、すべての SQL 関数を使用できるわけではありません。サポートされていない関数を使用すると、エラーが発生します。「Ultra Light SQL 関数」313 ページを参照してください。
SQL 文	X	X	SQL 文のスコープは、SQL Anywhere とは異なります。「Ultra Light SQL 文」416 ページを参照してください。
統合 HTTP サーバ	X		

機能	SQL Anywhere	Ultra Light	考慮事項
データベースファイルとネットワーク通信の強力な暗号化	X	X	
イベントのスケジュールと処理	X	X[1]	Ultra Light イベントモデルは SQL Anywhere とは異なります。
高パフォーマンス、セルフチューニング、コストベースのクエリオプティマイザ	X		Ultra Light にはクエリオプティマイザがありますが、SQL Anywhere のオプティマイザほど機能が豊富ではありません。
複数のスレッド対応 API	X	X	Ultra Light には独特の柔軟なアーキテクチャがあるので、アプリケーション開発者は、変化する配備環境や複数の異なる配備環境を対象としたアプリケーションを作成できます。「Windows Mobile 用の Ultra Light API の利点」17 ページを参照してください。
カーソルのサポート	X	X	「Ultra Light および Ultra Light Java Edition データベースの制限事項」を参照してください。

機能	SQL Anywhere	Ultra Light	考慮事項
動的なキャッシュサイズ設定	X	X[1]	Ultra Light では、データベースのキャッシュの初期サイズ、最小サイズ、最大サイズのいずれかまたはすべてを設定できます。 キャッシュのサイズは、Ultra Light によって最大サイズまで継続的に最適化されます (指定した場合)。次の項を参照してください。 ● 「Ultra Light CACHE_SIZE 接続パラメータ」168 ページ ● 「Ultra Light CACHE_MIN_SIZE 接続パラメータ」167 ページ ● 「UltraLite CACHE_MAX_SIZE 接続パラメータ」166 ページ Ultra Light Java Edition データベースでは、固定キャッシュサイズだけがサポートされます。
システムまたはアプリケーションの障害後のデータベースのリカバリ	X	X	
バイナリラージオブジェクト (BLOB) のサポート	X	X	Ultra Light では、BLOB のインデックス付けまたは比較を実行できません。
Windows パフォーマンスモニタの統合	X		
オンラインのテーブルとインデックスの断片化解除	X		
オンラインバックアップ	X		

機能	SQL Anywhere	Ultra Light	考慮事項
デスクトップから Windows Mobile デバイスへの直接デバイス接続		X[1]	SQL Anywhere データベースでは、Windows Mobile デバイスに配備するデータベースにデスクトップ接続できるようになるには、データベースサーバが必要です。Ultra Light では、接続文字列のプレフィクスとして **WCE:¥** を使用します。 「Windows Mobile」38 ページを参照してください。
インデックスの使用による高パフォーマンスの更新と検索	X	X	Ultra Light には、テーブルの検索時にインデックスを使用するか、ローを直接スキャンするかを決定するメカニズムがあります。 また、インデックスをハッシュしてデータ検索を高速化することもできます。 「Ultra Light max_hash_size 作成パラメータ」147 ページを参照してください。
HANA、Oracle、DB2、Sybase Adaptive Server Enterprise、Microsoft SQL Server、MySQL、または SQL Anywhere との同期	X	X	
組み込みの同期		X	SQL Anywhere とは異なり、Ultra Light では、クライアントエージェントによる同期を必要としません。同期は Ultra Light ランタイムに組み込まれているので、配備する必要があるコンポーネントが最小限に抑えられます。「Ultra Light クライアント」69 ページを参照してください。
プロセス内実行		X	
計算カラム	X		
宣言されたテンポラリテーブル／グローバルテンポラリテーブル	X		

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

機能	SQL Anywhere	Ultra Light	考慮事項
システム関数	X		
timestamp カラム	X	X	SQL Anywhere は、DEFAULT TIMESTAMP デフォルトをサポートしています。 Ultra Light がサポートしているのは、DEFAULT CURRENT TIMESTAMP デフォルトだけです。したがって、Ultra Light では、ローの更新時にタイムスタンプを自動的に更新することはできません。
ユーザベースのパーミッションのスキームによるオブジェクトベースの所有権とアクセスの決定	X		Ultra Light は、認証システムの必要がない、単一ユーザのデータベース向けに設計されています。ただし、最大 4 つのユーザ ID とパスワードを設定し、認証のためだけに使用できます。これらのユーザは、すべてのデータベースオブジェクトにアクセスできます。 「Ultra Light ユーザ」61 ページを参照してください。
空間データ	X	制限あり	Ultra Light は、ポイントデータのみをサポートしています。
完全なテキストデータ	X		

[1] Ultra Light Java Edition では使用できません。

Ultra Light および Ultra Light Java Edition データベースの制限事項

次の表に、Ultra Light データベースと Ultra Light Java Edition データベースに適用されるハード制限を示します。多くの場合、これらの制限はモバイルデバイスの最大容量を超えるものです。パフォーマンスとデバイスの容量を考慮すると、より厳しい制限が必要になります。

項目	Ultra Light データベースの制限事項	Ultra Light Java Edition データベースの制限事項
データベースとファイルサイズ	オペレーションシステムにサイズに対する制限がある場合は 4 GB 未満。	オペレーティングシステムによって制限されます。 データベースのページ数は 64000 に制限されています。
テンポラリファイルのサイズ	オペレーティングシステムによって制限されます。	適用外
キャッシュサイズ	デバイス上の使用可能メモリによって制限されます。	デバイス上の使用可能メモリによって制限されます。
動的なキャッシュサイズ設定	Ultra Light では、データベースのキャッシュの初期サイズ、最小サイズ、最大サイズのいずれかまたはすべてを設定できます。 キャッシュのサイズは、Ultra Light によって最大サイズまで継続的に最適化されます (指定した場合)。 次の項を参照してください。 ● 「Ultra Light CACHE_SIZE 接続パラメータ」 168 ページ ● 「Ultra Light CACHE_MIN_SIZE 接続パラメータ」 167 ページ ● 「UltraLite CACHE_MAX_SIZE 接続パラメータ」 166 ページ	Ultra Light Java Edition データベースでは、固定キャッシュサイズだけがサポートされます。
データベースでサポートされる同時オープン接続の最大数	最大 14	制限なし
すべてのデータベースへの同時オープン接続の最大数	モバイルデバイスでは最大 16、デスクトップでは最大 64。	制限なし
同時に実行できるデータベースの最大数	モバイルデバイスでは最大 8、デスクトップでは最大 32。	制限なし

項目	Ultra Light データベースの制限事項	Ultra Light Java Edition データベースの制限事項
データベースに同時に接続できるアプリケーションの最大数	Ultra Light エンジンを使用して、データベースに同時に接続する複数のアプリケーションを処理します。あるいは、一度にデータベースに接続できるアプリケーションを 1 つだけにします。	1
SQLCA (SQL Communication Area)	最大 63	適用外
テーブルあたりのロー数	最大 1600 万 場合によっては、ローに対する変更 (削除および更新) とその他のステータス情報がローデータとともに保持されます。この情報により、変更が同期されます。必然的に、同期間のテーブルのトランザクション数や、テーブルの同期の有無によっては、実際のローの制限が 1600 万より小さくなります。「Ultra Light でのトランザクション処理」477 ページを参照してください。	各データベースのページサイズと最大ページ数によって制限されます。
ローサイズ	パックされた各ローの長さはページサイズ以下である必要があります。「ローのパックとテーブル定義」50 ページを参照してください。 文字列がカラムサイズよりも短い場合は埋め込みなしで格納されます。LONG BINARY と LONG VARCHAR として宣言されたカラムは個別に格納されるため、除外されます。	パックされた各ローの長さはページサイズ以下である必要があります。「ローのパックとテーブル定義」50 ページを参照してください。 文字列がカラムサイズよりも短い場合は埋め込みなしで格納されます。LONG BINARY と LONG VARCHAR として宣言されたカラムは個別に格納されるため、除外されます。
データベースあたりのロー数	永続ストアによって制限されます。	永続ストアによって制限されます。
テーブルの大きさ	データベースのサイズによって制限されます。	データベースのサイズによって制限されます。

項目	Ultra Light データベースの制限事項	Ultra Light Java Edition データベースの制限事項
データベースあたりのテーブル数	データベースのサイズによって制限されます。	最大 32000
テーブルあたりのカラム数	ローサイズはページサイズによって制限されるため、テーブルあたりのカラム数の上限は、実際にはページサイズによって決まります。一般的に、実質的な制限は 4000 よりもかなり小さくなります。	ローサイズはページサイズによって制限されるため、テーブルあたりのカラム数の上限は、実際にはページサイズによって決まります。一般的に、実質的な制限は 4000 よりもかなり小さくなります。
テーブルあたりのインデックス数	データベースのサイズによって制限されます。	データベースのサイズによって制限されます。
パブリケーション数	最大 63	最大 63
データベースページサイズ	最小 1 K、最大 16 KB	最小 256 バイト、最大 16 KB
接続ごとのカーソル数	Ultra Light データベースへの特定の接続で使用できるカーソルの最大数は 64 です (すべてのプラットフォーム)。	Ultra Light データベースへの特定の接続で使用できるカーソルの最大数は 64 です (すべてのプラットフォーム)。
文字列	ローがページ上に収まる必要があります。	ローがページ上に収まる必要があります。
バイナリデータ型	ローがページ上に収まる必要があります。	ローがページ上に収まる必要があります。
long binary と long varchar のサイズ	データベースサイズによってのみ制限されます。	データベースサイズによってのみ制限されます。
blob サイズ	ファイルサイズによって制限されます。	最大 2^{24} バイト
使用できる独立性レベル	0 (コミットされていない読み出し) または 1 (コミットされた読み出し)	0 (コミットされていない読み出し)

項目	Ultra Light データベースの制限事項	Ultra Light Java Edition データベースの制限事項
カスケード更新および削除	削除と更新がカスケードされる宣言参照整合性は、Ultra Light データベースではサポートされていない機能です。ただし、同期中はこの目的で削除が自動的にカスケードされます。	サポートされていません。
イベントのスケジュールと処理	Ultra Light イベントモデルは SQL Anywhere とは異なります。	サポートされていません。
Ultra Light Java Edition データベースの互換性	Ultra Light データベースは、Ultra Light Java Edition データベースと互換性がありません。Ultra Light データベースと Ultra Light Java Edition データベースは、ロードとアンロードのユーティリティを使用して相互に変換することができます。 次の項を参照してください。 ● 「Ultra Light データベースのアンロードユーティリティ (ulunload)」 232 ページ ● 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」 219 ページ	Ultra Light データベースは、Ultra Light Java Edition データベースと互換性がありません。Ultra Light データベースと Ultra Light Java Edition データベースは、ロードとアンロードのユーティリティを使用して相互に変換することができます。 次の項を参照してください。 ● 「Ultra Light Java Edition データベースアンロードユーティリティ (uljunload)」 254 ページ ● 「Ultra Light Java Edition データベースロードユーティリティ (uljload)」 252 ページ

BlackBerry 用のオブジェクトストアの制限事項

BlackBerry スマートフォン上でのデータベースストアのサイズは、利用可能なオブジェクトハンドル数によって制限されます。利用可能なオブジェクトハンドル数は、BlackBerry の機能であり、フラッシュメモリのサイズで決まります。

Ultra Light には、データベース値をメモリに格納するため、ハンドルタイプに応じて 1 つ以上のオブジェクトハンドルが必要です。たとえば、10 個のカラムと 2 つのインデックスがあるテーブルのローには、最低 12 個のオブジェクトハンドルが必要です。

より大型のデータベースストアを使用するため、Ultra Light J API では、メモリに保持するロー数をユーザが制限できるようになっています。格納されたデータベースのローは、データベースページで結合されます。Ultra Light J API では、各データベースページに 1 つの永続的なオブジェクトハンドルだけが必要です。

参照

- 「SQL Anywhere のサイズと数の制限」『SQL Anywhere サーバ データベース管理』

CustDB サンプルアプリケーションの概要

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

CustDB は、次のコンポーネントで構成されています。

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

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

CustDB を使用すると、販売担当者は取引を追跡、監視し、次の 2 種類のユーザの情報をプールできます。

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

これらのユーザによって収集された情報は、統合データベースと同期できます。

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

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

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

Sybase Central を使用すると、統合データベース内の同期スクリプトをブラウズできます。Sybase Central は、データベースにスクリプトを追加するためのプライマリツールです。

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

SQL Anywhere CustDB データベース

これは統合データベースです。インストール中、このデータベース用に SQL Anywhere 16 CustDB という名前の ODBC データソースが作成されます。データベースファイルは、*%SQLANYSAMP16%¥UltraLite¥CustDB* にあります。このデータベースは、*makedbs.cmd* (Mac または Linux の場合 *makedbs.sh*) を使用して、いつでも再作成できます。

統合された *CustDB.db* ファイルに同期された変更内容を消去できるため、次のスクリプトを使用して新しいバージョンを作成します。*C:¥Program Files¥SQL Anywhere 16¥UltraLite¥CustDB¥newdb.bat*

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

Ultra Light CustDB データベース

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

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

- Ultra Light.NET の場合：*C:¥Program Files¥SQL Anywhere 16¥UltraLite.NET¥CustDB¥Common¥custdb.udb*
- その他のすべてのプラットフォームと API の場合：*C:¥Program Files¥SQL Anywhere 16¥UltraLite¥CustDB¥custdb.udb*

Ultra Light データベースは *makedbs.cmd* スクリプト (Mac または Linux の場合 *makedbs.sh*) でも再作成されます。

RDBMS 固有の構築スクリプト

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

C:¥Program Files¥SQL Anywhere 16¥MobiLink¥CustDB ディレクトリには次のファイルがあります。

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

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

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

Ultra Light リモートデータベースへの使いやすいインタフェースを提供するエンドユーザアプリケーションです。サポートされているプラットフォームごとにサンプルクライアントがインストールされます。

各クライアントアプリケーションには、*ReadMe.html* ファイルまたは *ReadMe.txt* ファイルが含まれています。各ファイルには、サンプルの構築と実行に必要な手順の概要が含まれています。

アプリケーションと ReadMe のロケーションは、開発環境によって異なります。「レッスン 1：CustDB アプリケーションの構築と実行」129 ページを参照してください。

同期ロジック

Ultra Light データベース SQL 文と同期の呼び出しは、C++ API の場合 *custdbcpp.cpp*、Embedded SQL の場合 *custdb.sql* にあります。

参照

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

CustDB ファイルのロケーション

CustDB アプリケーションは、さまざまな開発環境用に構築されています。

Windows 32 ビットデスクトップ用 Ultra Light

CustDB アプリケーションを実行する前に構築する必要はありません。

CustDB 実行ファイルは *%SQLANY16%¥UltraLite¥Windows¥x86* ディレクトリにあります。

Ultra Light for C/C++

- **C/C++ のすべてのバージョン**　C/C++ には多くの開発環境があるので、C/C++ CustDB プロジェクトファイルには複数のバージョンが用意されています。ほとんどのバージョンでは汎用ファイルが使用されます。これらのファイルは *C:¥Program Files¥SQL Anywhere 16¥UltraLite¥Custdb* ディレクトリにあります。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

C/C++ CustDB アプリケーションのすべてのバージョンについては、*C:¥Program Files¥SQL Anywhere 16¥UltraLite¥Custdb¥readme.txt* を参照してください。

- **Visual Studio**　プロジェクトファイルは、*%SQLANYSAMP16%¥UltraLite¥CustDB¥* ディレクトリにあります。
- **iPhone 用 Xcode**　Mac OS X および iPhone で開発する場合、*samples/ultralite/custdb/iphone* を参照してください。

Ultra Light.NET

Microsoft Visual Studio 用のプロジェクトファイルは *C:¥Users¥Public¥Documents¥SQL Anywhere 16¥Samples¥UltraLite.NET¥CustDB* ディレクトリにあります。

Windows Mobile 用の Ultra Light ソリューションの考慮事項

この項では、Windos Moble 開発に使用できる Ultra Light の設計オプションと、各オプションを実装した場合の利点について説明します。次の選択について説明します。

- **Ultra Light API の選択**　Windows Mobile 開発用の各 API を使用した場合の利点が説明されています。
  - Ultra Light C/C++
  - Ultra Light Embedded SQL
  - Ultra Light.NET
- **データ管理コンポーネントの選択**　Windows Mobile 開発用の次のデータ管理コンポーネントを使用した場合の利点が説明されています。
  - Ultra Light インプロセスランタイム環境
  - Ultra Light データベースエンジン

Windows Mobile 用の Ultra Light API の利点

Ultra Light C/C++、Embedded SQL、.NET API には、単純なテーブルベースのデータアクセスインタフェースや、複雑なクエリ用の動的 SQL など、いくつかのデータアクセスモデルがあります。これらの利点を組み合わせることによって、Ultra Light はアプリケーション開発者に、さまざまな配備環境向けのアプリケーションを作成できる柔軟なアーキテクチャを提供しています。

Ultra Light.NET API の利点

Ultra Light.NET API は通常、Windows Mobile 開発に推奨されます。SQL Anywhere .NET API は、Ultra Light コンポーネントと SQL Anywhere で共有される共通プログラミングモデルを提供し、.NET プログラミングは C/C++ と比較されるからです。

Ultra Light C/C++ と Embedded SQL API の利点

Ultra Light がさまざまな環境および使用状況で高いパフォーマンスを実現するのに対し、最低レベルの API である Embedded SQL と Ultra Light C++ API は、一般的に最高のパフォーマンスを実現します。

作成するアプリケーションのサイズを最小限に抑えたい場合は、Ultra Light C/C++ API を使用してください。これらのアプリケーションでは通常、小さなアプリケーションファイルサイズを維持しながら、最良のパフォーマンスを実現できます。

参照

- 「.NET アプリケーションプログラミング」『SQL Anywhere サーバ プログラミング』
- 「Ultra Light C++ アプリケーション開発」『Ultra Light C/C++ プログラミング』
- 「Embedded SQL を使用した Ultra Light C++ アプリケーション開発」『Ultra Light C/C++ プログラミング』
- 「Ultra Light.NET アプリケーション開発」『Ultra Light .NET プログラミング』

Windows Mobile 用の Ultra Light データ管理コンポーネント

Ultra Light では、個別データベースサーバを設定するための追加オーバヘッドを必要とせずに、サイズの小さいリレーショナルデータベースソリューションを構築できます。代わりに、Ultra Light プログラミングインタフェースは、2 つのアプローチ－「Ultra Light インプロセスランタイムライブラリ」と「Ultra Light データベースエンジン」 - のどちらかを使用します。どちらのアプローチも、接続とデータアクセス要求を制御します。

どちらのコンポーネントにも、Ultra Light データベースと Mobile Link 同期サーバをリンクする双方向の同期クライアントが組み込まれています。 Ultra Light の同期クライアントの詳細については、「Ultra Light 同期クライアント」2 ページを参照してください。

Ultra Light インプロセスランタイムライブラリ

データベースに一度にアクセスする必要があるアプリケーションが 1 つだけの場合は、Ultra Light インプロセスランタイムを推奨します。

ランタイムとアプリケーションは一体のプロセスであり、データベースはアプリケーションに固有のものとなります。ランタイムライブラリによって、Ultra Light データベースと組み込み同期処理が管理されます。

ランタイムにリンクするには、エンジンとは別のインポートライブラリ/DLL ペアを使用する必要があります。

Ultra Light では、静的リンクと動的リンクの両方がサポートされます。

- **静的リンク**　静的リンクは必要なデバイスメモリが少なく、デバイス上で 1 つの Ultra Light アプリケーションだけを使用する場合には、より効率的です。

- **動的リンク**　動的リンクは、デバイス上で複数の Ultra Light アプリケーションを使用する場合には、デバイスメモリをより節約できます。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

Ultra Light データベースエンジン (*uleng16.exe* ユーティリティ)

Ultra Light エンジンが使用可能なプラットフォームは、Windows デスクトップと Windows Mobile のみです。エンジンは別個の実行プログラムであり、複数のアプリケーションからの同時アクセスがサポートされます。各アプリケーションは、Ultra Light エンジンと通信するために、クライアントライブラリを使用する必要があります。

Ultra Light エンジンは Ultra Light ランタイムよりも多くのシステムリソースを必要とし、クライアントとデータベース間で大量のデータが移動する場合には、パフォーマンスが低下することがあります。

エンジンに接続するには、ランタイムとは別のインポートライブラリ/DLL ペアを指定する必要があります。

Ultra Light エンジンは、次の状況で必要になります。

- 複数のプロセスが同じデータベースファイルにアクセスし、しかも同時にアクセスする可能性がある場合 (同時にとは、複数のプロセスに同時に同じデータベースに対する開かれた接続があることを意味します)。
- 中央管理を使用して、Ultra Light アプリケーションデータベースを管理する場合。

参照

- 「Ultra Light アプリケーションのビルドと配備の仕様」113 ページ
- 「Ultra Light C++ アプリケーションの構築と配備」『Ultra Light C/C++ プログラミング』
- 「Ultra Light エンジンの起動」124 ページ
- 「Ultra Light エンジンユーティリティ (uleng16)」207 ページ
- 「Ultra Light の同時実行性」475 ページ
- 「Ultra Light、Ultra Light Java Edition、SQL Anywhere の機能比較」3 ページ
- 「Ultra Light および Ultra Light Java Edition データベースの制限事項」9 ページ

Ultra Light データベースの作成方法

データベースには、次の 3 つの作成方法があります。

●データベース作成用の Ultra Light 管理ツールを使用してデスクトップで作成する方法。

●Ultra Light の API を使用してデバイス上で作成する方法。

デバイス上で作成する方法については、主に各 API に固有の Ultra Light プログラミングマニュアルで説明しています。

●デバイス上で Ultra Light データベースを作成するように設定された中央管理リモートタスク。

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

参照

- 「Ultra Light データベース作成パラメータ」25 ページ
- 「データベースオプションへのアクセス」39 ページ

データベース作成ウィザードでの Ultra Light データベースの作成

Sybase Central を使用してデータベースを作成します。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

使用可能なデータベース作成パラメータのナビゲーションを使用して作成する場合は、この方法を使用します。このウィザードを使用すると、選択するターゲットのプラットフォームに基づいて設定できる項目が制限されるので、選択が簡単になります。データベースが作成されると、コマンドライン構文が表示されます。この構文は、記録して、ulinit ユーティリティで使用できます。

♦ タスク

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 16]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
2. **[ツール]** » **[Ultra Light 16]** » **[データベースの作成]** をクリックします。
3. データベース作成ウィザードの指示に従います。

結果

データベースが作成されます。

次の手順

これで、データベースに接続してテーブルおよびその他のデータベースオブジェクトを作成できます。

参照

- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベース作成パラメータ」25 ページ

コマンドプロンプトでの Ultra Light データベースの作成

コマンドプロンプトでデータベースを作成するには、次のユーティリティのいずれかを選択します。

- 空の新しい Ultra Light データベースまたは SQL Anywhere のリファレンスデータベーススキーマに基づく Ultra Light データベースを作成する場合は、ulinit ユーティリティを使用します。このユーティリティをオプションとともに使用すると、データベースを設定できます。「SQL Anywhere データベースから Ultra Light データベースへの変換」33 ページを参照してください。

- 新しい Ultra Light データベースのスキーマかデータまたはその両方のソースとして使用する XML ファイルがある場合は、ulload ユーティリティを使用します。「XML ファイルからの Ultra Light データベースの作成」23 ページを参照してください。

- 集中管理：配備したすべてのデバイスに Mobile Link エージェントが設定されている場合、またはアプリケーションで初期の Ultra Light データベースを配備できない場合は、この方法を選択します。これらのデバイスに新しい Ultra Light データベースを作成するようにリモートタスクを設定できます。これで、このデータベースを管理者が集中管理できます。「リモートデータベースの集中管理」『Mobile Link サーバ管理』を参照してください。

新しい Ultra Light データベースの作成 (コマンドラインの場合)

デフォルトを受け入れる場合は、新しい Ultra Light データベースファイルを指定する次の ulinit ユーティリティを実行してください。

```
ulinit test.udb
```

サポートされるオプションの詳細については、「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページを参照してください。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

Mobile Link 同期モデルでの Ultra Light データベースの作成

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

複数のリモート Ultra Light データベースと集中型の統合データベースがある同期システムを作成している場合は、この方法を使用します。

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

参照

- 「Sybase Central の Mobile Link プラグイン」『Mobile Link クイックスタート』

リモートデータベースの集中管理による Ultra Light データベースの作成

Mobile Link は、Ultra Light データベースを作成できるデータベース作成コマンドを提供します。

参照

- 「create database コマンド」『Mobile Link サーバ管理』
- 「リモートデータベースの集中管理」『Mobile Link サーバ管理』

XML ファイルからの Ultra Light データベースの作成

Ultra Light データベースを管理する中間のフォーマットとして XML を使用します。

前提条件

Ultra Light では任意の XML ファイルは使用できません。*%SQLANY16%¥Bin32* ディレクトリと *%SQLANY16%¥Bin64* ディレクトリにはスキーマ定義を含む *usm.xsd* ファイルがあります。このファイルを使用して、XML フォーマットを確認してください。

内容と備考

XML を使用すると次のことができます。

- データベースのプロパティやオプションが異なる新しいデータベースにデータをロードする。
- Ultra Lite の旧バージョンで作成されたデータベースからスキーマをアップグレードする。
- テキストバージョンの Ultra Light データベースを作成する。

♦ タスク

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

結果

データベースが作成されます。

次の手順

これで、データベースに接続してテーブルおよびその他のデータベースオブジェクトを作成できます。

例

sample.xml 内のテーブルフォーマットとデータから、新しい Ultra Light データベースを *sample.udb* ファイルに作成するには、次のコマンドを実行します。

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

参照

- 「XML としてエクスポートされるリレーショナルデータ」『SQL Anywhere サーバ SQL の使用法』
- 「データベース作成ウィザードでの Ultra Light データベースの作成」21 ページ
- 「コマンドプロンプトでの Ultra Light データベースの作成」22 ページ
- 「SQL Anywhere データベースから Ultra Light データベースへの変換」33 ページ
- 「Ultra Light のアップグレード」『SQL Anywhere 16 変更点とアップグレード』
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- 「Ultra Light データベース作成パラメータ」25 ページ

初回接続時の Ultra Light データベースの作成

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

考慮事項

追加データベースの作成と SQL コードを追加することで、アプリケーションのサイズが大幅に増大する可能性があります。ただし、アプリケーションをデバイスに配備するだけなので、配備

作業は簡単になります。また、運用前の開発サイクルで、テストのためにデバイス上のデータベースを削除し、再構築する場合にもこの方法を使用できます。

参照

- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET] 『Ultra Light .NET プログラミング』
- 「Ultra Light および Ultra Light Java Edition のデータベースの作成および接続方法」『Ultra Light Java プログラミング』

Ultra Light データベース作成パラメータ

作成パラメータは、Ultra Light データベースの最初の作成時に、データベースを設定するために使用されます。これらの設定を変更する場合には、データベースを再作成する必要があります。

Ultra Light データベースを作成するときに管理できるオプションがいくつかあります。これらのオプションは Ultra Light の多様な用途に役立つように設計されています。作成時に指定したパラメータのほとんどは後で変更できません。

データベースの作成時に作成パラメータを指定するには、ulinit または ulload の各ユーティリティを使用するか、サポートされているクライアントインタフェースから指定します。

ブール作成パラメータは、YES、Y、ON、TRUE、T、1 のいずれかによってオンになり、NO、N、OFF、FALSE、F、0 のいずれかによってオフになります。パラメータの大文字と小文字は区別されません。

Ultra Light 作成パラメータは、プログラミングインタフェースからデータベースを作成する場合はセミコロンで区切られた文字列として指定され、コマンドラインユーティリティを使用する場合は個別のコマンドラインパラメータとして指定されます。次に例を示します。

```
ulinit --case --utf8_encoding=1 test.udb
```

または、複数の -c オプションを指定することもできます。

データベースオプションまたは接続パラメータを使用して、データベースの他の設定を行うことができます。

作成パラメータの値へのアクセス

データベース作成後に作成パラメータの値を変更することはできません。ただし、Sybase Central で対応するデータベースプロパティを表示することはできます。

Ultra Light J API の場合、getCreationString メソッドを使用して、setCreationString メソッドで登録した作成文字列を表示できます。

その他の Ultra Lght API の場合は、その Ultra Light API に適切な GetDatabaseProperty 関数を呼び出すことによって、Ultra Light アプリケーションからデータベースプロパティにプログラム的にアクセスできます。

参照

- 「Ultra Light データベースオプション」191 ページ
- 「Ultra Light データベースへの接続」35 ページ
- 「Ultra Light 接続パラメータ」164 ページ
- 「データベースプロパティの読み取り」39 ページ
- ULConnection.GetDatabaseProperty メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseSchema.GetDatabaseProperty メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ConfigPersistent.getCreationString メソッド [Android] [Ultra Light J]『Ultra Light Java プログラミング』
- ConfigPersistent.setCreationString メソッド [Android] [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light 文字セット

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

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

- 適切なソート順。データベースに格納する文字を適切にソートできる照合を選択してください。

- デバイスのプラットフォーム。サポートされているデバイスによって要件が異なり、一部のデバイスでは、UTF-8 で文字をエンコードする必要があります。複数のデバイスをサポートする必要がある場合は、データベースを共有できるかどうかを確認する必要があります。

- データを同期する場合は、統合データベースでサポートされている言語と文字セット。Ultra Light データベースと統合データベースで使用する文字セットに互換性があることを確認します。互換性がなく、一方のデータベースの文字セットにある文字が他方の文字セットになかった場合、データが失われたり、予期せず変更される可能性があります。Ultra Light を多言語環境に配備する場合も、Ultra Light データベースを UTF-8 でエンコードしてください。

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

  1. Ultra Light データベースの文字が Unicode に変換されます。

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

Ultra Light Java Edition の文字セットと照合

Ultra Light Java Edition のデータベースは、UTF-8 (Unicode) としてエンコードされます。照合は Java のデフォルトのソート順であり、SQL Anywhere がサポートする UTF8BIN 照合と同等です。Ultra Light は、Mobile Link サーバとの同期中に、UTF8 文字セットと照合を使用することを Mobile Link に通知します。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

参照

- 「Ultra Light での文字セットのエンコードに関するプラットフォーム要件」27 ページ
- 「Ultra Light collation 作成パラメータ」141 ページ
- 「Ultra Light utf8_encoding 作成パラメータ」163 ページ
- 「文字セット」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light 接続パラメータ」164 ページ
- 「文字セットの考慮事項」『Mobile Link サーバ管理』
- 「Ultra Light case 作成パラメータ」139 ページ
- 「データベースセキュリティ」29 ページ

Ultra Light での文字セットのエンコードに関するプラットフォーム要件

各プラットフォームには特定の文字セットとエンコードの要件があります。

Windows デスクトップと Windows Mobile

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

参照

- 「Ultra Light utf8_encoding 作成パラメータ」163 ページ
- 「文字セット」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light 接続パラメータ」164 ページ
- 「文字セットの考慮事項」『Mobile Link サーバ管理』
- 「データベースセキュリティ」29 ページ

Ultra Light でサポートされている照合

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

```
ulinit -Z
```

Ultra Light Java Edition のデータベースでサポートされるのは、UTF8BIN 照合だけです。

照合ラベル	説明
1250LATIN2	Code Page 1250、Windows Latin 2、中央／東ヨーロッパ言語
1250POL	Code Page 1250、Windows Latin 2、ポーランド語

照合ラベル	説明
1251CYR	Code Page 1251、Windows キリル語
1252LATIN1	Code Page 1252、Windows Latin 1、西ヨーロッパ言語
1252NOR	Code Page 1252、Windows Latin 1、ノルウェー語
1252SPA	Code Page 1252、Windows Latin 1、スペイン語
1252SWEFIN	Code Page 1252、Windows Latin 1、スウェーデン語／フィンランド語
1253ELL	Code Page 1253、Windows ギリシア語、ISO8859-7 拡張付き
1254TRK	Code Page 1254、Windows トルコ語、ISO8859-9 拡張付き
1254TRKALT	Code Page 1254、Windows トルコ語、拡張付き ISO8859-9、I-dot と I-no-dot は同じ
1255HEB	Code Page 1255、Windows ヘブライ語、ISO8859-8 拡張付き
1256ARA	Code Page 1256、Windows アラビア語、ISO8859-6 拡張付き
1257LIT	Code Page 1257、Windows バルト諸国語、リトアニア語
874THAIBIN	Code Page 874、Windows Thai、ISO8859-11、バイナリ順序
932JPN	Code Page 932、日本語シフト JIS、Microsoft 拡張文字付き
936ZHO	Code Page 936、中国語 (簡体文字)、PRC GBK
949KOR	Code Page 949、韓国語 KS C 5601-1987 コード、完成型
950ZHO_HK	Code Page 950、中国語 (繁体文字)、Big 5 コード (HKSCS を含む)
950ZHO_TW	Code Page 950、中国語 (繁体文字)、Big 5 コード
EUC_CHINA	中国語 (簡体文字)、GB 2312-80 コード
EUC_JAPAN	日本語の EUC JIS X 0208-1990 と JIS X 0212-1990 コード
EUC_KOREA	Code Page 1361、韓国語 KS C 5601-1992 8 ビットコード、Johab (組合型)
EUC_TAIWAN	Code Page 964、EUC-TW エンコード
ISO1LATIN1	ISO8859-1、ISO Latin 1、西ヨーロッパ言語、Latin 1 順序
ISO9LATIN1	ISO8859-15、ISO Latin 9、西ヨーロッパ言語、Latin 1 順序

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

データベースセキュリティ

データベースには、暗号化または難読化のオプションがあります。暗号化の場合はデータベースのデータが非常に安全に表現されますが、難読化はデータベースの内容が簡単には読み取れないというだけです。

デフォルトでは、データベースは暗号化も難読化もされません。テキストカラムやバイナリカラムは、16 進エディタなどの表示ツールを使用して読むことができます。データをプレーンテキストとして格納したくない場合は、次のオプションを検討してください。

- **難読化**　このオプションは、データベース内のデータにアクセスしようとする軽い試行を防ぐことはできますが、強力な暗号化と同様の厳重なセキュリティは提供しません。難読化は、パフォーマンスにほとんど影響しません。デバイスで使用するのが単純な難読化の場合、特別な設定は不要です。
- **AES 256 ビットの暗号化**　このオプションを使用すると、AES 256 ビットアルゴリズムでデータベースが暗号化されます。強力な暗号化は、巧妙で容赦ないデータへのアクセス試行に対するセキュリティを提供しますが、パフォーマンスに大きな影響を与えます。デバイスで使用するのが AES 暗号化の場合、特別な設定は不要です。
- **AES 256 ビット FIPS 140-2 認定暗号化 (Windows と Windows Mobile のみ)**　米国政府とカナダ政府の FIPS 140-2 規格で認定された暗号化ライブラリ (Certicom 認定の暗号化モジュールを使用) を使用できます。AES FIPS 認定暗号化を使用するには、デバイスを適切に設定する必要があります。

> **注意**
> 別途ライセンスが必要な必須コンポーネント
>
> FIPS 認定の暗号化には別のライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
>
> 「別途ライセンスが必要なコンポーネント」『SQL Anywhere 16 紹介』を参照してください。

データベースを暗号化または難読化する場合は、データベースのキャッシュサイズへの影響を考慮してください。5 ～ 10% オーバヘッドが増大するため、パフォーマンスが低下する可能性があります。パフォーマンスへの影響の度合いは、キャッシュのサイズによって異なります。キャッシュがきわめて小さい場合は、暗号化によって著しいオーバヘッドが追加されることがあります。一方、キャッシュが十分に大きければ、まったく影響を感じない場合もあります。 キャッシュサイズの詳細については、「Ultra Light データベースのキャッシュサイズの調整」461 ページを参照してください。

データベースの難読化

データを難読化するには、データベースの作成時にデータベース作成パラメータとして obfuscate=1 を指定します。エンドユーザが対応する接続パラメータを指定する必要はありません。

Ultra Light J API でデータを難読化するには、データベース作成中に ConfigPersistent.enableObfuscation メソッドを使用します。

データベースの暗号化

高い効果を得るために、暗号化キーは、文字、数字、特殊記号を組み合わせて作成する必要があります。暗号化キーは、長くするほど推測されにくくなります。

注意
データベースを暗号化した後は、暗号化キーは復元できません。

Sybase Central ウィザードでは、**[データベースを暗号化する]** オプションをクリックしてから **[強力な暗号化と AES を使用]** をクリックすることにより、Ultra Light データベース作成中に暗号化オプションを指定できます。

ulinit ユーティリティでは、DBKEY パラメータを使用して暗号化キーを指定できます。FIPS 認定の暗号化を使用するかどうかを指定するには、FIPS 作成パラメータを使用します。

Ultra Light API 暗号化オプションは、データベース作成時に使用できます。Ultra Light Java Edition データベースの場合は、DBKEY パラメータの代わりに enableAesDBEncryption メソッドおよび setEncryptionKey メソッドを使用してデータベースキーを指定します。

警告
暗号化キーはデータベースの作成後に変更できますが、変更する場合は細心の注意が必要です。

この操作は負荷が高く、回復できません。操作が途中で停止すると、データベースが完全に失われる可能性があります。

データベースに強力な暗号化を適用した場合、暗号化キーのコピーを必ず安全な場所に保管してください。暗号化キーがわからなくなった場合は、Sybase 製品の保守契約を結んでいるサポートセンタに依頼してもデータにはアクセスできません。アクセスできなくなったデータベースは、廃棄して、新しくデータベースを作成する必要があります。

詳細については、次の項を参照してください。

- ULConnection.ChangeEncryptionKey メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.ChangeEncryptionKey メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.changeEncryptionKey メソッド [Ultra Light J]『Ultra Light Java プログラミング』

DBKEY パラメータは、データベースへの接続時に指定される必要があります。指定されないと接続に失敗します。暗号化キーは機密情報として扱う必要があります。

参照

- 「Ultra Light obfuscate 作成パラメータ」150 ページ
- 「Ultra Light DBKEY 接続パラメータ」175 ページ
- 「Ultra Light fips 作成パラメータ」146 ページ
- 「Ultra Light データベースの作成方法」21 ページ
- 「Ultra Light データベース作成パラメータ」25 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- ConfigPersistent.enableObfuscation メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ConfigPersistent.setCreationString メソッド [Android] [Ultra Light J]『Ultra Light Java プログラミング』
- ConfigPersistent.enableAesDBEncryption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- ConfigPersistent.getEncryptionKey メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- ConfigPersistent.setEncryptionKey メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- 「別途ライセンスが必要なコンポーネント」『SQL Anywhere 16 紹介』

SQL Anywhere データベースから Ultra Light データベースへの変換

SQL Anywhere リファレンスデータベースから Ultra Light データベースを作成するには、-a オプションを指定して ulinit ユーティリティを実行します。新しいデータベースは、可能な限りリファレンスデータベースと同じ設定で作成されます。

SQL Anywhere リファレンスデータベースはデータベーステンプレートとして機能し、Ultra Light データベーススキーマの作成に次の設定を使用します。

- データベースの設定 (照合順など)
- テーブル定義
- 同期パブリケーション

リファレンスデータベースのパブリケーションの一部として、データを含めたり、カラム、テーブル、インデックスを選択することもできます。

> **注意**
> SQL Anywhere 以外の RDBMST から Ultra Light データベースを初期化するには、Sybase Central で**同期モデル作成ウィザード**を使用します。スキーマ情報の取得を要求されたら、統合データベースに接続してください。

変換に関する考慮事項

ulinit ユーティリティを実行する前に、次のリファレンスデータベースタスクが必要かどうか検討してください。

- **必要に応じてテーブル、キー、インデックス、および同期パブリケーションを追加する。** 必要に応じてテーブルを追加し、プライマリキーを設定します。Ultra Light アプリケーション内で必要な外部キー関係を指定することもできます。

  インデックスを使用すると、特に低速のデバイスでパフォーマンスが大きく向上します。プライマリキーカラムには自動的にインデックスが付きますが、他のカラムタイプには付きません。

  > **ヒント**
  > Ultra Light アプリケーションで情報を特定の順序で取り出すことが多い場合は、リファレンスデータベースにインデックスを追加することを検討してください。

  異なるテーブルを異なる時期に同期する場合は、同期パブリケーションを使用します。複数の同期パブリケーションを使用して、テーブルのサブセットを定義し、サブセットごとに同期の優先度を設定できます。

- **望ましくない影響を及ぼす可能性のあるデータベースオプションまたはテーブルスキーマを更新する。** たとえば、SQL Anywhere データベースのカラムに含まれる句が Ultra Light ではサポートされていない場合、デフォルト値は無視され、新しいデータベースには Ultra Light のデフォルト値が指定されます。

- **Ultra Light でサポートされていない場合は照合順を変更する。** Ultra Light では、リファレンスデータベースに定義されている照合順の名前が使用されます。utf8_encoding プロパティを設定すると、UTF-8 を使用してエンコードできます。

  Ultra Light でサポートされている照合および対応するコードページのリストを表示するには、コマンドプロンプトで -Z オプションを指定して ulinit を実行します。CHAR に対する UCA 照合順など、サポートされていない照合順がリファレンスデータベースで使用されている場合は、次の手順でサポートされている照合順に変更します。

  1. Unload ユーティリティを使用して、SQL Anywhere リファレンスデータベースをアンロードします。

  2. 異なる照合を使用する新しい SQL Anywhere データベースを作成し、Interactive SQL で *reload.sql* スクリプトを実行します。

参照

- 「アンロードユーティリティ (dbunload)」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light のテーブルとカラム」50 ページ
- 「インデックスを使用する場合」58 ページ
- 「Ultra Light でのデータのパブリッシュ」78 ページ
- 「インデックススキャンの作成と保持」463 ページ
- 「Ultra Light データベースの作成方法」21 ページ

例

次のコマンドでは、*MySADb* データソースに定義されている既存の SQL Anywhere リファレンスデータベースから、*customer.udb* という名前の新しい Ultra Light データベースが作成されます。リファレンスデータベースのテーブルは TestPublication に定義されています。作成された Ultra Light データベースには、TestPublication 内とすべて同じデータベースオプションとテーブルが含まれ、データベースは **mykey** 暗号化キーで暗号化されます。

```
ulinit -a "DSN=MySADb;UID=JimmyB;PWD=secret" -n TestPublication -k mykey customer.udb
```

次のコマンドでは、*MySource.db* という名前の既存の SQL Anywhere データベースから、*customer.udb* という名前の新しい Ultra Light データベースが作成されます。作成されたデータベース内のテーブルとインデックスは、Pub1 スキーマパブリケーションに含まれるものと同じです。Pub2 同期パブリケーションが Ultra Light データベースに作成されます。

```
ulinit -a DBF=MySource.db;UID=JimmyB;PWD=secret customer.udb -n Pub1 -s Pub2
```

Ultra Light データベースへの接続

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

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

参照

- 「Ultra Light 接続パラメータ」164 ページ
- Ultra Light C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- Ultra Light J : 「Ultra Light および Ultra Light Java Edition のデータベースの作成および接続方法」『Ultra Light Java プログラミング』

Ultra Light 接続文字列とパラメータ

「接続文字列」は、データベース接続を定義して確立できるように、アプリケーションから渡される一連の接続パラメータです。一部のパラメータは接続を開くために常に必要ですが、その他のパラメータは単一の接続のデータベース機能を調整するために使用されます。

接続文字列は、セミコロンで区切ったリスト内の *keyword=value* のペアとして定義されます。次に、データベースファイル名、ユーザ ID、パスワードを指定する接続文字列フラグメントの例を示します。

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

これらのパラメータをデータベースに指定する方法は、Ultra Light ユーティリティから接続するか、Ultra Light アプリケーションから接続するかによって異なります。

Ultra Light のコマンドラインユーティリティでは、データベースへの接続が必要な場合に接続文字列が使用されます。

Ultra Light アプリケーションは、保管されたファイルまたはアプリケーションコード内から接続パラメータ値を読み取る Ultra Light API を使用して開発できます。ユーザ認証が不要な場合は固定接続文字列を提供できます。または、接続時にパラメータ値を提供するようにユーザに要求します。

接続文字列は、アセンブルされると、Ultra Light ランタイムに渡されて処理されます。接続試行が検証されると、データベースへの接続が許可されます。データベースファイルが存在しないか、または認証に失敗すると、接続エラーが発生します。

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

Ultra Light 管理ツールの接続パラメータの優先度

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

- OS 固有のオプションは、OS 固有でないオプションよりも優先されます。たとえば、CE_DBF は Windows CE デバイス上で DBF よりも優先されます。
- CE_FILE、desktop、device、NT_FILE のいずれかのパラメータが指定されている場合は、常に DBF より優先されます。
- 接続文字列で重複するパラメータを指定すると、最後に指定したパラメータが使用されます。他のパラメータはすべて無視されます。
- 接続文字列のパラメータが、ULSQLCONNECT 環境変数または接続オブジェクトで指定されたパラメータより優先されます。
- UID と PWD の**両方の**値が接続文字列でも ULSQLCONNECT でも指定されていない場合は、デフォルトの **UID=DBA** と **PWD=sql** が想定されます。

制限事項

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

参照

- 「Ultra Light 接続パラメータ」164 ページ
- Ultra Light C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- Ultra Light J : 「Ultra Light および Ultra Light Java Edition のデータベースの作成および接続方法」『Ultra Light Java プログラミング』

例

接続文字列は、各パラメータ名と値をセミコロンで区切り、1 行で入力できます。

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

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

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

Ultra Light 接続パラメータと ULSQLCONNECT 環境変数

ULSQLCONNECT 環境変数を使用すると、開発中に同じ接続パラメータを Interactive SQL に指定する必要がなくなります。Interactive SQL は、ULSQLCONNECT 環境変数に対応する唯一のツールです。この変数は、カスタムアプリケーションには使用できません。

ULSQLCONNECT 環境変数はオプションで、デフォルトでは設定されません。ULSQLCONNECT 環境変数は、*keyword=value* をセミコロンで区切って定義したパラメータのリストとして設定します。

指定した値が、Ultra Light デスクトップ管理ツールのデフォルトの値になります。Interactive SQL に追加パラメータが必要な場合や、この環境変数で設定したデフォルト値を変更する必要がある場合は、これらの値を設定してください。ユーザが指定する値が常にこの環境変数より優先されます。

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

参照

- 「Ultra Light 管理ツールの接続パラメータの優先度」36 ページ

例

この例では、ULSQLCONNECT 環境変数を使用して *c:¥database¥myfile.udb* というファイルに接続し、**demo** パスワード **test** を使用してユーザを認証します。

ULSQCONNECT 環境変数に、次の変数を設定します。

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

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

接続パラメータでの Ultra Light ファイルパスの指定

デバイスの物理記憶領域によって、データベースをファイルとして保存するかどうか、およびデータベースを指定するときの命名規則が決定されます。

注意
Ultra Light エンジンを使用して、データベースへのマルチプロセスアクセスに対応する場合は、エンジンが別のロケーションから起動される可能性があるため、絶対ファイルパスを使用してください。

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

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

ヒント
Ultra Light 管理ツールを使用して、接続しているデバイスにすでに配備されているデータベースを管理できます。「Windows Mobile」38 ページを参照してください。

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

```
Connection = DatabaseManager::OpenConnection("UID=JDoe;PWD=ULdb;CE_DBF=¥database
¥MyCEDB.udb;NT_FILE=MyDB.udb")
```

デスクトップ

デスクトップでは、絶対パスまたは相対パスを使用できます。

Windows Mobile

Windows Mobile デバイスでは、絶対パスが必要です。

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

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

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

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

参照

- 「Ultra Light DBF 接続パラメータ」173 ページ
- 「Ultra Light NT_FILE 接続パラメータ」181 ページ
- 「Ultra Light CE_FILE 接続パラメータ」170 ページ

Ultra Light データベースのタスクと機能

データベースプロパティの読み取り

データベース作成パラメータに対応していないデータベースプロパティの設定を変更できます。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

各 Ultra Light API には、データベースプロパティにアクセスするためにアプリケーション内で使用できる GetDatabaseProperty メソッドが含まれています。

DB_PROPERTY SQL 関数を呼び出す方法でも、データベースプロパティにアクセスできます。

♦ タスク

1. Sybase Central を使用してデータベースに接続します。
2. データベースを右クリックし、**[プロパティ]** をクリックします。

結果

データベースプロパティは、**[データベースプロパティ]** ウィンドウの **[一般]** タブおよび **[同期情報]** タブに表示されます。**[同期情報]** タブでは、データベースプロパティはプロパティ名のアルファベット順で表示されます。データベースプロパティを値の順にソートするには、**[値]** カラムをクリックします。

参照

- 「Ultra Light データベースプロパティ」187 ページ
- ULConnection.GetDatabaseProperty メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.GetDatabasePropertyInt メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseSchema.GetDatabaseProperty メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- 「DB_PROPERTY 関数 [システム]」348 ページ

データベースオプションへのアクセス

データベースの動作を設定するデータベースオプションを表示し変更します。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

Sybase Central から、永続データベースオプションの設定を表示および変更できます。一時的に、Sybase Central から、Ultra Light データベースオプションを表示または設定**できない**場合があります。

データベースオプションは、いつでも設定または変更できます。一時的なデータベースオプションは、データベースの実行中のみ保持されます。

オプションの値は、SET OPTION SQL 文を使用して設定します。

♦ タスク

1. Sybase Central を使用してデータベースに接続します。
2. データベースを右クリックし、**[オプション]** をクリックします。
3. オプションを設定またはリセットするには、**[値]** フィールドに新しい値を入力します。
4. **[すぐに設定]** または **[すぐにリセット]** をクリックして変更をコミットします。

結果

データベースオプションの設定が変更され、保存されます。

参照

- 「Ultra Light データベースオプション」191 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- ULConnection.SetDatabaseOption メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.SetDatabaseOptionInt メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseSchema.SetDatabaseOption メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- 「データベースプロパティの読み取り」39 ページ

Ultra Light のイベント通知

Ultra Light でイベントと通知がサポートされるようになりました。通知は、イベントが発生したときに送信されるメッセージであり、パラメータに関する追加情報も示されます。Ultra Light にはシステムイベントがありますが、イベントにはユーザ定義のものもあります。

イベント通知によって、接続間または同じデータベースに接続されているアプリケーション間での調整および通知を行うことができます。通知はキューで管理されます。通知の管理に使用さ

れるキューは、接続のデフォルトキューまたは明示的に作成および命名されたキューです。イベントの発生時に、通知が登録されたキュー (または接続) に送信されます。

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

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

- Commit
- SyncComplete
- TableModified

ユーザイベントは、アプリケーションによって定義およびトリガすることもできます。

イベントと通知の API は、サポートされている各言語で提供されます。また、API 機能にアクセスするための SQL 関数も提供されています。

イベント

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

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

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

キューの操作

キューは、作成および破棄することができます。

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

DestroyNotificationQueue は、指定されたイベント通知キューを破棄します。キューに未読の通知がある場合は、警告が表示されます。未読通知は破棄されます。接続のデフォルトのイベントキューが作成されている場合、接続が閉じると破棄されます。

イベントの操作

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

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

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

通知の操作

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

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

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

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

その他の考慮事項

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

独立性レベル

独立性レベルは、あるトランザクションの操作が、同時に処理されている別のトランザクションの操作からどの程度参照できるかを定義します。Ultra Light は、接続に対してデフォルトの独立性レベル、read_committed を使用します。Ultra Light のデフォルトの独立性レベルでは、コミットされていないローを独立させることによって、データの整合性を保持します。

独立性レベル	特性
0 - read_uncommitted	●ダーティリード、繰り返し不可能読み出し、幻ローを許可する。 ●同時トランザクションがローを変更しないこと、またはローに対しての変更がロールバックされないことは保証されない。
1 - read_committed	●繰り返し不可能読み出しと幻ローを許可する。 ●ダーティリードを防ぐ。 ●トランザクション中にクエリの結果が変更されないという保証はない。

次のいずれかのアプローチを使用して、独立性レベルを read_committed から read_uncommitted に変更できます。

- SET OPTION SQL 文と isolation_level データベースオプションを使用します。

  たとえば次の文では、コミットされていない読み出しに独立性レベルが設定されます。

  ```
  SET OPTION isolation_level = 'READ_UNCOMMITTED'
  ```

- Ultra Light C++ API の場合は、ULConnection SetDatabaseOption メソッドを使用して独立性レベルを変更します。

  Ultra Light.NET API の場合は、ULConnection.BeginTransaction メソッドまたは ULDatabaseSchema.SetDatabaseOption メソッドを使用して、read_committed 独立性レベルでトランザクションを作成します。

  Ultra Light J API の場合は、Connection.setOption メソッドを使用します。

> **注意**
> Ultra Light Java Edition データベースでは、read_uncommitted の独立性レベルだけがサポートされます。 「BlackBerry スマートフォンでのデータ同期」『Ultra Light Java プログラミング』を参照してください。

Ultra Light Java Edition データベースでの同時実行性とロッキング

- **ロッキング** 2 つの異なる接続は、同一のローを同時に変更することはできません。2 つの接続で同じローを操作しようとすると、2 つ目の接続でエラーが返され、最初の接続が現在のトランザクションをコミットまたはロールバックするまでローを変更できなくなります。

- **可視性** データベース上で 1 つの接続が操作を行うと、その他の接続から即座に参照可能になります。

参照

- 「SET OPTION 文 [Ultra Light]」453 ページ
- 「Ultra Light isolation_level オプション」195 ページ
- ULConnection.SetDatabaseOption メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.BeginTransaction メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULDatabaseSchema.SetDatabaseOption メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』

read_uncommitted 独立性レベルの特性

Ultra Light を isolation_level 0 (read_uncommitted) で操作すると、次の影響が生じることがあります。

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

  ダーティリードの影響を示すチュートリアルについては、「チュートリアル：ダーティリードの知識」『SQL Anywhere サーバ SQL の使用法』を参照してください。

  幻ローを示すチュートリアルについては、「チュートリアル：幻ローの知識」『SQL Anywhere サーバ SQL の使用法』を参照してください。

- アプリケーションは、繰り返し不可能読み出しを実行できます。このシナリオでは、アプリケーションがデータベースからローを読み出し、他の操作の実行に移ります。そこへ、別のアプリケーションがローの更新や削除を行い、変更をコミットします。最初のアプリケーションが元のローを再度読み出すと、更新された情報が取得されるか、元のローが削除されたことを発見します。

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

参照

- ULConnection.SetDatabaseOption メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.BeginTransaction メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULDatabaseSchema.SetDatabaseOption メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- 「SET OPTION 文 [Ultra Light]」453 ページ
- 「Ultra Light isolation_level オプション」195 ページ

例

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

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

   注意
   ローを読み込んだりフェッチしたりしても、ローはロックされません。接続 A がローをフェッチしても変更はしない場合、接続 B はこのローを変更できます。

2. 接続 A は現在のローを変更するときに、バッファ内のコピーを変更します。接続 A が Update メソッドを呼び出すか結果セットを閉じると、バッファ内のコピーはデータベースに書き戻されます。

3. このローに対して書き込みロックが掛けられ、他のトランザクションがこのローを変更できなくなります。この変更は、接続 A がコミットを実行するまで、コミットされないままになります。

4. 変更によっては、接続 B が現在のローをフェッチすると、次の現象が発生する可能性があります。

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

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

Ultra Light データベースの検証

Sybase Central の**データベース検証ウィザード**や、Ultra Light データベース検証ユーティリティ (dbvalid)、または Ultra Light API の ValidateDatabase メソッドなどのツールを使用して、データベースが有効であることを定期的に確認してください。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

データベースファイルの破損は、データベースサーバがデータベース内の破損部分にアクセスするまで判明しないことがあります。

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

Ultra Light データベースは、次のいずれかの方法で検証できます。

- Sybase Central の**データベース検証ウィザード**
- ulvalid コマンドラインユーティリティ
- Ultra Light API の ValidateDatabase メソッド

Ultra Light Java Edition データベースの検証はサポートされていません。

♦ タスク

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

結果

データベースが検証されます。

参照

- ULConnection.ValidateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.ValidateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.validateDatabase メソッド [Android] [Ultra Light J]『Ultra Light Java プログラミング』
- 「Ultra Light データベース検証ユーティリティ (ulvalid)」235 ページ

Ultra Light および Ultra Light Java Edition データベースのバックアップとリカバリ

Ultra Light または Ultra Light Java Edition データベースを使用しているアプリケーションが予期せずに停止した場合、アプリケーションを再起動すると、データベースは自動的に一貫性が保持された状態にリカバリされます。予期せぬ障害が発生する前にメモリにフラッシュされたすべ

てのコミット済みトランザクションは、データベースに保持されます。障害発生時にフラッシュされていないすべてのトランザクションは、ロールバックされます。

Ultra Light データベースでは、リカバリの実行にトランザクションログを使用しません。各ローのステータス情報を保存して、リカバリ時のローの処理を決定します。

バックアップ

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

参照

- 「Ultra Light データベースのローのステータス管理」476 ページ
- 「単一のトランザクションまたはグループ化されたトランザクションのフラッシュ」478 ページ
- 「Mobile Link クライアントとしての Ultra Light」69 ページ

Ultra Light データベースのスキーマ

データベースの論理的なフレームワークを「スキーマ」といいます。

Ultra Light データベースのスキーマ

Ultra Llight データベースのスキーマを更新するには、適切なデータ定義言語 (DDL) 文を使用します。または、ALTER DATABASE SCHEMA FROM FILE 文を使用し、SQL スクリプトを使用してスキーマ定義を変更します。

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

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

DDL 文を実行すると、DDL 文によるスキーマの変更が完了するまで、その他のデータベースの操作はブロックされます。

Ultra Light Java Edition データベースのスキーマ

Ultra Light Java Edition データベースのスキーマは、Ultra Light Java Edition データベースのメタデータを含むシステムテーブルのカタログとして管理されます。システムテーブルのメタデータには、以下が含まれています。

- **テーブル定義。** systable システムテーブルに格納されています。
- **カラム定義。** syscolumn システムテーブルに格納されています。
- **インデックス定義。** sysindex および sysindexcolumn システムテーブルに格納されています。
- **パブリケーション定義。** syspublications および sysarticles システムテーブルに格納されています。
- **外部キー定義。** sysforeignkey および sysfkcol システムテーブルに格納されています。
- **ユーザ名とパスワード。** sysuldata システムテーブルに格納されています。

参照

- 「ALTER DATABASE SCHEMA FROM FILE 文 [Ultra Light]」418 ページ
- 「Ultra Light データベーススキーマのアップグレードの配備」122 ページ
- 「systable システムテーブル」266 ページ
- 「syscolumn システムテーブル」262 ページ
- 「sysindex システムテーブル」264 ページ
- 「sysindexcolumn システムテーブル」265 ページ
- 「syspublications システムテーブル」266 ページ
- 「sysarticles システムテーブル」262 ページ
- 「sysforeignkey システムテーブル」263 ページ
- 「sysfkcol システムテーブル」264 ページ
- 「sysuldata システムテーブル」267 ページ

Ultra Light のテーブルとカラム

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

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

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

Ultra Light では、ベーステーブルだけを作成できます。ベーステーブルは、永続的なデータを保管するテーブルです。Ultra Light では、グローバルテンポラリテーブルや宣言されたテンポラリテーブルはサポートされていません。

ローのパックとテーブル定義

Ultra Light では、次の 2 つの形式のローを扱います。

- **アンパックされたロー**　非圧縮形式です。個々のカラム値を読み込んだり書き込んだりするには、その前に、該当するローをアンパックする必要があります。

- **パックされたロー**　アンパックされたローの圧縮形式です。各カラム値が圧縮され、ロー全体に必要なメモリが最小限に抑えられています。パックされたローのサイズは、完全に各カラムの値によって決まります。たとえば、2 つのローが同じテーブルに属すことは可能ですが、それぞれがパックされたサイズは大きく異なることがあります。LONG BINARY カラムと LONG VARCHAR カラムは、パックされたローとは別に格納されます。

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

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

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

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

参照

- 「Ultra Light page_size 作成パラメータ」 151 ページ
- 「データベーステーブル」『SQL Anywhere 16 紹介』
- 「データベースの作成」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light のシステムテーブル」 237 ページ

Ultra Light のテーブルの作成

永続的なリレーショナルデータを保持するためにベーステーブルを作成できます。

前提条件

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

♦ タスク

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

結果

テーブルが作成されます。テーブルとそのデータは、それらを明示的に削除しないかぎり維持されます。

次の手順

カラムを追加するか、またはインデックスを作成します。

参照

- 「CREATE TABLE 文 [Ultra Light]」432 ページ
- 「Ultra Light テーブルへのカラムの追加」52 ページ

Ultra Light テーブルへのカラムの追加

カラムの作成後、そのカラムを Ultra Light テーブルに追加できます。

前提条件

テーブルにすでにデータが格納されている場合、カラムを追加できるのは、そのカラムの定義にデフォルト値が含まれるか、そのカラムに NULL 値が許可される場合だけです。

♦ タスク

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

結果

カラムがテーブルに追加されます。

参照

- 「オブジェクト名の考慮事項」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light SQL データ型」293 ページ
- 「カラムデータ型の考慮事項」『SQL Anywhere サーバ データベース管理』
- 「CREATE TABLE 文 [Ultra Light]」432 ページ
- 「ALTER TABLE 文 [Ultra Light]」422 ページ

Ultra Light のカラム定義の変更

テーブルのカラム定義の構造を変更するには、さまざまなカラムの属性を変更するか、またはカラム全体を削除します。

前提条件

変更したカラム定義は、カラムにすでに格納されているデータの条件を満たす必要があります。たとえば、カラムにすでに NULL のエントリがある場合は、カラムを変更して NULL を不許可にすることはできません。

♦ タスク

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

結果

新しいカラム属性でテーブルが保存されます。

参照

- 「オブジェクト名の考慮事項」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light SQL データ型」293 ページ
- 「カラムデータ型の考慮事項」『SQL Anywhere サーバ データベース管理』
- 「ALTER TABLE 文 [Ultra Light]」422 ページ

Ultra Light テーブルの削除

必要がなくなったテーブルは削除できます。

前提条件

次の条件を満たすテーブルを削除できます。

●パブリケーションでアーティクルとして使用されていない

●別のテーブルの外部キーで参照されているカラムがない

上記以外の場合は、パブリケーションを変更するか、外部キーを**削除してから**、テーブルを削除します。

♦ タスク

1. Sybase Central を使用して Ultra Light データベースに接続します。
2. 左ウィンドウ枠で、**[テーブル]** をダブルクリックします。

3. テーブルを右クリックして、**[削除]** をクリックします。

4. **[はい]** をクリックします。

結果

テーブルが削除されます。

参照

- 「DROP TABLE 文 [Ultra Light]」442 ページ

Ultra Light テーブル内の情報のブラウズ

Ultra Light データベースのテーブル内のデータを表示します。

前提条件

データベースが接続され、選択されている必要があります。

内容と備考

テーブルには、ユーザテーブルとシステムテーブルがあります。現在のデータベースの表示でシステムテーブルの表示と非表示を切り替えてテーブルをフィルタできます。Ultra Light には所有権の概念がないので、すべてのユーザがすべてのテーブルをブラウズできます。

♦ タスク

1. Sybase Central を使用して Ultra Light データベースに接続します。

2. テーブルのリストを表示するには、**[テーブル]** をダブルクリックします。

3. テーブルデータを表示するには、テーブルを右クリックして、右ウィンドウ枠の **[データ]** タブをクリックします。

結果

テーブルとデータが表示されます。

参照

- 「Ultra Light のシステムテーブル」237 ページ

Ultra Light データベースからのデータのコピーアンドペースト

Sybase Central では、コピーアンドペーストおよびドラッグアンドドロップを使用できます。したがって、データベース内または複数のデータベース間でオブジェクトを共有または移動できま

す。コピーアンドペーストまたはドラッグアンドドロップで、次の表に示すデータを共有できます。

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

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

Sybase Central

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

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

Interactive SQL

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

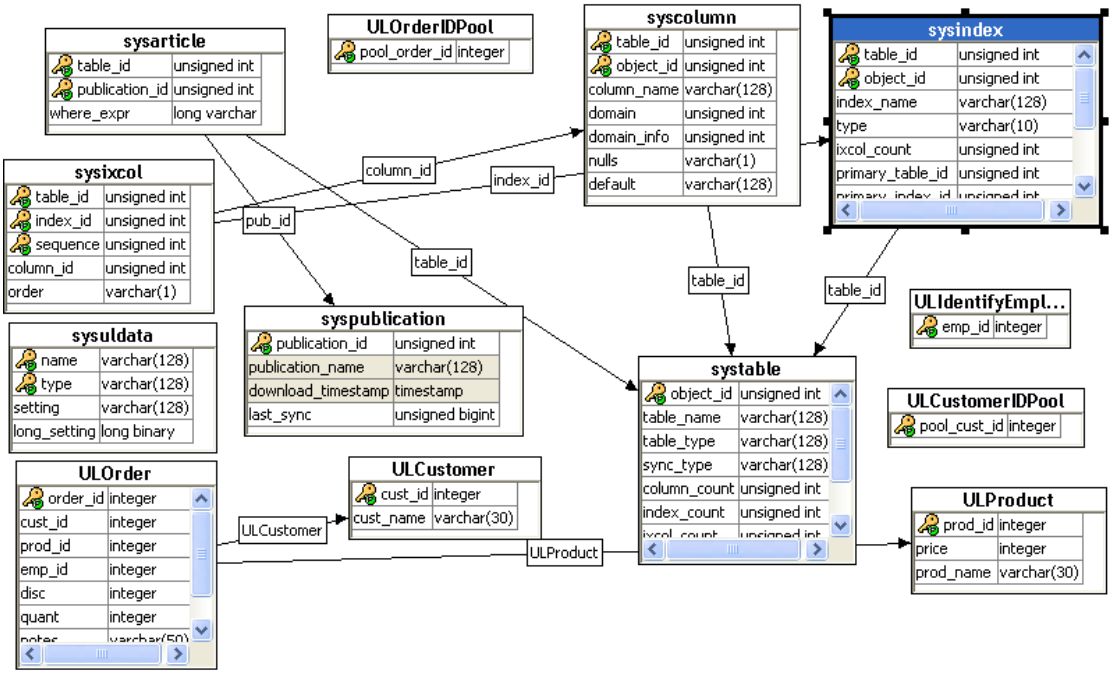
- 結果を指定のオブジェクトにコピーするには、SELECT 文を使用します。
- データベース内の別の場所にある 1 つのローまたは複数のローをテーブルに挿入するには、INSERT 文を使用します。

参照

- 「INSERT 文 [Ultra Light]」445 ページ
- 「SELECT 文 [Ultra Light]」451 ページ

Ultra Light プラグインからの ER 図の表示

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



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

参照

- 「データベースの作成」『SQL Anywhere サーバ データベース管理』

Ultra Light インデックス

インデックスとは、1 つ以上のテーブルカラムに含まれているデータ値の順序に基づく、テーブル内のローのポインタセットです。インデックスが作成されると、データベースオブジェクトとして Ultra Light によって自動的に保持されます。Ultra Light は、クエリを最適化するときに、既存のインデックスをスキャンして、クエリで指定されているテーブルにインデックスがあるかどうかを確認します。インデックスによってより短時間でローを返すことができる場合は、インデックスが使用されます。アプリケーションで Ultra Light のテーブル API を使用している場合は、ローをスキャンする順序を決定するインデックスを指定できます。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

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

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

Ultra Light は、次のインデックスをサポートしています。インデックスは、シングルカラムまたはマルチカラム (複合インデックス) で作成できます。LONG VARCHAR カラムや LONG BINARY カラムをインデックス付けすることはできません。

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

[1] 外部キーは、プライマリキーまたはユニークキーを参照できます。

[2] 一意性制約と同じです。

複合インデックスについて

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

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

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

参照

- 「インデックススキャンの作成と保持」463 ページ
- 「Ultra Light の実行プラン」469 ページ
- 「複合インデックス」『SQL Anywhere サーバ SQL の使用法』
- Ultra Light.NET : 「ULTable クラスを使用したデータの作成と修正」『Ultra Light .NET プログラミング』
- ULCommand.Prepare メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- UltraLite for C++: 「ULTable クラスを使用したデータの作成と修正」『Ultra Light C/C++ プログラミング』
- ULPreparedStatement クラス [Ultra Light C++]『Ultra Light C/C++ プログラミング』

インデックスを使用する場合

インデックスは次の場合に使用します。

- **Ultra Light で参照整合性を保つ必要がある場合**　インデックスは Ultra Light に、テーブル内のローに一意性制約を強制するための手段も提供します。よく似ているデータにインデックスを追加する必要はありません。

- **特定のクエリのパフォーマンスがアプリケーションにとって重要である場合**　インデックスによってクエリのパフォーマンスが向上し、そのクエリのパフォーマンスがアプリケーションにとって重要であり、そのクエリを頻繁に使用する場合は、インデックスを使用します。テーブルが非常に小さい場合を除き、インデックスによって検索のパフォーマンスが大幅に向上します。データを頻繁に検索する場合は一般にインデックスを使用することをおすすめします。

- **複雑なクエリがある場合**　複雑なクエリ (たとえば、JOIN 句、GROUP BY 句、ORDER BY 句を使用するもの) は、インデックスを使用するとパフォーマンスが大幅に向上する可能性があります。ただし、パフォーマンスの向上の程度を確認するのは困難である場合があります。したがって、インデックスを使用した場合と使用しなかった場合でクエリをテストし、パフォーマンスを確認することをおすすめします。

- **Ultra Light テーブルのサイズが大きい場合**　ローを検索する平均時間は、テーブルのサイズに比例して長くなります。したがって、非常に大きなテーブルの検索効率を向上させるには、インデックスの使用を検討してください。インデックスを使用すると、インデックスが付いているカラムについては、Ultra Light でローが短時間で見つかります。インデックスがない場合、Ultra Light ではテーブル内のすべてのローを検索し、ローが探索条件を満たすかどうかを確認する必要があるので、大きなテーブルの場合は時間がかかります。

- **Ultra Light クライアントアプリケーションで大量の挿入、更新、または削除の操作を行わない場合**　Ultra Light ではインデックスがデータ自体とともに保持されるので、これらの操作では、インデックスによってパフォーマンスが下がります。したがって、インデックスの使用は、前述のように頻繁に問い合わせるデータだけに制限してください。Ultra Light のデフォルトのインデックス (プライマリキーと一意性制約のインデックス) で十分である可能性があります。

- **WHERE 句か ORDER BY 句またはその両方を使用する場合**　これらの句の対象となるカラムにインデックスを使用すると、これらの句の評価を高速化できます。特にインデックスに

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

よってマルチカラムの ORDER BY 句を最適化できます。ただし、インデックスと ORDER BY 句でのカラムの配置がまったく同じである必要があります。

インデックスのタイプ

Ultra Light は、さまざまなタイプのインデックスをサポートしています。具体的には、ユニークキー、ユニークインデックス、ユニークでないインデックスをサポートしています。これらのタイプでは、インデックスで許可されることが異なります。

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

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

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

参照

- 「Ultra Light インデックスの追加」59 ページ

Ultra Light インデックスの追加

データベースにインデックスを追加すると、検索プロセスが高速になります。

前提条件

データベースが接続されている必要があります。

内容と備考

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

♦ タスク

1. Sybase Central を使用して Ultra Light データベースに接続します。

2. **[インデックス]** を右クリックし、**[新規]** » **[インデックス]** を選択します。
3. ウィザードの指示に従います。

結果

インデックスが作成されます。

例

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

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

設定したデフォルトの最大ハッシュサイズでインデックスが作成されます。デフォルト以外の値を使用するインデックスを作成するには、WITH MAX HASH SIZE *value* 句を使用してこのインデックスインスタンスの新しい値を設定します。

参照

- 「CREATE INDEX 文 [Ultra Light]」428 ページ

Ultra Light インデックスの削除

インデックスを削除すると、データベースから削除されます。

前提条件

この作業を実行するための前提条件はありません。

♦ タスク

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

結果

インデックスはデータベースから削除されます。

参照

- 「DROP INDEX 文 [Ultra Light]」440 ページ

Ultra Light ユーザ

一般的な Ultra Light データベースには、1 つのユーザ ID とパスワードが含まれます。Ultra Light データベースは、特に指定しなければ、**DBA** のデフォルトのユーザ ID と **sql** のデフォルトのパスワードを使用して作成されます。Ultra Light Java Edition データベースは、**DBA** のデフォルトのパスワードを使用して作成されます。

ユーザスキーマの変更はオプションで、「必須ではありません」。ほとんどのアプリケーションは、データベースレベルの認証は不要で、アプリケーションとそのデータへのアクセスにはデバイスレベルのパスワード認証で十分であるとみなされます。

ユーザを認証しない主な理由としては、配備先が単一ユーザのデバイスである、ユーザがアプリケーションを起動するたびにプロンプトを表示するのは不便である、などがあります。

データベースレベルの認証が不要であれば、データベース接続文字列にユーザ ID またはパスワードを含める必要はありません。最も単純な Ultra Light 接続文字列は、**DBF=***ファイル名*です。以降では、Ultra Light でのユーザ ID の実装方法と、明示的なユーザ認証が必要な場合のユーザ ID の使用方法について説明します。

カスタムユーザ認証インタフェースを使用して Ultra Light アプリケーションを開発する場合、Ultra Light データベースに格納されている Ultra Light ユーザ ID とパスワードのハッシュを効果的に使用して、ユーザが提供したクレデンシャルを検証できます。独自のパスワードハッシュアルゴリズムを作成する必要はありません。Ultra Light データベースにユーザを追加すると、そのユーザのユーザ ID とパスワードのハッシュが格納されます。さらに、UID と PWD の接続パラメータを使用してデータベースに接続を試みることによって、アプリケーションでユーザ提供のクレデンシャルを検証できます。**UID=***ユーザ名*、**PWD=***パスワード*です。Ultra Light データベースに正常に接続できれば、そのユーザは認証されたことを意味します。

警告

SQL Anywhere のユーザとは異なり、Ultra Light ユーザの作成と管理は、オブジェクト所有権や特定のデータベースのロールと権限のためではなく、認証のみを目的としたものです。認証されたユーザは、データベースへの完全なアクセス権を取得します。

ユーザ ID とパスワードを作成すると、Ultra Light データベースへの接続を制御できますが、データベースファイル内のデータの機密性が保護されるわけではありません。コンテンツはプレーンテキストとして格納されるので、直接読み取ることができます。

データベースのコンテンツの機密性を保護するには、ファイルを暗号化することを推奨します。ファイルを暗号化する場合は、ユーザ ID とパスワードの代わりに暗号化キーを使用してユーザを認証できます。

データがプレーンテキストとして格納されないように、ファイルを難読化して格納することもできますが、このアプローチでもデータの機密性が完全に保護されるわけではありません。

詳細については、「データベースセキュリティ」29 ページと「Ultra Light DBKEY 接続パラメータ」175 ページを参照してください。

注意

Ultra Light ユーザ ID は、Mobile Link ユーザ名とは異なります。

制限事項

ユーザ ID とパスワードは、Ultra Light Java Edition データベースには格納されません。これらのデータベースは単一パスワードによって保護されるので、ユーザの作成または認証は不要です。詳細については、「Ultra Light および Ultra Light Java Edition のデータベースの作成および接続方法」『Ultra Light Java プログラミング』を参照してください。

Ultra Light ユーザ ID には、次の制限事項が適用されます。

- Ultra Light では、データベースごとに最大 4 つのユニークなユーザ ID がサポートされます。
- ユーザ ID とパスワードは、Sybase Central、SQL 文、またはアプリケーションの Ultra Light API メソッドを使用して変更できます。
- ユーザ ID の長さは 31 文字までに制限されます。
- ユーザ ID に、先頭の一重引用符 (')、先頭の二重引用符 (")、またはセミコロン (;) を含めることはできません。
- ユーザ ID は常に大文字と小文字が区別されません。パスワードは常に大文字と小文字が区別されます。
- ユーザ ID の名前を変更することはできません。新しいユーザ ID を追加して、既存のユーザ ID を既存のデータベース接続から削除する必要があります。
- Ultra Light API を使用してユーザをプログラムによってリストすることはできません。データベースの既存ユーザをリストするには、データベースツールを使用する必要があります。
- Ultra Light データベースに初めて接続するときの **UID** と **PWD** は、データベースの作成時に設定した値です。これらの接続パラメータが指定されていない場合、Ultra Light は、**DBA** のユーザ ID と **sql** のパスワードを使用して接続を試みます。データベースの作成時にユーザ名とパスワードを明示的に設定しなかった場合は、データベースに接続するときにユーザ名とパスワードを入力する必要はありません。

参照

- 「Ultra Light UID 接続パラメータ」186 ページ
- 「Ultra Light PWD 接続パラメータ」182 ページ

Ultra Light ユーザの管理に使用する接続パラメータ

UID と **PWD** の接続パラメータを使用して、Ultra Light データベースでユーザを作成または認証することができます。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

注意
接続パラメータの代わりに、アプリケーション内の次の Ultra Light API メソッドを使用して、Ultra Light データベースへのユーザのアクセス権を付与したり、取り消したりすることができます。

- ULConnection.GrantConnectTo メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.RevokeConnectFrom メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULGrantConnectTo メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULRevokeConnectFrom メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.GrantConnectTo メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULConnection.RevokeConnectFrom メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

付与と取り消しのメソッドは、Ultra Light J API では使用できません。Android スマートフォンの場合は、ConfigPersistent.setConnectionString メソッドを使用して、UID と PWD の接続パラメータを設定します。

Ultra Light データベースでの接続パラメータの使用

ほとんどの Ultra Light API では、DatabaseManager オブジェクトの createDatabase メソッドを使用して、接続パラメータと作成パラメータを指定し、新しい Ultra Light データベースを作成できます。

次に、Ultra Light C++ API の CreateDatabase メソッドに **UID** と **PWD** のパラメータを渡して、新しい Ultra Light データベースのデフォルトユーザを作成する例を示します。

```
ULConnection * conn;
ULError ulerr;
ULDatabaseManager::CreateDatabase("dbf=sample.udb;uid=default-name;pwd=default-password",
&ulerr);
```

次に、Ultra Light C++ API の OpenConnection メソッドに **UID** と **PWD** のパラメータを渡して、既存の Ultra Light データベースでユーザを認証する例を示します。

```
ULConnection * conn;
ULError ulerr;
ULDatabaseManager::OpenConnection("dbf=sample.udb;uid=test-name;pwd=test-password", &ulerr);
```

Android スマートフォンの Ultra Light データベースでの接続パラメータの使用

Ultra Light J API の場合、Ultra Light J API の Configuration オブジェクトの setConnectionString メソッドを使用して、ユーザを作成または認証できます。

次に、Ultra Light J API の createDatabase メソッドに **UID** と **PWD** のパラメータを渡して、新しい Ultra Light データベースのデフォルトユーザを作成する例を示します。

```
ConfigFile config =
   DatabaseManager.createConfigurationFileAndroid("DBname.udb", getApplicationContext());
config.setConnectionString("uid=default-name;pwd=default-password");
Connection conn = DatabaseManager.createDatabase(config);
```

次に、Ultra Light J API の connect メソッドに **UID** と **PWD** のパラメータを渡して、既存の Ultra Light データベースでユーザを認証する例を示します。

```
ConfigFile config =
   DatabaseManager.createConfigurationFileAndroid("DBname.udb", getApplicationContext());
config.setConnectionString("uid=test-name;pwd=test-password");
Connection conn = DatabaseManager.connect(config);
```

setConnectionString メソッドの代わりに、setPassword または setUserName メソッドを使用して、ユーザを作成または認証することもできます。

Ultra Light Java Edition データベースでのユーザの作成または認証

Ultra Light Java Edition データベースは、ユーザ認証をサポートしていません。代わりに、Configuration.setPassword メソッドを使用してデータベースのパスワードを設定します。

次に、既存の Ultra Light Java Edition データベースにパスワードを設定する例を示します。

```
ConfigNonPersistent config =
   DatabaseManager.createConfigurationNonPersistent("DBname.ulj");
config.setPassword("my_password");
Connection conn = DatabaseManager.connect(config);
```

参照

- ConfigPersistent.setUserName メソッド [Android] [Ultra Light J]『Ultra Light Java プログラミング』
- Configuration.setPassword メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- 「Ultra Light UID 接続パラメータ」186 ページ
- 「Ultra Light PWD 接続パラメータ」182 ページ
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ConfigPersistent.setConnectionString メソッド [Android] [Ultra Light J]『Ultra Light Java プログラミング』
- DatabaseManager.createDatabase メソッド [Ultra Light J]『Ultra Light Java プログラミング』

SQL 文を使用した Ultra Light ユーザの管理

CREATE USER、ALTER USER、DROP USER の SQL 文を使用して、Ultra Light データベースのユーザを管理できます。

これらの SQL 文は、Ultra Light Java Edition データベースでは使用できません。

注意
SQL 文の代わりに、アプリケーション内の次の Ultra Light API メソッドを使用して、Ultra Light データベースへのユーザのアクセス権を付与したり、取り消したりすることができます。

- ULConnection.GrantConnectTo メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.RevokeConnectFrom メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULGrantConnectTo メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULRevokeConnectFrom メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.GrantConnectTo メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULConnection.RevokeConnectFrom メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

付与と取り消しのメソッドは、Ultra Light J API では使用できません。Android スマートフォンの場合は、CREATE USER、ALTER USER、または DROP USER の SQL 文を文字列変数として構成し、これを Connection.prepareStatement メソッドに渡します。

例

次に、Ultra Light J API を使用して Android スマートフォンの既存の Ultra Light データベースに接続し、CREATE USER 文を使用して新しいユーザを作成する例を示します。

```
ConfigFile config =
   DatabaseManager.createConfigurationFileAndroid("DBname.udb", getApplicationContext());
Connection conn = DatabaseManager.connect(config);

String sql_string = "CREATE USER test-user IDENTIFIED BY test-password";
PreparedStatement authenticator = conn.prepareStatement(sql_string);
authenticator.execute();
authenticator.close();
```

参照

- 「CREATE USER 文 [Ultra Light]」439 ページ
- 「ALTER USER 文 [Ultra Light]」425 ページ
- 「DROP USER 文 [Ultra Light]」443 ページ
- ULPreparedStatement クラス [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.PrepareStatement メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULCommand クラス [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULCommand.Prepare メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- PreparedStatement インタフェース [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.prepareStatement メソッド [Ultra Light J]『Ultra Light Java プログラミング』

Sybase Central を使用した Ultra Light ユーザの作成

Sybase Central を使用して Ultra Light データベースのユーザを作成できます。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

Sybase Central は、Ultra Light Java Edition データベースとは互換性がありません。

注意
Sybase Central の代わりに、アプリケーション内の次の Ultra Light API メソッドを使用して、Ultra Light データベースへのユーザのアクセス権を付与できます。

- ULConnection.GrantConnectTo メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULGrantConnectTo メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.GrantConnectTo メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

付与のメソッドは、Ultra Light J API では使用できません。Android スマートフォンの場合は、CREATE USER 文を文字列変数として構成し、これを Connection.prepareStatement メソッドに渡します。

♦ タスク

1. Sybase Central を使用して Ultra Light データベースに接続します。
2. **[ユーザ]** フォルダを右クリックし、**[新規]** » **[ユーザ]** をクリックします。
3. ウィザードの指示に従います。

結果

新しいユーザが作成されます。

参照

- 「CREATE USER 文 [Ultra Light]」439 ページ

Sybase Central を使用した Ultra Light ユーザの削除

Sybase Central を使用して Ultra Light データベースのユーザを明示的に削除できます。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

Sybase Central は、Ultra Light Java Edition データベースとは互換性がありません。

注意
Sybase Central の代わりに、アプリケーション内の次の Ultra Light API メソッドを使用して、Ultra Light データベースへのユーザのアクセス権を拒否できます。

- ULConnection.RevokeConnectFrom メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULRevokeConnectFrom メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.RevokeConnectFrom メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

拒否のメソッドは、Ultra Light J API では使用できません。DROP USER 文を文字列変数として構成し、これを Connection.prepareStatement メソッドに渡します。

♦ タスク

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

結果

ユーザがデータベースから削除されます。

参照

- 「DROP USER 文 [Ultra Light]」443 ページ

Mobile Link クライアントとしての Ultra Light

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

Ultra Light クライアント

Ultra Light データベースを同期するには、アプリケーションで、Mobile Link サーバのアドレスおよび他の必要な情報を示す同期パラメータを設定することと、同期機能を呼び出すか、または SYNCHRONIZE SQL 文を実行することが必要となります。選択するオプションは、使用している API によって異なります。

Ultra Light クライアントの同期の動作のカスタマイズ

Ultra Light にカスタム同期のサポートを追加するには、次に示すいくつかの作業を行います。

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

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

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

- **統合データベースにアップロードするデータサブセットを指定する。** 省略可。デフォルトですべてのデータを同期しない場合にのみ行います。同期するデータを指定するには、1 つまたは複数のサブセット方法を使用します。

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

- **Ultra Light アプリケーションから同期を初期化し、セッションに関するパラメータを指定する。** 必須。同期のプログラミングは 2 段階で行います。まずセッションを設定してから、同期操作を初期化します。

セッションの設定を行うには、同期の通信ストリーム (ネットワークプロトコル) とそのストリームのパラメータを選択し、同期スクリプトのバージョンを設定し、Mobile Link ユーザを識別します。ただし、ほかにも設定可能なパラメータがあります。たとえば、デフォルトの双方向同期を一方向のみに変更するには、upload_only と download_only のパラメータを使用します。

その他の重要な同期の動作は、Mobile Link サーバで Mobile Link 同期スクリプトを使用して制御します。これらのコードを以下に示します。

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

自分専用の同期スクリプトを作成すれば、リモートデータベース間でデータを適切に分割できます。

参照

- 「ユニークなプライマリキー」『Mobile Link サーバ管理』
- 「Ultra Light でのプライマリキーの一意性」 70 ページ
- 「Oracle 統合データベース」『Mobile Link サーバ管理』
- 「Ultra Light precision 作成パラメータ」 153 ページ
- 「Ultra Light クライアントの同期設計」 74 ページ
- 「Ultra Light アプリケーションの同期の設定」 80 ページ
- 「Mobile Link 統合データベース」『Mobile Link サーバ管理』
- 「同期スクリプトの作成」『Mobile Link サーバ管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバ管理』
- 「リモートデータベース間でローを分割する」『Mobile Link サーバ管理』

Ultra Light でのプライマリキーの一意性

Ultra Light では、Mobile Link でサポートされている任意の方法でプライマリキーの一意性を維持できます。

方法の一つとして、GLOBAL AUTOINCREMENT カラムを使用できます。GLOBAL AUTOINCREMENT は AUTOINCREMENT と同じですが、ドメインが分割されます。Ultra Light では、データベースのグローバル ID に割り当てられた分割からのみカラムの値が設定されます。各 Ultra Light データベースには、ユニークな整数のグローバルデータベース ID が割り当てられます。

もう 1 つの方法として、UUID プライマリキーカラムを使用できます。UUID では、必要なデータの量は増えますが、別個のデータベース識別子は不要です。

参照

- 「ユニークなプライマリキー」『Mobile Link サーバ管理』
- 「Ultra Light global_database_id オプション」 194 ページ

Ultra Light の GLOBAL AUTOINCREMENT カラム

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

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

global_database_id データベースオプションを使用すると、Ultra Light データベースに値を設定できます。Ultra Light を配備するときには、各データベースに対して必ず異なる ID 番号を割り当てる必要があります。

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

Ultra Light は、次の規則に従います。

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

- カラムに現在の分割の値が含まれていても、そのすべてが $p(n + 1)$ 未満であれば、この範囲内でこれまで使用した最大値より 1 大きい値が次のデフォルト値になる。

- デフォルトのカラム値は、現在の分割以外のカラムの値の影響を受けない。つまり、$pn + 1$ より小さいか $p(n + 1)$ より大きい数には影響されない。Mobile Link 同期を介して別のデータベースからレプリケートされた場合に、このような値が存在する可能性があります。

  たとえば、Ultra Light データベースにグローバル ID 番号 1 を割り当て、分割サイズが 1000 の場合、データベースのデフォルト値は 1001 ～ 2000 の範囲から選択されます。このデータベースの別のコピーで、ID 番号 2 が割り当てられたデータベースからは、2001 ～ 3000 の範囲で同一カラムのデフォルト値が指定されます。

- グローバル ID 番号は負の値には設定できないので、GLOBAL AUTOINCREMENT カラムの値は常に正の数です。ID 番号の最大値を制限するのは、カラムのデータ型と分割サイズだけです。

- グローバル ID の値を設定しないか、分割内の値がなくなった場合は、カラムに NULL 値が挿入されます。NULL 値が許可されていない場合にローを挿入しようとすると、エラーが発生します。

GLOBAL AUTOINCREMENT と宣言されたカラムに使用可能な値がなくなったか、なくなりそうになったら、新しいグローバルデータベース ID を設定する必要があります。GLOBAL AUTOINCREMENT の値は、グローバル ID 番号によって識別される分割から、最大値に達するまで選択されます。値を超えると、NULL 値が生成され始めます。新しいグローバルデータベー

ス ID 番号を割り当てることで、Ultra Light で別の分割から適切な値を設定できるようになります。

新しいグローバルデータベース ID を選択する方法の 1 つとして、未使用のグローバルデータベース ID の値のプールを管理できます。このプールは、プライマリキープールと同じ方法で管理されます。

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

参照

- 「Ultra Light global_database_id オプション」194 ページ
- 「プライマリキープール」『SQL Remote』
- 「分割サイズ」73 ページ
- 「Ultra Light global_database_id オプション」194 ページ
- ULCommand.Connection プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULConnection.Synchronize メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULSetDatabaseID メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.GlobalAutoIncrementUsage プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULGlobalAutoincUsage メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- Connection.setDatabaseId メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.getDatabaseId メソッド [BlackBerry] [Ultra Light J]『Ultra Light Java プログラミング』

最後に割り当てられた GLOBAL AUTOINCREMENT 値の調べ方

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

- **Ultra Light for C/C++**　ULConnection オブジェクトの GetLastIdentity 関数を使用します。
- **Ultra Light.NET**　ULConnection クラスの LastIdentity プロパティを使用します。
- **API Ultra Light J**　Connection インタフェースの GetLastIdentity メソッドを使用します。

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

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

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

参照

- ULConnection.GetLastIdentity メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.LastIdentity プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.getLastIdentity メソッド [Ultra Light J]『Ultra Light Java プログラミング』

分割サイズ

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

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

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

参照

- 「CREATE TABLE 文 [Ultra Light]」432 ページ
- 「ALTER TABLE 文 [Ultra Light]」422 ページ
- ULTableSchema.GetColumnPartitionSize メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULTableSchema.GetGlobalAutoincPartitionSize メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULGlobalAutoincUsage メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- Connection.getLastIdentity メソッド [Ultra Light J]『Ultra Light Java プログラミング』

GLOBAL AUTOINCREMENT カラムの分割サイズの変更

GLOBAL AUTOINCREMENT カラムの分割サイズを大きくすると、分割内のサイズの不足がほとんどなくなります。

前提条件

Ultra Light データベースに接続されている必要があります。

♦ **タスク**

1. カッコ内に分割サイズを指定して、GLOBAL AUTOINCREMENT カラムを持つテーブルを作成します。

   次の SQL コードを実行します。

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

   2 つのカラム–顧客 ID 番号を保持する整数カラムと顧客名を保持する文字列カラム) を持つ単純な参照テーブルが作成されます。ID の分割サイズは 5000 です。

2. Sybase Central で Ultra Light データベースに接続します。

3. 顧客テーブルの ID カラムを右クリックして、**[プロパティ]** をクリックします。

4. **[値]** タブをクリックします。

5. **[分割サイズ]** フィールドに、5000 を超える正の整数を入力します。

結果

[分割サイズ] フィールドに入力した値に応じて、ID カラムの分割サイズが更新されます。

Ultra Light クライアントの同期設計

Ultra Light データベース内のすべてのデータは、デフォルトで同期されます。Mobile Link リモートデータベースとして Ultra Light を初めて配備する場合、最初はデフォルトの動作を使用することを検討します。

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

設計時の確認事項	「はい」の場合の処理
統合データベースから変更をダウンロードするが、ローカルの変更をデータベースにアップロードしないかどうか。	download_only テーブル名サフィックスで、同期をダウンロードするのみのテーブルを識別できます。ローカルテーブルに対して行われた変更は、統合データベースにアップロードされません。
同期からテーブルを除外するかどうか。	テーブル名サフィックス nosync を使用して、同期しないテーブルを指定できます。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

設計時の確認事項	「はい」の場合の処理
データが変更されてないときも含め、常にテーブル全体を同期するかどうか。	テーブル名サフィックス allsync を使用すると、変更が検出されなくてもテーブル全体を同期できます。
テーブル全体を同期するか、特定の条件を満たすローだけを同期するか。重要度または緊急度により、同期の優先度が高いデータがあるかどうか。	パブリケーションには、同期が必要なテーブルをリストするアーティクルが含まれます。アーティクルには、ローが定義した基準を満たすかどうかに基づいてアップロードするローを指定する WHERE 句を含めることができます。 複数のパブリケーションを使用すると、特定の Ultra Light データを他のデータよりも前にアップロードする必要がある場合の優先度の問題に対処できます。
外部キーの循環があるため、同期するテーブルの順序を指定する必要があるかどうか。	TableOrder 同期パラメータを使用すると、外部キーの循環があるときの同期操作の順序を指定できます。ただし、外部キーの循環は一般に Ultra Light にはおすすめしません。
同期の動作を制御するかどうか。たとえば、アップロードと同時にダウンロードを行うかどうか。また、同期の方向を双方向から一方向に変更するかどうか。	次の場所で適切な同期パラメータを使用します。 ●アプリケーションの同期構造体 (または同期列挙) ●ulsync ユーティリティの -e オプション
Ultra Light クライアントで TLS を有効にするかどうか。	選択した暗号化アルゴリズムにより、デバイスで実行するプラットフォームに従ってデバイスを設定する方法が決まります。

参照

- 「Ultra Light ダウンロード専用テーブル」76 ページ
- 「Ultra Light 非同期テーブル」76 ページ
- 「Ultra Light の allsync (すべて同期) テーブル」77 ページ
- 「Ultra Light でのデータのパブリッシュ」78 ページ
- 「Ultra Light のテーブルの順序」79 ページ
- 「Ultra Light の同期パラメータ」89 ページ
- 「Ultra Light のネットワークプロトコルのオプション」111 ページ
- 「同期処理」『Mobile Link クイックスタート』
- 「アップロードとダウンロード」『Mobile Link クイックスタート』

Ultra Light 非同期テーブル

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

注意
テーブルの同期タイプは、アップロードが必要な未同期の変更がない場合にのみ変更できます。

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

```
ALTER TABLE purchase_comments_nosync
RENAME comments
```

修正するには、次のようにこのサフィックスを含む文に書き換えます。

```
ALTER TABLE purchase_comments_nosync
RENAME comments_nosync
```

またはパブリケーションを使用して同じ効果を上げることができます。

注意
SYNCHRONIZE OFF 句でテーブルを作成または変更する代わりに、テーブル名に **_nosync** フレーズを付加して非同期テーブルに変更することができます。

参照

- 「Download Only 同期パラメータ」94 ページ
- 「Upload Only 同期パラメータ」108 ページ
- 「Additional Parameters 同期パラメータ」90 ページ
- 「CREATE TABLE 文 [Ultra Light]」432 ページ
- 「ALTER TABLE 文 [Ultra Light]」422 ページ
- 「パブリケーション」『Mobile Link クライアント管理』
- 「Ultra Light パブリケーション」84 ページ
- 「CREATE PUBLICATION 文 [Ultra Light]」430 ページ
- 「同期スクリプトの作成」『Mobile Link サーバ管理』

Ultra Light ダウンロード専用テーブル

SYNCHRONIZE DOWNLOAD を使用してテーブルを作成することにより、テーブル全体をアップロード操作から除外します。これらのテーブルを統合データベースに同期しないテーブル用に使用できます。同期からは除外されていますが、テーブルの使用方法自体は Ultra Light データベース内の他のテーブルとまったく同じです。

またはパブリケーションを使用して同じ効果を上げることができます。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

注意
テーブルの同期タイプは、アップロードが必要な未同期の変更がない場合にのみ変更できます。

注意
SYNCHRONIZE DOWNLOAD 句でテーブルを作成または変更する代わりに、テーブル名に **_download_only** フレーズを付加してダウンロード専用テーブルに変更することができます。

参照

- 「Download Only 同期パラメータ」94 ページ
- 「Upload Only 同期パラメータ」108 ページ
- 「Additional Parameters 同期パラメータ」90 ページ
- 「CREATE TABLE 文 [Ultra Light]」432 ページ
- 「ALTER TABLE 文 [Ultra Light]」422 ページ
- 「パブリケーション」『Mobile Link クライアント管理』
- 「Ultra Light パブリケーション」84 ページ
- 「CREATE PUBLICATION 文 [Ultra Light]」430 ページ
- 「同期スクリプトの作成」『Mobile Link サーバ管理』

Ultra Light の allsync (すべて同期) テーブル

SYNCHRONIZE ALL を使用してテーブルを作成または変更すると、前回の同期セッション後に何も変更されていなくてもすべてのテーブルのデータが同期されるように、アップロード時の同期の動作を指定できます。

注意
テーブルの同期タイプは、アップロードが必要な未同期の変更がない場合にのみ変更できます。

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

注意
SYNCHRONIZE ALL 句でテーブルを作成または変更する代わりに、テーブル名に **_allsync** フレーズを付加して allsync テーブルに変更することができます。

参照

- 「Download Only 同期パラメータ」94 ページ
- 「Upload Only 同期パラメータ」108 ページ
- 「Additional Parameters 同期パラメータ」90 ページ
- 「CREATE TABLE 文 [Ultra Light]」432 ページ
- 「ALTER TABLE 文 [Ultra Light]」422 ページ
- 「パブリケーション」『Mobile Link クライアント管理』
- 「Ultra Light パブリケーション」84 ページ
- 「CREATE PUBLICATION 文 [Ultra Light]」430 ページ
- 「同期スクリプトの作成」『Mobile Link サーバ管理』

Ultra Light でのデータのパブリッシュ

パブリケーションは、同期するデータを指定するアーティクルを定義します。パブリケーションは、Sybase Central または SQL を使用して Ultra Light データベースに追加できます。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

パブリケーションアーティクルは 1 つのテーブルの全体を構成します。または、テーブル内にデータのサブセットを定義できます。特定のテーブルのローのサブセットを定義する場合は、オプションで述部 (WHERE 句) を含めることができます。パブリケーションは、SYNCHRONIZE OFF を使用してテーブルを作成する方法よりも柔軟です。Ultra Light データベースのデータサブセットを**別々**に同期させるには、複数のパブリケーションを使用します。パブリケーションをアップロード専用またはダウンロード専用の同期パラメータと組み合わせると、優先度の高い変更を効率よく同期することが可能です。

> **注意**
> Ultra Light でのユーザパブリケーションの最大数は 63 です。

Ultra Light パブリケーションでは、カラムのサブセットの定義と、SQL Anywhere で使用できる SUBSCRIBE BY 句がサポートされていません。Ultra Light テーブルのカラムが統合データベースのテーブルと正確に一致しない場合は、Mobile Link スクリプトを使用して、これらの違いを解消してください。

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

♦ タスク

1. Ultra Light プラグインを使用して、Ultra Light データベースに接続します。
2. **[パブリケーション]** フォルダを右クリックし、**[新規]** » **[パブリケーション]** をクリックします。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

3. 新しいパブリケーションの名前を入力します。**[次へ]** をクリックします。
4. **[テーブル]** タブで、**[一致するテーブル]** リストから 1 つのテーブルをクリックします。**[追加]** をクリックします。

   右側の **[選択したテーブル]** リストに、そのテーブルが表示されます。
5. テーブルを追加します。
6. 必要な場合は、**[WHERE]** タブをクリックし、パブリケーションに含めるローを指定します。カラムのサブセットは指定できません。
7. **[完了]** をクリックします。

結果

新しいパブリケーションが作成されます。

参照

- 「Ultra Light パブリケーション」84 ページ
- 「Download Only 同期パラメータ」94 ページ
- 「Upload Only 同期パラメータ」108 ページ
- 「Additional Parameters 同期パラメータ」90 ページ
- 「パブリケーション」『Mobile Link クライアント管理』
- 「CREATE PUBLICATION 文 [Ultra Light]」430 ページ
- 「同期スクリプトの作成」『Mobile Link サーバ管理』

Ultra Light のテーブルの順序

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

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

●外部キー循環。循環に含まれるすべてのテーブルをリストする必要があります。

●統合データベースとは異なる外部キー関係

外部キー循環に関連する同期の問題の防止

テーブル順序は、外部キー循環を使用する Ultra Light データベースに特に重要です。循環が形成されるようにテーブルをリンクさせると、循環が発生します。ただし、統合データベースと Ultra Light リモートデータベース間の循環が異なる場合は、外部キー循環は複雑になるので推奨しません。

外部キー循環がある場合は、プライマリテーブルへの操作が先に実行され、関連する外部テーブルへの操作が後に実行される順序にテーブルを並べてください。TableOrder パラメータによっ

て、外部テーブルへの挿入が外部キーの参照整合性制約を満たしていることが保証されます (削除などの他の操作でも同様です)。

テーブル順序とは別に、同期の問題を防ぐ方法として、トランザクションがコミットされるまで、参照整合性のチェックを延期できます。統合データベースが SQL Anywhere データベースの場合は、1 つの外部キーを **check on commit** (コミット時にチェック) に設定します。このように設定すると、外部キーの参照整合性が、操作の開始時ではなくコミットフェーズでチェックされます。次に例を示します。

```
CREATE TABLE c (
   id INTEGER NOT NULL PRIMARY KEY,
   c_pk INTEGER NOT NULL
);
CREATE TABLE p (
   pk INTEGER NOT NULL PRIMARY KEY,
   c_id INTEGER NOT NULL,
   FOREIGN KEY p_to_c (c_id) REFERENCES c(id)
);
ALTER TABLE c
  ADD FOREIGN KEY c_to_p (c_pk)
  REFERENCES p(pk)
  CHECK ON COMMIT;
```

他のデータベースベンダーの統合データベースを使用している場合は、参照総合性のチェックが同様の方法で行われているかどうかを確認します。同じであれば、この方法を実装します。方法が異なる場合は、すべての外部キー循環をなくすようにテーブル関係を再設計してください。

参照

- 「Additional Parameters 同期パラメータ」90 ページ
- 「参照整合性と同期」『Mobile Link クイックスタート』

Ultra Light アプリケーションの同期の設定

Ultra Light では、設定されている通信ストリーム (ネットワークプロトコル) で Mobile Link サーバとの特定の接続を開始することで同期が開始されます。ネットワークに直接接続している場合の同期のサポートに加えて、Windows Mobile デバイスでは ActiveSync 同期もサポートされています。

接続の定義

各 Ultra Light リモートデータベースは、ネットワークプロトコルによって Mobile Link サーバと同期します。ネットワークプロトコルは同期ストリームパラメータを使用して設定します。サポートされているネットワークプロトコルには、TCP/IP、HTTP、HTTPS、TLS があります。選択するプロトコルについて、Mobile Link サーバのホストやポートなどその他の必要な接続情報を定義するストリームパラメータを指定する必要があります。また、Mobile Link ユーザ情報と同期スクリプトのバージョンも指定してください。

同期動作の定義

同期動作は、さまざまな同期パラメータを設定することで制御できます。パラメータの設定方法は、使用している Ultra Light インタフェースによって異なります。

検討する必要がある重要な動作は、次のとおりです。

- **同期の方向**　デフォルトでは、同期は双方向です。一方向だけの同期を使用する場合は、upload_only パラメータまたは download_only パラメータを使用します。一方向の同期を実行することによって、同期に必要な時間を最小限に抑えることができます。また、ダウンロード専用同期では、Ultra Light データベースに対してあらかじめすべての変更をコミットする必要はありません。コミットしなかった変更は同期に含まれず、アップロードされません。したがって、不完全なトランザクションでも問題は起こりません。

  ダウンロード専用同期を使用する場合は、ダウンロードと重なっているローがローカルレベルで変更されていないことを確認してください。ローカルレベルでデータが変更されていると、SQLE_DOWNLOAD_CONFLICT エラーによって、Ultra Light アプリケーションの同期は失敗します。

- **同期中の変更**　アップロードフェーズでは、Ultra Light アプリケーションは、読み込み専用として Ultra Light データベースにアクセスできます。ダウンロードフェーズでは、読み込み／書き込みアクセスが許可されますが、アプリケーションがローを変更してからダウンロードによってこのローが変更されようとすると、ダウンロードが失敗し、ロールバックが行われます。DisableConcurrency 同期パラメータを設定することで、同期中のデータへの同時アクセスを無効にすることができます。

アプリケーションに同期機能を追加するには、通常、次の手順を行います。

1. 同期情報の構造体のフィールドとして、セッションに必要な同期パラメータとプロトコルオプションを指定します。

   たとえば、Ultra Light C/C++ API を使用する場合は、ul_sync_info 構造体に適切な値を設定して、アプリケーションに同期を追加します。

```
ul_sync_info info;
    // define a sync structure named "info"
     ULEnableTcpipSynchronization( &sqlca );
    // use a TCP/IP stream
    conn->InitSynchInfo( &info );
    // initialize the structure
    info.stream = ULSocketStream();
    // specify the Socket Stream
    info.stream_parms= UL_TEXT( "host=myMLserver;port=2439"  );
    // set the MobiLink host information
    info.version = UL_TEXT( "custdb 11.0" );
    // set the MobiLink version information
    info.user_name = UL_TEXT( "50" );
    // set the MobiLink user name
    info.download_only =ul_true;
    // make the synchronization download-only
```

2. 同期を初期化します。

   直接同期する場合は、API 固有の同期メソッドを呼び出します。これらのメソッドは、同期操作の成功または失敗を示すブール値を返します。同期が失敗した場合は、別の構造体で詳細なエラーステータスのフィールドを検証して、追加のエラー情報を取得できます。

   ActiveSync 同期の場合は、ActiveSync プロバイダからの同期メッセージを取得し、DoSync メソッドを使用して ULSynchronize メソッドを呼び出す必要があります。

3. 同期の進行状況をユーザにレポートする場合は、observer コールバック関数を使用します。

ヒント
DLL が非常に大きいか、ネットワーク接続の信頼性が低いため、DLL を正常に実行できない場合は、再開可能なダウンロードを実装できます。「ダウンロードの失敗」『Mobile Link サーバ管理』と「ダウンロードの失敗の再開」『Mobile Link サーバ管理』を参照してください。

参照

- 「Windows Mobile での Ultra Light と ActiveSync」87 ページ
- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link サーバ管理』
- 「アップロードとダウンロード」『Mobile Link クイックスタート』
- 「Ultra Light の同期パラメータ」89 ページ
- 「Ultra Light のネットワークプロトコルのオプション」111 ページ
- Ultra Light.NET： 「Mobile Link データ同期」『Ultra Light .NET プログラミング』
- Ultra Light C/C++： 「Mobile Link データ同期」『Ultra Light C/C++ プログラミング』
- Ultra Light C/C++： 「ActiveSync の同期設定」『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL アプリケーションの同期の設定」『Ultra Light C/C++ プログラミング』
- Ultra Light J API： 「Mobile Link データ同期」『Ultra Light Java プログラミング』

Mobile Link ファイル転送

Ultra Light では、Mobile Link サーバを使用してファイルを転送する機能がサポートされています。

その他のすべての API については、次の場合に Mobile Link のファイル転送メカニズムを使用します。

- 特にセキュリティ対策に企業のファイアウォールを使用していて、複数のファイルを複数のデバイスに配備する必要がある場合。Mobile Link は、企業のファイアウォールを介して同期を処理するように設定済みなので、MLFileTransfer メカニズムにより、アップグレードや他のタイプのファイル転送のためのデバイスのプロビジョニングが非常に便利になります。

- 特定の Mobile Link ユーザ ID を対象とするファイルがある場合。ユーザ ID ごとに、Mobile Link サーバで 1 つまたは複数のユーザ固有のディレクトリを作成する必要があります。ファイルのバージョンが 1 つだけの場合は、デフォルトのディレクトリを使用できます。

ファイル転送のしくみ

ファイルをデバイスにダウンロードする際に使用できるファイル転送方法は 2 種類あります。デスクトップ転送用の mlfiletransfer ユーティリティを実行する方法か、または使用する API に適切な関数を呼び出して Ultra Light アプリケーションをコーディングする方法です。どちらの方法でも次の処理が必要です。

1. 転送先を指定します。

デスクトップから mlfiletransfer ユーティリティを使用するか、API に適切な関数を使用するかに関係なく、転送先のデバイスまたはデスクトップコンピュータにあるファイルのローカルパスとファイル名を設定する必要があります。アプリケーションで、またはエンドユーザがこの情報を指定しなかった場合、ファイルは転送元のファイル名で現在のフォルダに保存されます。

ターゲットの転送先フォルダは、次のようにデバイスのオペレーティングシステムによって異なります。

- Windows Mobile では、転送先が NULL の場合、ファイルはルートフォルダ (¥) に格納されます。

  ファイル名は、Windows Mobile のファイルの命名規則に従っている必要があります。「Windows Mobile」38 ページを参照してください。

- デスクトップコンピュータでは、転送先が NULL の場合、ファイルは現在のフォルダに格納されます。

  ファイル名は、デスクトップシステムのファイルの命名規則に従っている必要があります。「デスクトップ」38 ページを参照してください。

- iPhone では、アプリケーションのドキュメントフォルダにファイルを格納してください。ドキュメントフォルダのロケーションを取得するには、NSDocumentDirectory パラメータを使用して NSSearchPathForDirectories/uDomains を呼び出します。

- BlackBerry スマートフォンでは、FileTransfer オブジェクトを使用してロケーションを設定します。

2. ユーザが識別され、正しいファイルがダウンロードされるように、Mobile Link ユーザクレデンシャルを設定します。

   このユーザ名とパスワードは他のデータベースユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するために使用されます。

3. 使用するストリームタイプを設定し、目的とするストリームのパラメータを定義します。このパラメータは、Ultra Light で Mobile Link 同期用にサポートされているパラメータと同じです。

   ほとんどの同期ストリームでは、Mobile Link サーバのアドレスを識別したり、その他の動作を制御したりするパラメータが必要です。ストリームタイプを、プラットフォームに適さない無効な値に設定すると、ストリームタイプは TCP/IP に設定されます。ただし、HTTP のみをサポートしている Ultra Light Java Edition データベースを除きます。

4. 転送メカニズムに必要な動作を指定します。

   たとえば、転送先にファイルがすでに存在し、変更されていない場合でも、ダウンロードを強制したり、部分的なダウンロードを再開したりするようにプロパティを設定できます。また、進行状況をモニタし、レポートするかどうかも設定できます。

5. Mobile Link サーバが実行中であり、-ftr オプションを指定して起動されたことを確認します。

6. 転送を開始し、必要な場合はダウンロードの進行状況をモニタします。

   ダウンロードの進行状況を表示することで、ユーザはダウンロードをキャンセルし、後で再開できます。

参照

- 「Ultra Light の同期パラメータ」89 ページ
- 「Ultra Light のネットワークプロトコルのオプション」111 ページ
- 「-ftr mlsrv16 オプション」『Mobile Link サーバ管理』
- 「Mobile Link ファイル転送ユーティリティ (mlfiletransfer)」『Mobile Link クライアント管理』
- MLFileDownload メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- MLFileUpload メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULFileTransfer クラス [Ultra Light.NET]『Ultra Light .NET プログラミング』
- FileTransfer インタフェース [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light パブリケーション

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

パブリケーションはアーティクルのセットで構成されます。各アーティクルはテーブル全体、またはテーブル内のローです。WHERE 句を使用してこのローのセットを定義できます。

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

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

使用上の注意

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

- カラムは常に、CREATE TABLE 文で定義された順序で送信されます。

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

- Ultra Light ではオブジェクトの所有権がサポートされていないので、すべてのユーザがパブリケーションを削除できます。

参照

- 「Ultra Light のテーブルの順序」 79 ページ
- 「パブリケーション」『Mobile Link クライアント管理』
- 「Ultra Light クライアントの同期設計」 74 ページ
- 「同期スクリプトの作成」『Mobile Link サーバ管理』

Ultra Light でのテーブル全体のパブリッシュ

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

前提条件

この作業を実行するための前提条件はありません。

内容と備考

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

♦ タスク

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

結果

パブリケーションが作成されます。

次の手順

なし。

参照

- 「CREATE PUBLICATION 文 [Ultra Light]」 430 ページ
- 「Ultra Light クライアント」 69 ページ

Ultra Light テーブルのローのサブセットのパブリッシュ

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

前提条件

この作業を実行するための前提条件はありません。

内容と備考

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

♦ タスク

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

結果

アップロードされるローは、変更されたローのうち WHERE 句の探索条件に一致するローだけに制限されます。

次の手順

なし。

参照

- 「CREATE PUBLICATION 文 [Ultra Light]」430 ページ
- 「Ultra Light クライアント」69 ページ

Ultra Light のパブリケーションの削除

テーブルのパブリケーションを削除すると、Ultra Light データベース内のすべてのテーブルとそのテーブルのローを同期できます。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

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

♦ タスク

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

結果

パブリケーションが削除されます。

参照

- 「DROP PUBLICATION 文 [Ultra Light]」441 ページ
- 「Ultra Light クライアント」69 ページ

Windows Mobile での Ultra Light と ActiveSync

Windows Mobile デバイスのデータは、Ethernet や Wi-Fi 接続を介して同期することもできますが、この項では、ActiveSync 同期用にデスクトップとデバイスを設定して、Ultra Light データベースが他の ActiveSync 操作と同時に同期されるようにする方法について説明します。別の方法を使用して同期を直接行う場合は、適切な同期機能を使用してアプリケーションのプログラムを設定する必要があります。

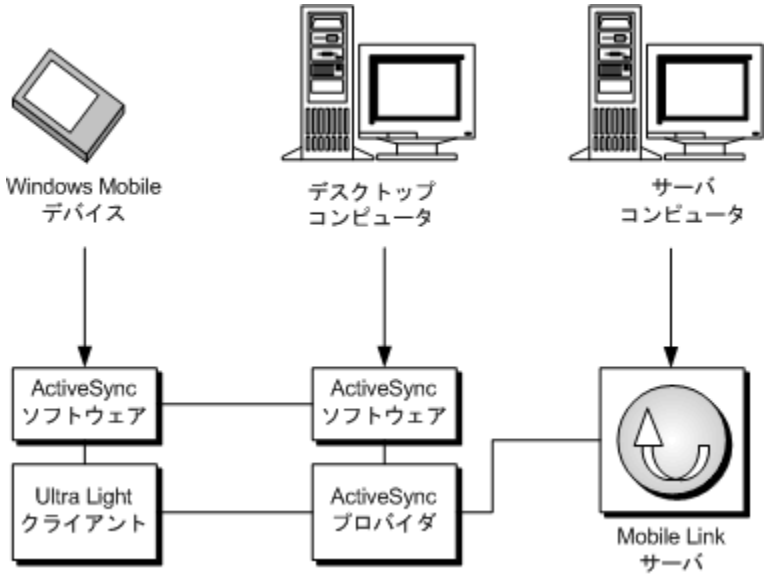
ActiveSync 起動の同期を使用するには、次のことが必要です。

- ActiveSync を使用して ActiveSync 起動の同期を使用するのに必要なすべてのアプリケーションを登録する。
- ActiveSync プロバイダをデスクトップにインストールし、デバイスに配備する。

  プロバイダがサポートされているプラットフォームを確認するには、http://www.sybase.com/detail?id=1002288 を参照してください。

ActiveSync アーキテクチャ

次の図は、ActiveSync アーキテクチャで必要なコンピューティングレイヤを示しています。



ActiveSync プロバイダは、デスクトップだけでなくデバイスにもインストールしてください。1 台のコンピュータでは 1 つの ActiveSync プロバイダのみ使用できます。ただし、複数の Ultra Light アプリケーションを Windows Mobile デバイスにインストールしている場合は、アプリケーションを同じプロバイダに登録して同時に同期が行われるようにすることができます。

参照

- Ultra Light C/C++ : 「ActiveSync の同期設定」『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL アプリケーションの同期の設定」『Ultra Light C/C++ プログラミング』

ActiveSync 同期の概要

1. ActiveSync によって同期セッションが開始されます。
2. ActiveSync プロバイダは、同期の通知メッセージをデバイス上の最初に登録されているアプリケーションに送信します。アプリケーションが実行中でない場合は、起動されます。
3. 登録されたアプリケーションごとに WndProc が起動されます。
4. アプリケーションは、メッセージが ActiveSync からの同期通知メッセージであると判断すると、ULIsSynchronizeMessage を呼び出してデータベース同期プロシージャを起動します。
5. 同期処理が完了すると、アプリケーションは ULSignalSyncIsComplete を呼び出して、同期が終了したことをプロバイダに通知します。
6. プロバイダに登録されている各アプリケーションについて、手順 2 ～ 5 が繰り返されます。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

Ultra Light の同期パラメータ

同期パラメータは、Ultra Light データベースと Mobile Link サーバ間の同期を制御します。パラメータの設定方法は、使用している Ultra Light インタフェースによって異なります。この項では、パラメータの影響について説明し、これらの設定方法が説明された他の項へのリンクを示します。

注意
この項で説明するパラメータが適用されるのは、Ultra Light リモートデータベースのみです。SQL Anywhere リモートデータベースを同期させる場合は、「Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)」『Mobile Link クライアント管理』を参照してください。

必須パラメータ

次のパラメータは必須です。

- Stream Type
- User Name
- Version

これらのパラメータが設定されていない場合、同期関数は例外 (SQLCode.SQLE_SYNC_INFO_INVALID またはこれと同じもの) をスローします。

パラメータの競合

次のパラメータは、いずれか 1 つのみ指定できます。

- Download Only
- Ping
- Upload Only

これらのパラメータの 1 つ以上を true に設定した場合、同期関数は例外 (SQLCode.SQLE_SYNC_INFO_INVALID またはこれと同じもの) をスローします。

参照

- ULSynchronize メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULSyncParms クラス [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULConnection.Synchronize メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- SyncParms クラス [Ultra Light J]『Ultra Light Java プログラミング』
- 「Stream Type 同期パラメータ」105 ページ
- 「User Name 同期パラメータ」110 ページ
- 「Version 同期パラメータ」111 ページ
- 「Download Only 同期パラメータ」94 ページ
- 「Ping 同期パラメータ」100 ページ
- 「Upload Only 同期パラメータ」108 ページ

Additional Parameters 同期パラメータ

この同期パラメータを使用すると、他の定義済みパラメータでは簡単に指定できない追加パラメータをアプリケーションで指定できます。このパラメータフィールドには、あまり使用されないパラメータが指定されます。

追加パラメータは、複数の keyword=value 設定をセミコロンで区切った文字列として指定されます。

構文

構文は、使用する API によって異なります。Ultra Light J では使用できません。

指定可能な値

次のプロパティを追加パラメータ設定の一部として指定できます。

プロパティ名	説明
AllowDownloadDupRows	同期時に、ダウンロードされたローにプライマリキーの重複が検出された場合にエラーにしないようにします。 このプロパティを **0** に設定すると、エラーが発生し、ダウンロードをロールバックします。**1** に設定すると、警告が発生し、ダウンロードを続行します。 このプロパティは Ultra Light C/C++ でのみ使用できます。
CheckpointStore	同期中にデータベースのチェックポイントを追加して、同期プロセス中のデータベースの拡張を制限します。 このプロパティを **1** に設定するとこの機能が有効になります。多くの更新を伴う大量のダウンロードには最適ですが、同期処理に時間がかかるようになります。デフォルトでは **0** に設定されています。
DisableConcurrency	同期のアップロードフェーズ中は、他のスレッドからのデータベースへのアクセスを禁止します。 このプロパティを **0** に設定すると、データベースへの同時アクセスが許可されます。許可しないためには **1** に設定します。デフォルトでは、このプロパティは **0** に設定されています。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

プロパティ名	説明
TableOrder	Ultra Light のデフォルトのテーブルの順序が適切でない場合に、優先同期に必要なテーブルの順序を設定します。 このプロパティを設定して、テーブル名をアップロード順に並べたリストを作成します。Ultra Light ではカンマで区切られたリストを使用し、ulsync ではセミコロンで区切られたリストを使用します。デフォルトでは、外部キーの関係に基づいた順序で並べられます。統合データベースの外部キーが Ultra Light リモートに一致し、外部キー循環がない場合は、通常、デフォルトの順序を使用できます。 一重または二重引用符でテーブル名を囲みます。たとえば、"Customer,Sales" と 'Customer,Sales' の両方が Ultra Light でサポートされています。 同期に含まれていないテーブルを指定すると、そのテーブルは無視されます。リストしないテーブルは、リモートデータベースで定義した外部キーに基づいて適切に保存されます。 ダウンロードのテーブルの順序は、アップロードで定義した順序と同じです。 Ultra Light テーブルが次のいずれかに該当する場合は、テーブル順序を明示的に設定するだけですみます。 ●外部キー循環に含まれる場合。循環に含まれるすべてのテーブルをリストする必要があります。 ●統合データベース内の外部キーの関係が異なる場合。

参照

- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.AdditionalParms プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のように追加パラメータを設定できます。

```
ul_sync_info info;
// ...
info.additional_parms = UL_TEXT(
   "AllowDownloadDupRows=1;
   CheckpointStore=1;
   DisableConcurrency=1;
   TableOrder=Customer,Sales"
);
```

Authentication Parameters 同期パラメータ

Mobile Link イベントの認証パラメータにパラメータを渡します。

構文

構文は、使用する API によって異なります。

備考

パラメータには、ユーザ名やパスワードなどがあります。

このパラメータを使用する場合、パラメータの数も指定してください。

指定可能な値

文字列の配列。文字列の値として NULL は使用できませんが、空の文字列は指定できます。

参照

- 「Number of Authentication Parameters パラメータ」97 ページ
- 「認証パラメータ」『Mobile Link サーバ管理』
- 「authenticate_parameters 接続イベント」『Mobile Link サーバ管理』
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.AuthenticationParms プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setAuthenticationParms メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_char * Params[ 3 ] = { UL_TEXT( "parm1" ),
                          UL_TEXT( "parm2" ),
                          UL_TEXT( "parm3" ) };
// ...
info.num_auth_parms = 3;
info.auth_parms = Params;
```

Authentication Status 同期パラメータ

このフィールドは同期によって設定され、Mobile Link のユーザ認証のステータスをレポートます。Mobile Link サーバが、この情報をクライアントに提供します。

構文

構文は、使用する API によって異なります。

指定可能な値

使用できる値は、インタフェース固有の列挙に格納されています。たとえば、C/C++ アプリケーションの列挙は次のようになります。

定数	値	説明
UL_AUTH_STATUS_UNKNOWN	0	認証ステータスが不明です。接続がまだ同期していない可能性があります。
UL_AUTH_STATUS_VALID	1	ユーザ ID とパスワードは、同期時には有効でした。
UL_AUTH_STATUS_VALID_BUT_EXPIRES_SOON	2	ユーザ ID とパスワードは、同期時には有効でしたが、まもなく有効期限が切れます。
UL_AUTH_STATUS_EXPIRED	3	認証に失敗しました。ユーザ ID またはパスワードの有効期限が切れています。
UL_AUTH_STATUS_INVALID	4	認証に失敗しました。ユーザ ID またはパスワードが不正です。
UL_AUTH_STATUS_IN_USE	5	認証に失敗しました。ユーザ ID がすでに使用されています。

備考

統合データベースでカスタム **authenticate_user** 同期スクリプトが異なる値を返す場合は、authenticate_user 接続イベントで指定されているルールに従って値が解釈されます。

カスタム認証スキームを実装している場合、authenticate_user または authenticate_user_hashed 同期スクリプトは、このパラメータの有効値の 1 つを返します。

このパラメータは Mobile Link サーバによって設定されるため、読み込み専用です。

参照

- 「authenticate_user 接続イベント」『Mobile Link サーバ管理』
- 「Mobile Link ユーザ」『Mobile Link クライアント管理』
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncResult.AuthStatus プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncResult.getAuthStatus メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_sync_info info;
// ...
returncode = info.auth_status;
```

Authentication Value 同期パラメータ

このフィールドは同期によって設定され、カスタム Mobile Link のユーザ認証スクリプトの結果をレポートします。Mobile Link サーバが、この情報をクライアントに提供します。

構文

構文は、使用する API によって異なります。Ultra Light J では使用できません。

備考

デフォルトの Mobile Link ユーザ認証メカニズムによって設定される値については、authenticate_user 接続イベントと Authentication Status 同期パラメータの項で説明します。

このパラメータは Mobile Link サーバによって設定されるため、読み込み専用です。

参照

- 「authenticate_user 接続イベント」『Mobile Link サーバ管理』
- 「authenticate_user_hashed 接続イベント」『Mobile Link サーバ管理』
- 「Authentication Status 同期パラメータ」92 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncResult.AuthValue プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncResult.getAuthValue メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_sync_info info;
// ...
returncode = info.auth_value;
```

Download Only 同期パラメータ

この同期中は、Ultra Light データベースから変更がアップロードされないようにします。

構文

構文は、使用する API によって異なります。

デフォルト

False

指定可能な値

ブール式

競合するパラメータ

Ping と Upload Only

備考

ダウンロードのみの同期の場合、データ変更はアップロードされません。ただし、スキーマに関する情報と進行状況のカウンタに格納されている値はアップロードされます。ダウンロードされたデータが、アップロードされていないリモートでの変更と矛盾する場合、同期は失敗し、ロールバックされます。

Ultra Light Java Edition データベーに、ダウンロードのみの同期を使用する場合は、定期的に完全な同期を実行して、リモートデータベースの同期時にスキャンされるトランザクションログの量を減らす必要があります。そうしないと、同期が完了するのに次第に時間がかかるようになります。

参照

- 「Upload Only 同期パラメータ」108 ページ
- 「同期のステータスの追跡」3 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.DownloadOnly プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setDownloadOnly メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

次の例は、ulsync ユーティリティを使用して DownloadOnly 同期パラメータを設定する方法を示しています。

```
ulsync -c DBF=myuldb.udb
   "MobiLinkUid=remoteA;ScriptVersion=2;DownloadOnly=ON;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.download_only = ul_true;
```

Ignored Rows 同期パラメータ

このフィールドは同期によって設定され、同期中にスクリプトがないために Mobile Link サーバによってローが無視されたことを示します。

構文

構文は、使用する API によって異なります。

指定可能な値

ブール式

備考

このパラメータは読み込み専用です。

参照

- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncResult.IgnoredRows プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncResult.getIgnoredRows メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_sync_info info;
// ...
res = info.ignored_rows;
```

Keep Partial Download 同期パラメータ

同期時の通信エラーが原因でダウンロードが失敗したときに、変更をロールバックしないで部分的なダウンロードを保持するかどうかを制御します。

構文

構文は、使用する API によって異なります。Ultra Light J では使用できません。

デフォルト

False (ダウンロードが失敗したときにすべての変更をロールバック)

指定可能な値

ブール式

参照

- 「ダウンロードの失敗の再開」『Mobile Link サーバ管理』
- 「Resume Partial Download 同期パラメータ」 102 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.KeepPartialDownload プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.keep_partial_download = ul_true;
```

New Password 同期パラメータ

ユーザ名に対する新しい Mobile Link パスワードを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

備考

このパラメータはオプションです。

参照

- 「Mobile Link ユーザ」『Mobile Link クライアント管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.NewPassword プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setPassword メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;NewMobiLinkPwd=mynewpassword;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.new_password = UL_TEXT( "mlnewpass" );
```

Number of Authentication Parameters パラメータ

Mobile Link イベントの認証パラメータに渡す認証パラメータの数を指定します。

構文

構文は、使用する API によって異なります。Ultra Light J では必要ありません。

デフォルト

カスタム認証スクリプトに渡されるパラメータはありません。

備考

Authentication Parameters とともにパラメータを使用して、カスタム認証スクリプトに情報を提供します。

参照

- 「Authentication Parameters 同期パラメータ」92 ページ
- 「authenticate_parameters 接続イベント」『Mobile Link サーバ管理』
- 「認証パラメータ」『Mobile Link サーバ管理』
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.AuthenticationParms プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.num_auth_parms = 3;
```

Observer 同期パラメータ

同期をモニタするコールバック関数またはイベントハンドラへのポインタを指定します。使用するために実装する必要があるコールバック関数のシグネチャは **ul_sync_observer_fn** 型です。

```
typedef void(UL_CALLBACK_FN *ul_sync_observer_fn)( ul_sync_status * status );
```

構文

構文は、使用する API によって異なります。

参照

- 「User Data 同期パラメータ」109 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncProgressListener インタフェース [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setSyncObserver メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.observer=callfunction;
```

Partial Download Retained 同期パラメータ

このフィールドは同期によって設定され、同期時の通信エラーが原因でダウンロードが失敗したときに、変更をロールバックしないで、ダウンロードされたこの変更が適用されたかどうかを示します。

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

構文

構文は、使用する API によって異なります。Ultra Light Java Edition データベースではサポートされていません。

指定可能な値

ブール式

備考

ダウンロードエラーが発生して部分的なダウンロードが保持された場合、同期時にパラメータが設定されます。

部分的ダウンロードが保持されるのは、Keep Partial Download が true に設定されている場合のみです。

参照

- 「Keep Partial Download 同期パラメータ」96 ページ
- 「ダウンロードの失敗の再開」『Mobile Link サーバ管理』
- 「Resume Partial Download 同期パラメータ」102 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncResult.PartialDownloadRetained プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

このパラメータにアクセスするには、次のように入力します。

```
ul_sync_info info;
// ...
returncode=info.partial_download_retained;
```

Password 同期パラメータ

ユーザ名に対する Mobile Link パスワードを指定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

備考

このパラメータはオプションです。

この Mobile Link のユーザ名とパスワードは他のデータベースユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するためだけに使用されます。

Mobile Link クライアントにパスワードが設定されている場合は、New Password パラメータを使用してパスワードを変更します。

参照

- 「User Name 同期パラメータ」110 ページ
- 「New Password 同期パラメータ」96 ページ
- 「Mobile Link ユーザ」『Mobile Link クライアント管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.Password プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setPassword メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;MobiLinkPwd=mypassword;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.password = UL_TEXT( "mypassword" );
```

Ping 同期パラメータ

Ultra Light クライアントと Mobile Link サーバ間の通信を確認します。このパラメータが true に設定されている場合は、同期は行われません。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

false

指定可能な値

ブール式

備考

Mobile Link サーバは、ping を受信すると、統合データベースに接続し、ユーザを認証し、ユーザ認証ステータスと値をクライアントに送信します。

ping に成功した場合、Mobile Link サーバは情報メッセージを発行します。ping に失敗した場合は、エラーメッセージを発行します。

Mobile Link サーバがコマンドラインオプション -zu+ を指定して実行されていると、Mobile Link ユーザ ID が ml_user システムテーブルに見つからない場合は Mobile Link サーバがユーザを ml_user に追加します。

Mobile Link サーバに次のスクリプトが存在する場合、ping 要求に対してこれらのスクリプトを実行できます。

- begin_connection
- authenticate_user
- authenticate_user_hashed
- authenticate_parameters
- end_connection

参照

- 「-pi dbmlsync オプション」『Mobile Link クライアント管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.PingOnly プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setPingOnly メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;Ping=True;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.ping = ul_true;
```

Publications 同期パラメータ

同期させるパブリケーションを指定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync で使用することもできます。

デフォルト

すべてのパブリケーションを同期します。

備考

C/C++ で同期する場合は、パブリケーション同期パラメータを 「パブリケーションリスト」、つまり、パブリケーション名のカンマ区切りリストで設定します。

参照

- 「Ultra Light でのデータのパブリッシュ」78 ページ
- 「Ultra Light パブリケーション」84 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULConnection.SYNC_ALL_DB フィールド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULConnection.SYNC_ALL_PUBS フィールド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setPublications メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;Publications=UL_PUB_MYPUB1,UL_PUB_MYPUB2;Stream=ht
tp"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.publications = UL_TEXT( "Pubs1,Pubs3" );
```

Resume Partial Download 同期パラメータ

失敗したダウンロードを再開します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。Ultra Light Java Edition データベースではサポートされていません。

デフォルト

false

指定可能な値

ブール式

備考

同期によって変更はアップロードされず、失敗したダウンロードで変更のみがダウンロードされます。

参照

- 「ダウンロードの失敗の再開」『Mobile Link サーバ管理』
- 「Keep Partial Download 同期パラメータ」96 ページ
- 「Partial Download Retained 同期パラメータ」98 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.ResumePartialDownload プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.resume_partial_download = ul_true;
```

Send Download Acknowledgement 同期パラメータ

クライアントがダウンロード確認を提供することを Mobile Link サーバに指示します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

false

指定可能な値

ブール式

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.SendDownloadAck プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setAcknowledgeDownload メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb
"MobiLinkUid=remoteA;ScriptVersion=2;SendDownloadACK=true;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.send_download_ack = ul_true;
```

Stream Error 同期パラメータ

通信エラーレポート情報を保持する構造体です。

構文

構文は、使用する API によって異なります。

適用対象

このパラメータは、C/C++ インタフェースにのみ適用されます。

指定可能な値

このパラメータにはデフォルト値がないので、サポートされているフィールドの一つを使用して明示的に設定する必要があります。ul_stream_error のフィールドは、次のとおりです。

- **stream_error_code**　エラーコードのサフィックスについては、*%SQLANY16%¥SDK¥Include¥sserror.h* を参照してください。
- **system_error_code**　システム固有のエラーコード。エラーコードの詳細については、プラットフォームのマニュアルを参照してください。Windows プラットフォームの場合は、Microsoft Developer Network (MSDN) マニュアルを参照してください。

  Windows における一般的なシステムエラーを次に示します。

  - **10048 (WSAADDRINUSE)**　アドレスがすでに使用されています。
  - **10053 (WSAECONNABORTED)**　ソフトウェアが接続のアボートを引き起こしました。
  - **10054 (WSAECONNRESET)**　通信の他方の側がソケットを閉じました。
  - **10060 (WSAETIMEDOUT)**　接続がタイムアウトしました。
  - **10061 (WSAECONNREFUSED)**　接続が拒否されました。通常、Mobile Link サーバが稼働していないか、指定されたポートで受信していません。http://msdn2.microsoft.com/en-us/library/ms740668.aspx を参照してください。
- **error_string**　stream_error_code の追加情報 (ある場合) の文字列。文字列は空の場合もあれば、空でない場合もあります。 error_string の値が空でない場合、stream_error_code の値に追加情報が提供されます。たとえば、書き込みエラー (エラーコード 9) の場合、エラー文字列は書き込もうとしたバイト数を示します。

備考

Ultra Light C++ コンポーネント以外の Ultra Light アプリケーションは、Sync Result パラメータの一部として通信エラー情報を受け取ります。

stream_error フィールドは ul_stream_error 型の構造体です。

```
typedef struct {
    ss_error_code stream_error_code;
    asa_uint16    alignment;
    asa_int32     system_error_code;
    char          error_string[UL_STREAM_ERROR_STRING_SIZE];
} ul_stream_error, * p_ul_stream_error;
```

この構造体は、*%SQLANY16%¥SDK¥Include¥sserror.h* で定義されています。

SQLE_COMMUNICATIONS_ERROR をチェックします。

```
ULConnection * conn;
ul_sync_info info;
...
conn->InitSynchInfo( &info );
if( !conn->Synchronize( &info ) ) {
    ULError const * error = conn->GetLastError();
    char buf[256];
    if( error->GetSQLCode() == SQLE_MOBILINK_COMMUNICATIONS_ERROR ) {
        error->GetString( buf, sizeof(buf) );
        printf( "%s¥n", buf );
        // more handling for communication error
    }
}
```

参照

- 「Mobile Link 通信エラーメッセージ」『エラーメッセージ』
- 「Sync Result 同期パラメータ」 107 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』

Stream Type 同期パラメータ

同期に使用する Mobile Link ネットワークプロトコルを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

備考

このパラメータは必須です。デフォルト値はありません。

ほとんどのネットワークプロトコルでは、Mobile Link サーバアドレスや他の動作を識別するプロトコルオプションが必要です。これらのオプションは、Stream Parameters パラメータで指定します。

ネットワークプロトコルがオプションを必要とする場合は、Stream Parameters を使用してそのオプションを渡すか、あるいは Stream Parameters パラメータを NULL に設定します。

ストリームのタイプには次の種類がありますが、すべてのターゲットプラットフォームですべての関数を使用できるわけではありません。

ネットワークプロトコル	説明
HTTP	HTTP を介して同期します。
HTTPS	HTTPS を介して同期します。 HTTPS プロトコルは、基本のセキュリティレイヤとして TLS を使用します。TCP/IP を介して機能します。
TCP/IP	TCP/IP を介して同期します。このプロトコルは、Ultra Light Java Edition データベースではサポートされていません。
TLS	トランスポートレイヤセキュリティ (TLS) 付きの TCP/IP を介して同期します。TLS は、デジタル証明書とパブリックキー暗号方式を使用して、クライアント／サーバ通信をセキュリティ保護します。

サポートされるプラットフォームのリストについては、http://www.sybase.com/detail?id=1061806 を参照してください。

参照

- 「Stream Parameters 同期パラメータ」106 ページ
- 「証明書作成ユーティリティ (createcert)」『SQL Anywhere サーバ データベース管理』
- 「証明書ビューアユーティリティ (viewcert)」『SQL Anywhere サーバ データベース管理』
- 「トランスポートレイヤセキュリティ」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- 「Ultra Light のネットワークプロトコルのオプション」111 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.Stream プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定します。

```
Connection conn;
ul_sync_info info;
...
conn.InitSynchInfo( &info );
info.stream = "http";
```

Stream Parameters 同期パラメータ

ネットワークプロトコルを設定するオプションを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

NULL

指定可能な値

文字列

備考

このパラメータはオプションです。このパラメータは、セミコロンで区切られたネットワークプロトコルオプションのリストを受け入れます。各オプションは *keyword=value* の形式で、許可されるキーワードセットはネットワークプロトコルによって異なります。

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- 「Ultra Light のネットワークプロトコルのオプション」111 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.StreamParms プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- StreamHTTPParms インタフェース [Ultra Light J]『Ultra Light Java プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.stream_parms= UL_TEXT( "host=myserver;port=2439" );
```

Sync Result 同期パラメータ

同期のステータスをレポートします。

構文

構文は、使用する API によって異なります。

備考

このパラメータは Ultra Light によって設定され、読み込み専用です。

C/C++ インタフェースでは、この情報を ul_sync_info 構造体の中で複数の別個のパラメータで受け取ります。それ以外の場合は、この情報は、さまざまな情報がそれぞれ別個のフィールドに格納された複合パラメータとして定義されます。

- **Authentication Status**　認証の成功または失敗をレポートします。
- **Ignored Rows**　無視されるローの数をレポートします。
- **Stream Error 情報**　Stream Error 情報には、Stream Error Code、Stream Error Context、Stream Error ID、Stream Error System が含まれています。

- **Upload OK**　アップロードフェーズの成功または失敗をレポートします。

参照

- 「Authentication Status 同期パラメータ」92 ページ
- 「Ignored Rows 同期パラメータ」95 ページ
- 「Stream Error 同期パラメータ」104 ページ
- 「Upload OK 同期パラメータ」108 ページ
- ULSyncParms クラス [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ul_sync_result 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULGetSyncResult メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- SyncResult クラス [Ultra Light J]『Ultra Light Java プログラミング』

Upload OK 同期パラメータ

このフィールドは同期によって設定され、Mobile Link サーバにアップロードされたデータのステータスをレポートします。

構文

構文は、使用する API によって異なります。

備考

このパラメータは Ultra Light によって設定されるため、読み込み専用です。

同期後、このパラメータには、アップロードが成功した場合は **true**、それ以外の場合は **false** が入ります。同期エラーがあったかどうかについてこのパラメータを確認することによって、エラーが発生する前にデータが正常にアップロードされたかどうかを確認できます。

参照

- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncResult.UploadOK プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncResult.isUploadOK メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_sync_info info;
// ...
returncode = info.upload_ok;
```

Upload Only 同期パラメータ

現在の同期中にダウンロードは発生しないことを示します。これにより、特に低速の通信リンクでは、通信時間を節約できます。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

false

指定可能な値

ブール式

競合するパラメータ

Download Only、Ping、Resume Partial Download

備考

true に設定すると、クライアントは Mobile Link サーバからのアップロード確認を待ってから、同期セッションを正常終了します。

参照

- 「Ultra Light クライアントの同期設計」74 ページ
- 「Download Only 同期パラメータ」94 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.UploadOnly プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.isUploadOnly メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;UploadOnly=True;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.upload_only = ul_true;
```

User Data 同期パラメータ

アプリケーション固有の情報を同期 observer で使用できるようにします。

適用対象

C/C++ アプリケーションのみ。Ultra Light.NET などのコンポーネントは、ユーザデータの処理に別個のパラメータを必要としないので、User Data パラメータはありません。

構文

構文は、使用する API によって異なります。

備考

同期 observer コールバック関数またはイベントハンドラの実装時に、User Data パラメータを使用して情報を指定することによって、アプリケーション固有の情報を使用可能にできます。

参照

- 「Observer 同期パラメータ」98 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』

User Name 同期パラメータ

必須。Mobile Link サーバが認証用に使用する文字列です。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

備考

このパラメータは必須です。空の文字列と NULL 文字列は一般に拒否されます。

パラメータにはデフォルト値がないので、明示的に設定してください。

リモート ID が使用されている場合は、ユーザ名はユニークである必要はありません。

この Mobile Link のユーザ名とパスワードは他のデータベースユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するためだけに使用されます。

同期システムに含まれるユーザの場合は、Mobile Link サーバを使用してユーザ名を登録する必要があります。ユーザ名は、統合データベースの ml_user Mobile Link システムテーブルの name カラムに格納されます。

参照

- 「リモート ID」『Mobile Link クライアント管理』
- 「Password 同期パラメータ」99 ページ
- 「Mobile Link ユーザ」『Mobile Link クライアント管理』
- 「Ultra Light ユーザ認証」『Mobile Link クライアント管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.UserName プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setUserName メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.user_name= UL_TEXT( "remoteA" );
```

Version 同期パラメータ

統合データベースのバージョンを定義します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

備考

このパラメータは必須です。空の文字列と NULL 文字列は一般に拒否されます。

統合データベースの同期スクリプトは、それぞれバージョン文字列で区別されます。たとえば、異なるバージョン文字列によって特定される 2 つの download_cursor スクリプトが存在する場合があります。

参照

- 「スクリプトバージョン」『Mobile Link サーバ管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページ
- ul_sync_info 構造体 [Ultra Light C および Embedded SQL データタイプ]『Ultra Light C/C++ プログラミング』
- ULSyncParms.Version プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- SyncParms.setVersion メソッド [Ultra Light J]『Ultra Light Java プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;ScriptVersion=2;Stream=http"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_sync_info info;
// ...
info.version = UL_TEXT( "default" );
```

Ultra Light のネットワークプロトコルのオプション

アプリケーションでネットワークプロトコルを設定する必要があります。各 Ultra Light データベースは、ネットワークプロトコルによって Mobile Link サーバと同期します。使用可能なネッ

トワークプロトコルには、TCP/IP、HTTP、HTTPS、TLS があります。Windows Mobile での ActiveSync 通知もサポートされています。

設定したネットワークプロトコルでは、対応するプロトコルオプションのセットから選択することによって、Ultra Light アプリケーションが Mobile Link 同期サーバを特定して正しく通信できるようにします。Mobile Link クライアントのネットワークプロトコルオプションは、アドレス情報 (ホストとポート) やプロトコル固有の情報などを提供します。

参照

- 「Mobile Link クライアントネットワークプロトコルオプション」『Mobile Link クライアント管理』
- 「トランスポートレイヤセキュリティを使用する Ultra Light クライアントの設定」『SQL Anywhere サーバ データベース管理』
- 「Mobile Link クライアントネットワークプロトコルオプション」『Mobile Link クライアント管理』
- 「Stream Parameters 同期パラメータ」106 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」224 ページの -x オプション

同期ストリームのオプション

Stream Parameters パラメータを設定することによって、アプリケーション内で Mobile Link サーバを特定するのに必要な情報を提供できます。

参照

- 「Stream Parameters 同期パラメータ」106 ページ
- ULSyncParms.StreamParms プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

Ultra Light の配備

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

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

Ultra Light アプリケーションで使用する必要があるモジュールは、目標とするプラットフォーム、使用するインタフェース、および使用したい機能によって異なります。

参照

- 「Windows Mobile 用の Ultra Light データ管理コンポーネント」 18 ページ
- 「Ultra Light C++ アプリケーションの構築と配備」『Ultra Light C/C++ プログラミング』
- 「Ultra Light.NET アプリケーション開発」『Ultra Light .NET プログラミング』
- 「Ultra Light J アプリケーションの構築と配備」『Ultra Light Java プログラミング』

Ultra Light アプリケーションのビルドと配備の仕様

この項には、Ultra Light アプリケーションのビルドと配備の要件を示す表が含まれています。

> **注意**
> Ultra Light エンジンのバージョンによっては、**_dev** サフィックスを含むディレクトリ (*x86_dev* ディレクトリなど) に置かれていることがあります。これらのバージョンには、デバッグを目的としたプラットフォーム上の問題の診断に使用できる開発時のログ機能が含まれています。このログ出力は、ユーザではなく、Sybase のエンジニアが使用することを目的としたものです。運用システムでは、**_dev** ディレクトリにないバージョンのエンジンを使用してください。

Ultra Light アプリケーションおよび暗号化のビルドと配備の要件

次の表に、Ultra Light データベースの暗号化の要件を含め、サポート対象のすべてのプラットフォームとデバイスにおける Ultra Light アプリケーションのビルドと配備の最小要件を示します。

プラットフォームまたはデバイス	最小要件	AES 暗号化の要件	FIPS 140-2 AES 暗号化の要件
Windows Mobile とデスクトップ (静的リンクを使用する Ultra Light C/C++)	リンク先： ● *ulrt.lib*[1] ● *ulbase.lib*[1]	データベースの作成時または接続時に、DBKEY 作成パラメータを使用して暗号化キーを設定します。 EnableAesDBEncryption メソッドを呼び出します。	データベースの作成時と接続時に、暗号化キーに DBKEY 作成パラメータを設定します。 データベースの作成時に作成パラメータ **fips=yes** を設定します。 EnableAesFipsDBEncryption メソッドを呼び出します。 展開： ● *ulfips16.dll*[2] ● *sbgse2.dll*[2]
Windows Mobile とデスクトップ (動的リンクを使用する Ultra Light C/C++)	リンク先： ● *ulimp.lib*[1, 10] ● *ulbase.lib*[1] 展開： ● *ulrt16.dll*[1]	データベースの作成時または接続時に、DBKEY 作成パラメータを使用して暗号化キーを設定します。 EnableAesDBEncryption メソッドを呼び出します。	データベースの作成時と接続時に、暗号化キーに DBKEY 作成パラメータを設定します。 データベースの作成時に作成パラメータ **fips=yes** を設定します。 EnableAesFipsDBEncryption メソッドを呼び出します。 展開： ● *ulfips16.dll*[2] ● *sbgse2.dll*[2]

プラットフォームまたはデバイス	最小要件	AES 暗号化の要件	FIPS 140-2 AES 暗号化の要件
Windows Mobile とデスクトップ (Ultra Light エンジンを使用する Ultra Light C/C++)	リンク先： ● *ulrtc.lib*[1] ● *ulbase.lib*[1] 展開： ● *uleng16.exe*[2]	データベースの作成時または接続時に、DBKEY 作成パラメータを使用して暗号化キーを設定します。	データベースの作成時と接続時に、暗号化キーに DBKEY 作成パラメータを設定します。 データベースの作成時に作成パラメータ **fips=yes** を設定します。 展開： ● *ulfips16.dll*[2] ● *sbgse2.dll*[2]
Windows Mobile とデスクトップ (Ultra Light.NET)	参照の追加先： ● **iAnywhere.Data.UltraLite** ● **iAnywhere.Data.UltraLite.resources** 展開： ● *iAnywhere.Data.UltraLite.dll*[7] ● *iAnywhere.Data.UltraLite.resources.dll*[8] ● *ulnet16.dll*[6]	データベースの作成時または接続時に、DBKEY 作成パラメータを使用して暗号化キーを設定します。	データベースの作成時と接続時に、暗号化キーに DBKEY 作成パラメータを設定します。 データベースの作成時に作成パラメータ **fips=yes** を設定します。 展開： ● *ulfips16.dll*[2] ● *sbgse2.dll*[2]

プラットフォームまたはデバイス	最小要件	AES 暗号化の要件	FIPS 140-2 AES 暗号化の要件
Windows Mobile とデスクトップ (Ultra Light エンジンを使用する Ultra Light.NET)	参照の追加先： ● **iAnywhere.Data.UltraLite** ● **iAnywhere.Data.UltraLite.resources** 展開： ● *iAnywhere.Data.UltraLite.dll*[7] ● *iAnywhere.Data.UltraLite.resources.dll*[8] ● *ulnetclient16.dll*[6] ● *uleng16.exe*[2]	データベースの作成時または接続時に、DBKEY 作成パラメータを使用して暗号化キーを設定します。	データベースの作成時と接続時に、暗号化キーに DBKEY 作成パラメータを設定します。 データベースの作成時に作成パラメータ **fips=yes** を設定します。 展開： ● *ulfips16.dll*[2] ● *sbgse2.dll*[2]
Mac OS X および iOS (Ultra Light C++)	Xcode プロジェクトに追加： ● *libulrt.a*[9] ● *libulbase.a*[9] (Mac OS X のみ)	データベースの作成時または接続時に、DBKEY 作成パラメータを使用して暗号化キーを設定します。 EnableAesDBEncryption メソッドを呼び出します。	該当なし
Linux (UltraLite C/C++)	リンク先： ● *libulrt.a*[3] ● *libulbase.a*[3]	データベースの作成時または接続時に、DBKEY 作成パラメータを使用して暗号化キーを設定します。 EnableAesDBEncryption メソッドを呼び出します。	該当なし

プラットフォームまたはデバイス	最小要件	AES 暗号化の要件	FIPS 140-2 AES 暗号化の要件
Android (Ultra Light J)	Android プロジェクトに追加： ● *UltraLiteJNI16.jar*[5] ● *libultralitej16.so*[4]	データベースの作成時または接続時に、DBKEY 作成パラメータまたは setEncryptionKey メソッドを使用して暗号化キーを設定します。 EnableAesDBEncryption メソッドを呼び出します。	該当なし
BlackBerry (Ultra Light J)	展開： ● *UltraLiteJ16.cod*[11] ● *UltraLiteJ16.jad*[11][12]	データベースの作成時または接続時に、setEncryptionKey メソッドを使用して暗号化キーを設定します。 EnableAesDBEncryption メソッドを呼び出します。	該当なし

[1] Windows Mobile の場合、このファイルは *%SQLANY16%¥UltraLite¥CE¥Arm.50¥Lib* にあります。Windows の場合は、*%SQLANY16%¥UltraLite¥Windows¥x64¥Lib¥VS9* または *%SQLANY16%¥UltraLite¥Windows¥x86¥Lib¥VS9* にあります。

[2] Windows Mobile の場合、このファイルは *%SQLANY16%¥UltraLite¥CE¥Arm.50* にあります。Windows の場合は、*%SQLANY16%¥UltraLite¥Windows¥x64* または *%SQLANY16%¥UltraLite¥Windows¥x86* にあります。

[3] このファイルは */opt/sqlanywhere16/ultralite/linux/x64/lib* にあります。

[4] このファイルは、*%SQLANY16%¥UltraLite¥UltraLiteJ¥Android¥ARM* にあります。

[5] このファイルは、*%SQLANY16%¥UltraLite¥UltraLiteJ¥Android* にあります。

[6] Windows Mobile の場合、このファイルは *%SQLANY16%¥UltraLite¥CE¥Arm.50* にあります。Windows の場合は、*%SQLANY16%¥UltraLite¥UltraLite.NET¥x64* または *%SQLANY16%¥UltraLite¥UltraLite.NET¥win32* にあります。

[7] このファイルは、*%SQLANY16%¥UltraLite¥UltraLite.NET¥Assembly¥V2* にあります。

[8] このファイルは、*%SQLANY16%¥UltraLite¥UltraLite.NET¥Assembly¥V2¥ja* にあります。

[9] Mac OS X の場合、このファイルは、*/Applications/SQLAnywhere16/System/ultralite/macosx/x86_64* にあります。iOS の場合、Ultra Light ランタイムはインストール後にビルドしてください。*install-dir/ultralite/iphone/readme.txt* の指示に従ってください。

[10] このライブラリにリンクする場合は、コンパイル時に UL_USE_DLL プリプロセッサマクロを定義してください。たとえば、次のように指定します。

```
-DUL_USE_DLL
```

[11] このファイルは、*%SQLANY16%¥UltraLite¥UltraLiteJ¥BlackBerry4.2* にあります。

[12] 無線配信 (OTA : over-the-air) での配備にのみ必要です。また、アプリケーションとともに Ultra Light J を配備する独自の *.jad* ファイルを作成することもできます。

注意
いずれかのプラットフォームまたはデバイス上で Ultra Light データベースの難読化を使用するには、データベースの作成時に作成パラメータ **obfuscate=1** を指定する必要があります。

同期および圧縮のための追加のビルドと配備の要件

次の表に、同期を使用する Ultra Light アプリケーションのビルドと配備のためのストリーム、プロトコルオプション、およびコードの各要件を示します。

注意
HTTPS ストリームオプションを Ultra Light J API で有効にするには、SyncParms.HTTPS_STREAM 定数を Connection.createSyncParms メソッドに渡します。

同期タイプ	ストリームオプション指定	プロトコルオプション要件	Ultra Light C と C++ のメソッドコール要件
TCP/IP	"tcpip"	なし	● EnableTcpipSynchronization
HTTP	"http"	なし	● EnableHttpSynchronization
RSA TLS	"tls"	なし	● EnableTlsSynchronization ● EnableRsaSyncEncryption
RSA HTTPS	"https"	なし	● EnableHttpsSynchronization ● EnableRsaSyncEncryption

同期タイプ	ストリームオプション指定	プロトコルオプション要件	Ultra Light C と C++ のメソッドコール要件
RSA FIPS 140-2 TLS	"tls"	**fips=yes**	● EnableTlsSynchronization ● EnableRsaFipsSyncEncryption
RSA FIPS 140-2 HTTPS	"https"	**fips=yes**	● EnableHttpsSynchronization ● EnableRsaFipsSyncEncryption

次の表に、圧縮またはエンドツーエンド暗号化を使用する Ultra Light アプリケーションのビルドと配備のための追加のプロトコルオプションとコードの要件を示します。

圧縮とストリーム暗号化のオプション	プロトコルオプション要件	Ultra Light C と C++ および Ultra Light J のメソッドコール要件
ZLIB 圧縮	● **compression=zlib**	●「C/C++：」EnableZlibSyncCompression ●「Java：」setZlibCompression
RSA E2EE	● **e2ee_public_key=***key-file*	●「C/C++：」 EnableRsaE2ee ●「Java：」 setE2eePublicKey
RSA FIPS 140-2 E2EE	● **e2ee_public_key=***key-file* ● **fips=yes**	●「C/C++：」EnableRsaFipsE2ee ●「Java：」 該当なし

次の表に、圧縮および暗号化した同期化のための追加のビルドと配備の要件を示します。

注意
TCP/IP および HTTP 同期には、追加のビルドと配備の要件はありません。

プラットフォームまたはデバイス	ZLIB 圧縮の要件	RSA TLS、RSA HTTPS、および RSA E2EE の要件	RSA FIPS 140-2 TLS、RSA FIPS 140-2 HTTPS、および RSA FIPS 140-2 E2EE の要件
Windows Mobile とデスクトップ (静的リンクを使用する Ultra Light C/C++)	なし	リンク先： ● *ulrsa.lib*[1]	展開： ● *mlcrsafips16.dll*[2] ● *sbgse2.dll*[2]
Windows Mobile とデスクトップ (動的リンクを使用する Ultra Light C/C++)	展開： ● *mlczlib16.dll*[2]	展開： ● *mlcrsa16.dll*[2]	展開： ● *mlcrsafips16.dll*[2] ● *sbgse2.dll*[2]
Windows Mobile とデスクトップ (Ultra Light エンジンを使用する Ultra Light C/C++)	展開： ● *mlczlib16.dll*[2]	展開： ● *mlcrsa16.dll*[2]	展開： ● *mlcrsafips16.dll*[2] ● *sbgse2.dll*[2]
Windows Mobile とデスクトップ (Ultra Light.NET)	展開： ● *mlczlib16.dll*[2]	展開： ● *mlcrsa16.dll*[2]	展開： ● *mlcrsafips16.dll*[2] ● *sbgse2.dll*[2]
Windows Mobile とデスクトップ (Ultra Light エンジンを使用する Ultra Light.NET)	展開： ● *mlczlib16.dll*[2]	展開： ● *mlcrsa16.dll*[2]	展開： ● *mlcrsafips16.dll*[2] ● *sbgse2.dll*[2]
Mac OS X および iOS (Ultra Light C++)	なし	なし	該当なし
Linux (UltraLite C/C++)	なし	リンク先： ● *libulrsa.a*[3]	該当なし
Android (Ultra Light J)	なし	展開： ● *libmlcrsa16.so*[4]	該当なし

プラットフォームまたはデバイス	ZLIB 圧縮の要件	RSA TLS、RSA HTTPS、および RSA E2EE の要件	RSA FIPS 140-2 TLS、RSA FIPS 140-2 HTTPS、および RSA FIPS 140-2 E2EE の要件
BlackBerry (Ultra Light J)	なし	ファイル転送インタフェースを使用して DER でコード化されたファイルを転送するか、またはこれを SD カードに保存します。	該当なし

[1] Windows Mobile の場合、このファイルは *%SQLANY16%¥UltraLite¥CE¥Arm.50¥Lib* にあります。Windows の場合は、*%SQLANY16%¥UltraLite¥Windows¥x64¥Lib¥VS9* または *%SQLANY16%¥UltraLite¥Windows¥x86¥Lib¥VS9* にあります。

[2] Windows Mobile の場合、このファイルは *%SQLANY16%¥CE¥Arm.50* にあります。Windows の場合は、*%SQLANY16%¥Windows¥x64* または *%SQLANY16%¥Windows¥x86* にあります。

[3] このファイルは */opt/sqlanywhere16/ultralite/linux/x64/lib* にあります。

[4] このファイルは、*%SQLANY16%¥UltraLite¥UltraLiteJ¥Android¥ARM* にあります。

参照

- 「Ultra Light C++ アプリケーションの構築と配備」『Ultra Light C/C++ プログラミング』
- 「Ultra Light.NET アプリケーションの構築と配備」『Ultra Light .NET プログラミング』
- 「Ultra Light J アプリケーションの構築と配備」『Ultra Light Java プログラミング』
- 「Mobile Link クライアントネットワークプロトコルオプション」『Mobile Link クライアント管理』

Ultra Light および Ultra Light Java Edition データベースの配備に関する制限事項

次のいずれかの方法を使用して、デバイス上で初期データベースファイルを取得します。

- アプリケーションの Ultra Light API を使用して、初期データベースファイルを作成します。
- SQL スクリプトからスキーマファイルを作成し、ALTER DATABASE SCHEMA FROM FILE 文を使用します。[1]
- Ultra Light の FileTransfer メソッドを使用して、初期データベースファイルをダウンロードします (デバイス上に既存のファイルがない場合)。
- 初期データベースをアプリケーションとバンドルします。[1]
- Windows または Windows Mobile デバイスに配備する場合は、中央管理を使用して初期 UDB ファイルを送信するか、コマンドを送信して初期データベースファイルを作成し、SQL を実行してスキーマを提供します。[1]

[1] Ultra Light Java Edition データベースではサポートされません。

参照

- 「Ultra Light データベーススキーマのアップグレードの配備」 122 ページ
- 「ALTER DATABASE SCHEMA FROM FILE 文 [Ultra Light]」 418 ページ
- 「リモートデータベースの集中管理」『Mobile Link サーバ管理』

Ultra Light データベーススキーマのアップグレードの配備

スキーマのアップグレードを実行します。

前提条件

使用する SQL ファイルには、新しいスキーマ全体が含まれている必要があります。

内容と備考

Ultra Light データベーススキーマのアップグレードは、次のいずれかの方法で配備できます。

- **個々の DDL 文**　たとえば、Ultra Light C では、次の文を実行して新しいパブリケーションを作成できます。

```
dbconnection->ExecuteStatement("CREATE PUBLICATION p (table t)");
```

- **ALTER DATABASE SCHEMA FROM FILE 文**　DDL 文の要件がわからない場合、または個々の DDL 文を指定したくない場合は、この文を使用してスキーマのアップグレードを実行できます。

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

ALTER DATABASE SCHEMA FROM FILE 文を使用して Ultra Light データベーススキーマをアップグレードすると、Ultra Light で次の手順が実行されます。

1. 新しいデータベーススキーマと既存のデータベーススキーマの違いが比較されます。
2. 既存のデータベースのスキーマは変更されます。
3. 新しいスキーマに適合しないローは削除されます。次に例を示します。

   ○テーブルに一意性制約を追加し、同じ値を持つローが複数ある場合は、1 つのロー以外すべて削除される。

○カラムドメインを変更して変換エラーが発生した場合は、そのローが削除される。たとえば、VARCHAR カラムを INT カラムに変更した場合、ローの値が ABCD であれば、そのローは削除されます。

○新しいスキーマに新しい外部キーがあり、外部ローに一致するプライマリローがない場合は、これらのローが削除される。

4. ローが削除されると、SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE (130) 警告が発行されます。

♦ タスク

1. 完全に新しいスキーマを作成するには、DDL 文の SQL スクリプトを作成します。

   コンピュータ上にマスタスキーマを保持し、アプリケーションの変更に応じてスキーマを更新できます。

   スクリプトに必要な DDL 文を抽出するには、ulinit または ulunload のいずれかのユーティリティを使用します。これらのユーティリティを次のオプションとともに使用して、DDL 文が構文的に正しいことを確認してください。

   ●Ultra Light データベースの場合、ulunload ユーティリティを使用し、-n および -s [*schema-file*] オプションを指定する。次に例を示します。

   ```
   ulunload -c dbf=mydatabase.udb -n -s MySchema.sql
   ```

   ●SQL Anywhere データベースの場合、ulinit ユーティリティを使用し、-l [*file*] オプションを指定する。次に例を示します。

   ```
   ulinit -a "dsn=mysqlanywheredatabase" -l MySchema.sql
   ```

   ulunload および ulinit ユーティリティを使用しない場合は、スクリプトで次の事項を確認してください。

   ●スクリプトに、CREATE 文によって目的のスキーマ全体が宣言されている。

   ●表、カラム、パブリケーションの名前が変更されていない。RENAME 操作はサポートされていません。テーブルの名前を変更すると、DROP TABLE 操作または CREATE TABLE 操作として処理されます。

   ●影響が発生しない可能性がある文も含め、DDL 文以外の文が含まれていない。

   ●SQL 文の語はスペースで区切る。

   ●各行に 1 つの SQL 文だけが表示されている。

   ●コメントの先頭には、二重ハイフン (-) を付ける。コメントを指定できるのは、行の先頭のみです。

   ●各文が、**GO** という単語を含む 1 行によって区切られている。

2. 新しい SQL スクリプトファイルの配備

3. データベースが同期されていることを確認します。

4. デバイス上で新しい文を実行します。次に例を示します。

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

結果

スキーマが更新されます。

エラー通知

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

参照

- 「Ultra Light データベースのスキーマ」49 ページ
- 「ALTER DATABASE SCHEMA FROM FILE 文 [Ultra Light]」418 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースのアンロードユーティリティ (ulunload)」232 ページ

Ultra Light エンジンの起動

Ultra Light エンジンを使用して Windows または Windows Mobile デバイス上のデータを管理すると、Ultra Light アプリケーションにエンジンのディレクトリの場所を明示的に指定する必要がある場合を除き、アプリケーションによってエンジンが自動的に起動します。

Ultra Light アプリケーションによって Ultra Light エンジンの起動が試行されると、次のディレクトリが検索されます。

クライアントプラットフォーム	ディレクトリの場所
Windows デスクトップ	1. Ultra Light エンジンを起動するアプリケーションのディレクトリ 2. 現在の作業ディレクトリ 3. システムパス 4. SQL Anywhere インストールディレクトリ (*bin32* または *bin64* の下)、クライアントが 32 ビットか 64 ビットかによって異なる

クライアントプラットフォーム	ディレクトリの場所
Windows Mobile/CE	1. *¥Windows¥* 2. ¥ (ルートディレクトリ) 3. *¥UltraLiteDB¥*
Linux	1. Ultra Light エンジンを自動起動するアプリケーションのディレクトリ 2. *%SQLANY16%/bin32*

Ultra Light エンジンが異なる場所に格納されている場合は、START 接続パラメータを指定してエンジンを起動します。

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

```
"START=¥Program Files¥MyApp¥uleng16.exe"
```

参照

- 「Ultra Light アプリケーションのビルドと配備の仕様」113 ページ
- 「Ultra Light START 接続パラメータ」184 ページ

Ultra Light の ActiveSync プロバイダの配備

Ultra Light をエンドユーザに配備する場合は、エンドユーザのコンピュータに ActiveSync プロバイダを手動でインストールして登録してください。これにより、ActiveSync は特定のアプリケーションに対してプロバイダの特定のインスタンスを呼び出すタイミングを認識します。

前提条件

エンドユーザのコンピュータで次のことを確認します。

- ActiveSync Manager がインストールされていること。
- ActiveSync プロバイダファイルが開発コンピュータからエンドユーザのハードドライブにコピーされていること。

内容と備考

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

- ***mlasinst.exe*** ActiveSync プロバイダをインストールして、ActiveSync Manager で登録します。このユーティリティは、同期用に ActiveSync プロバイダで使用するアプリケーションも登録します。
- ***mlasdesk.dll*** ActiveSync Manager によってデスクトップにロードされる DLL。DLL ファイルのロケーションは、*mlasinst.exe* によって ActiveSync Manager に登録されます。
- ***mlasdev.dll*** ActiveSync Manager によってデバイスにロードされる DLL。DLL ファイルは、*mlasinst.exe* によってデバイスの正しいロケーションに配備されます。
- ***dblgen16.dll*** 言語リソースライブラリ。

サポートされているプロバイダプラットフォームのリストについては、http://www.sybase.com/detail?id=1002288 を参照してください。

♦ タスク

1. mlasinst を実行して、ActiveSync のプロバイダをインストールします。このユーティリティを使用して、Ultra Light アプリケーションをユーザの Windows Mobile デバイスに登録して配備することもできます。–ただし、可能かどうかは、使用するコマンドライン構文によります。Ultra Light アプリケーションで複数のファイルを使用する場合は、必要なファイルを手動でコピーしてください。

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

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

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

   ```
   mlasinst -v "%SQLANY16%¥UltraLite¥ce¥arm.50"
   "%SQLANY16%¥UltraLite¥ce¥arm.50¥custdb.exe" custdb.exe CustDB CUSTDBDEMO
   ```

   注意
   ActiveSync を使用して、後で Ultra Light アプリケーションを登録することもできます。
   「ActiveSync Manager を使用したアプリケーションの登録」127 ページを参照してください。

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

結果

ファイルがデバイスに配備され、Windows Mobile ActiveSync を使用して Ultra Light を操作できるようになります。

参照

- 「ActiveSync Manager を使用したアプリケーションの登録」127 ページ
- 「Microsoft ActiveSync プロバイダインストールユーティリティ (mlasinst)」『Mobile Link クライアント管理』

ActiveSync Manager を使用したアプリケーションの登録

ActiveSync 同期を使用するアプリケーションを ActiveSync で登録し、デバイスにコピーします。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

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

♦ タスク

1. ActiveSync を起動します。
2. [ActiveSync] ウィンドウで **[オプション]** をクリックします。
3. 情報タイプのリストから **[Mobile Link クライアント]** をクリックし、**[設定]** をクリックします。
4. **[Mobile Link 同期]** ウィンドウで **[新規]** をクリックします。
5. アプリケーションについて次の情報を入力します。
   - **[名前]**　ActiveSync ユーザインタフェースに表示されるアプリケーションを識別する名前。
   - **[クラス名]**　アプリケーションの登録済みクラス名。
   - **[パス]**　アプリケーションのデバイス上のロケーション。
   - **[引数]**　ActiveSync でアプリケーションの起動時に使用するコマンドライン引数。
6. **[OK]** をクリックしてアプリケーションを登録します。

結果

ActiveSync でアプリケーションが登録されます。

次の手順

アプリケーションをデバイスにコピーします。

参照

- 「アプリケーションに対するクラス名の割り当て」『Ultra Light C/C++ プログラミング』
- 「Microsoft ActiveSync プロバイダインストールユーティリティ (mlasinst)」『Mobile Link クライアント管理』

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

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

このチュートリアルでは、次の作業の手順を学習します。

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

サポートされているプログラミングインタフェースとプラットフォームごとに、複数のバージョンのアプリケーションコードがあります。ただし、このチュートリアルでは、Windows デスクトップ用アプリケーションのコンパイル済みバージョンだけを使用します。各バージョンは、各プラットフォームの変換によって変わります。

参照

- 「CustDB サンプルアプリケーションの概要」 14 ページ
- 「CustDB シナリオ」『Mobile Link クイックスタート』
- 「CustDB サンプル内のユーザ」『Mobile Link クイックスタート』
- 「CustDB データベース内のテーブル」『Mobile Link クイックスタート』

レッスン 1：CustDB アプリケーションの構築と実行

このレッスンでは、CustDB アプリケーションを構築して実行します。

前提条件

この作業を実行するための前提条件はありません。

♦ タスク

1. 非 Windows 環境の場合、CustDB アプリケーションを構築します。
   a. 適切な環境で CustDB プロジェクトファイルを開きます。
   b. ソースコードをコンパイルします。
2. CustDB アプリケーションを実行します。

   Windows 32 ビット環境の場合、*%SQLANY16%¥UltraLite¥Windows¥x86¥custdb.exe* を実行します。

結果

CustDB アプリケーションが実行されます。

次の手順

「レッスン 2：Mobile Link サーバの起動と初期同期の実行」130 ページに進みます。

参照

- 「CustDB ファイルのロケーション 」16 ページ

レッスン 2：Mobile Link サーバの起動と初期同期の実行

このレッスンでは、Mobile Link サーバを起動し、CustDB アプリケーションを使用して、CustDB データベースを Ultra Light データベースと同期します。

前提条件

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：CustDB アプリケーションの構築と実行」129 ページを参照してください。

♦ タスク

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 16]** » **[Mobile Link]** » **[同期サーバのサンプル]** をクリックします。または、次のコマンドを実行します。

   ```
   mlsrv16 -c "DSN=SQL Anywhere 16 CustDB" -vcrs
   ```

   Mac OS X または Linux では、*mobilink.sh* を使用します。

   ウィンドウには、Mobile Link サーバのステータスに関する情報が表示されます。

2. **[スタート]** » **[プログラム]** » **[SQL Anywhere 16]** » **[Ultra Light]** » **[Windows サンプルアプリケーション]** をクリックします。

3. **[ファイル]** メニューで、**[同期]** をクリックします。

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

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

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

結果

CustDB アプリケーションが統合データベースと同期します。

次の手順

「レッスン 3：Ultra Light データベース内のデータの更新」131 ページに進みます。

レッスン 3：Ultra Light データベース内のデータの更新

このレッスンでは、CustDB アプリケーションを使用して、リモート データベース内のデータの追加、更新、削除を行います。

前提条件

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：CustDB アプリケーションの構築と実行」129 ページを参照してください。

♦ タスク

1. 注文をブラウズします。

   注文のブラウズは、CustDB アプリケーションの各バージョンで同様の方法で行います。注文をブラウズするときは、ローカルの Ultra Light データベースのデータをスクロールします。顧客はアルファベット順でソートされているので、簡単にリストをスクロールして顧客の名前を見つけることができます。

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

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

2. 注文を追加します。

   注文の追加は、CustDB アプリケーションの各バージョンで同様の方法で行います。注文を追加すると、ローカルの Ultra Light データベースでデータが修正されます。このデータは、同期が行われるまで統合データベースとは共有されません。

   a. **[Order]** » **[New]** をクリックします。

   b. **[Customer]** リストで **Basements R Us** をクリックします。

   c. **[Product]** リストで **Screwmaster Drill** をクリックします。この製品の価格は **[Price]** フィールドに自動的に設定されます。

   d. **[Quantity]** フィールドに **20** と入力します。

   e. **[Discount]** フィールドに、**5** (パーセント) と入力し、**[OK]** をクリックします。

3. 注文の承認、拒否、削除を行います。

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

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

i. **[Previous]** をクリックしてこの顧客を探します。
ii. **[Order]** » **[Approve]** をクリックして、注文を承認します。
iii. **[Note]** リストで **Good** をクリックします。
iv. **[OK]** をクリックします。

注文情報には **[Approved]** のステータスが表示されます。

b. **Art's Renovations** からの注文を拒否します。
i. リストの次の項目である **Art's Renovations** からの注文を表示します。
ii. **[Order]** » **[Deny]** をクリックして、注文を拒否します。
iii. **[Note]** リストで、**[Discount Is Too High]** をクリックします。
iv. **[OK]** をクリックします。

注文情報には [Denied] のステータスが表示されます。

c. **Awnings R Us** からの注文を削除します。
i. リストの次の項目である **Awnings R Us** からの注文を表示します。
ii. **[Order]** » **[Delete]** を選択してこの注文を削除します。
iii. **[OK]** をクリックして、削除を確認します。

注文に削除済みのマークが付けられます。ただし、変更内容を統合データベースと同期するまでは、現在のデータが Ultra Light リモートデータベースに残ります。

結果

Ultra Light データベースのデータへの変更は保存されますが、CustDB データベースとは同期されません。

次の手順

「レッスン 4：Ultra Light データベースと統合データベースの同期」132 ページに進みます。

参照

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

レッスン 4：Ultra Light データベースと統合データベースの同期

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

前提条件

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：CustDB アプリケーションの構築と実行」129 ページを参照してください。

内容と備考

CustDB アプリケーションで同期を行うと、承認された注文情報がデータベースから削除されます。

♦ タスク

1. Ultra Light データベースを同期します。

   [ファイル] メニューで、**[データベースの同期]** をクリックします。

2. 同期されたことを確認します。

   リモートデータベースでは、**Awnings R Us** の注文が削除されたことを確認することで、必要なトランザクションがすべて完了したことを確認できます。このエントリがないことを確認するには、注文をブラウズします。

   統合データベースでは、データを調べて、必要な処理がすべて実行されたことを確認できます。

   ●Sybase Central を使用し、同期が実行されたことを確認します。

   a. **[スタート]** » **[プログラム]** » **[SQL Anywhere 16]** » **[管理ツール]** » **[Sybase Central]** をクリックします。
   b. **[接続]** » **[SQL Anywhere 16 に接続]** をクリックします。
   c. **[アクション]** ドロップダウンメニューで、**[ODBC データソースを使用した接続]** をクリックします。
   d. **[ODBC データソース名]** をクリックします。
   e. **[参照]** をクリックし、**SQL Anywhere 16 CustDB** をクリックします。
   f. **[OK]** をクリックします。
   g. **[接続]** をクリックします。
   h. **[テーブル]** をダブルクリックします。
   i. **[ULOrder]** をダブルクリックします。
   j. **[データ]** タブをクリックし、注文 5100 が承認され、注文 5101 が拒否され、注文 5102 が削除されていることを確認します。

   ●Interactive SQL を使用して、同期が実行されたことを確認します。

   a. Interactive SQL から統合データベースに接続します。

      i. **[スタート]** » **[プログラム]** » **[SQL Anywhere 16]** » **[管理ツール]** » **[Interactive SQL]** をクリックします。
      ii. **[アクション]** ドロップダウンリストで、**[ODBC データソースを使用した接続]** をクリックします。
      iii. **[ODBC データソース名]** をクリックし、**SQL Anywhere 16 CustDB** をクリックします。

b. 承認や拒否が同期されたことを確認するため、次の文を実行します。

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

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

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

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

結果

承認された注文はデータベースから削除され、この削除を確認しました。

次の手順

「レッスン 5：Mobile Link 同期スクリプトのブラウズ」134 ページに進みます。

レッスン 5：Mobile Link 同期スクリプトのブラウズ

このレッスンでは、CustDB の同期ロジックの機能をよく理解するために、同期スクリプトをブラウズします。

前提条件

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：CustDB アプリケーションの構築と実行」129 ページを参照してください。

♦ タスク

1. **[スタート]** » **[プログラム]** » **[SQL Anywhere 16]** » **[管理ツール]** » **[Sybase Central]** をクリックします。

2. **[Sybase Central]** の左ウィンドウ枠で、**[Mobile Link 16]** を右クリックして、**[プロジェクトを開く]** をクリックします。

3. *%SQLANYSAMP16%¥MobiLink¥CustDB* に移動し、*project.mlp* をクリックします。

4. **[統合データベース]** をダブルクリックし、CustDB 統合データベースをダブルクリックします。

   統合データベースをプロジェクトに追加するときに入力した接続情報に基づいて、統合データベースに接続されます。

5. **[接続スクリプト]** をダブルクリックします。

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

6. **[同期テーブル]** をクリックします。

7. 右ウィンドウ枠で、**[ULCustomer]** をダブルクリックします。

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

結果

同期スクリプトを確認しました。

次の手順

なし。

参照

- 「同期スクリプトの作成」『Mobile Link サーバ管理』
- 「同期論理のソースコード」『Mobile Link クイックスタート』
- 「同期の設計」『Mobile Link クイックスタート』
- 「Ultra Light クライアント」69 ページ
- 「接続スクリプト」『Mobile Link サーバ管理』
- 「テーブルスクリプト」『Mobile Link サーバ管理』

Ultra Light データベースのリファレンス

この項は、Ultra Light データベースのプロパティ、オプション、接続パラメータ、ユーティリティのリファレンスです。

Ultra Light 作成パラメータ

この項では、新しい Ultra Light データベースへの作成時に使用できる Ultra Light 作成パラメータについて説明します。

作成パラメータに加え、次の Ultra Light 接続パラメータを作成パラメータとして使用できます。

●「Ultra Light CE_FILE 接続パラメータ」170 ページ

●「Ultra Light DBF 接続パラメータ」

●「Ultra Light DBKEY 接続パラメータ」175 ページ

●「Ultra Light desktop 接続パラメータ」177 ページ

●「Ultra Light device 接続パラメータ」178 ページ

●「Ultra Light MIRROR_FILE 接続パラメータ」179 ページ

●「Ultra Light NT_FILE 接続パラメータ」181 ページ

●「Ultra Light PWD 接続パラメータ」182 ページ

●「Ultra Light RESERVE_SIZE 接続パラメータ」183 ページ

●「Ultra Light UID 接続パラメータ」186 ページ

メンバー

Ultra Light は、以下の作成パラメータに対応しています。

名前	説明
case	Ultra Light データベースの文字列を比較するときに大文字と小文字を区別するかどうかを設定します。 「Ultra Light case 作成パラメータ」139 ページを参照してください。
checksum_level	データベースのチェックサム検証のレベルを設定します。 「Ultra Light checksum_level 作成パラメータ」140 ページを参照してください。

名前	説明
collation	Ultra Light データベースで使用される照合順を設定します。このプロパティを UTF-8 プロパティと組み合わせて設定するかどうかで、データベースの文字セットが決まります。「Ultra Light 文字セット」26 ページ、「Ultra Light collation 作成パラメータ」141 ページおよび「Ultra Light utf8_encoding 作成パラメータ」163 ページを参照してください。
date_format	データベースから取り出した日付のデフォルトの文字列形式を設定します。「Ultra Light date_format 作成パラメータ」142 ページを参照してください。
date_order	年、月、日の日付の順序を解釈する方法を制御します。「Ultra Light date_order 作成パラメータ」145 ページを参照してください。
fips	Certicom 認定暗号化アルゴリズムを使用した AES FIPS 認定暗号化を制御します。「データベースセキュリティ」29 ページと「Ultra Light fips 作成パラメータ」146 ページを参照してください。
max_hash_size	デフォルトのインデックスハッシュサイズをバイト単位で設定します。「Ultra Light max_hash_size 作成パラメータ」147 ページを参照してください。
nearest_century	文字列から日付への変換で、2 桁の年の解釈を制御します。「Ultra Light nearest_century 作成パラメータ」149 ページを参照してください。
obfuscate	データベースのデータを難読化するかどうか制御します。難読化は単純暗号化です。「データベースセキュリティ」29 ページと「Ultra Light obfuscate 作成パラメータ」150 ページを参照してください。
page_size	データベースページサイズを定義します。「Ultra Light page_size 作成パラメータ」151 ページを参照してください。
precision	計算結果が小数の場合の最大桁数を指定します。「Ultra Light precision 作成パラメータ」153 ページを参照してください。
scale	計算結果が最大精度でトランケートされる場合の、小数点以下の最小桁数を指定します。「Ultra Light scale 作成パラメータ」154 ページを参照してください。
time_format	データベースから取り出した時刻の表示形式を設定します。「Ultra Light time_format 作成パラメータ」155 ページを参照してください。
timestamp_format	データベースから取り出したタイムスタンプの表示形式を設定します。「Ultra Light timestamp_format 作成パラメータ」157 ページを参照してください。

名前	説明
timestamp_increment	Ultra Light でのタイムスタンプのトランケート方法を指定します。「Ultra Light timestamp_increment 作成パラメータ」160 ページを参照してください。
timestamp_with_time_zone_format	このオプションでは、データベースから取り出した TIMESTAMP WITH TIME ZONE 値の表示形式を設定します。 「Ultra Light timestamp_with_time_zone_format 作成パラメータ」161 ページを参照してください。
utf8_encoding	Unicode 用の 8 ビットマルチバイトエンコードである UTF-8 フォーマットでデータをエンコードします。 「Ultra Light 文字セット」26 ページと「Ultra Light utf8_encoding 作成パラメータ」163 ページを参照してください。

参照

- 「作成パラメータの値へのアクセス」25 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ

Ultra Light case 作成パラメータ

Ultra Light データベースの文字列を比較するときに大文字と小文字を区別するかどうかを設定します。**case=respect** をプログラミングインタフェースの CreateDatabase メソッドの作成文字列パラメータに渡します (または、大文字と小文字が区別されるデータベースの場合は **case=ignore**)。

構文

ulinit **--case=***value database.udb*

指定可能な値

Ignore、Respect

デフォルト

Ignore

備考

データの大文字と小文字を区別するかしないかは、テーブル、インデックスなどに反映されます。Ultra Light データベースは、デフォルトでは、データは常に入力されたとおりの大文字と小文字で保持されていますが、比較において大文字と小文字を区別しません。識別子 (テーブル名、カラム名など) とユーザ ID は、データベースの大文字と小文字の区別に関係なく、常に大文字と小文字が区別されません。データベースの大文字と小文字の区別の設定に関係なく、パスワードは常に大文字と小文字が区別されます。

文字列の比較結果とソート順は、データベースで大文字と小文字を区別するかどうかによって異なります。

ただし、一部の照合では、識別子の大文字と小文字の区別を前提とする場合、特別な注意が必要です。特に、トルコ語の照合では、予期できない複雑なエラーが発生するような大文字と小文字の変換動作があります。最も一般的なエラーは、i または I という文字を含むシステムオブジェクトが見つからないというものです。

既存のデータベースの大文字と小文字を区別するかどうかは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードで大文字と小文字を区別するかどうかを設定できます。**[新しいデータベースの照合と文字セット]** ページで **[文字列の比較で大文字と小文字を区別する]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light の文字列」270 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

Ultra Light checksum_level 作成パラメータ

データベースのチェックサム検証のレベルを設定します。

構文

ulinit --checksum_level=*value*

指定可能な値

0, 1, 2

デフォルト

0

備考

チェックサムは、ディスク、フラッシュ、またはメモリに保管されているページのオフラインの破損を検出するために使用します。チェックサムによって、重要なページの破損が原因で他のデータが破損するのを防ぐことができます。選択するレベルによって、Ultra Light によって各

データベースページのチェックサムが計算、記録されてから、ページが記憶領域に書き込まれます。

計算されたチェックサムが、記憶領域から読み取られたページのチェックサムと一致しない場合は、ページの保管または取り出し中にページが変更されたか、壊れた可能性があります。チェックサムが一致しなかった場合、データベースでページがロードされるときに、Ultra Light によってデータベースが停止され、致命的なエラーがレポートされます。このエラーは解決できません。Ultra Light データベースを再作成し、データベースのエラーを iAnywhere にレポートしてください。

チェックサムが有効になっている Ultra Light データベースをアンロードしてから再ロードした場合、チェックサムのレベルは保持され、リストアされます。

checksum_level では、次の値がサポートされています。

- **0** データベースページにチェックサムを追加しない。
- **1** インデックスや同期ステータスのページなど、重要なデータベースページにチェックサムを追加し、ローページには追加しない。
- **2** すべてのデータベースページにチェックサムを追加する。

Sybase Central では、データベースを作成するウィザードでチェックサムの使用を設定できます。**[新しいデータベースの記憶領域設定]** ページで **[データベースページのチェックサムレベル]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- checksum_level データベースプロパティ： 「Ultra Light データベースプロパティ」187 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「Ultra Light のパフォーマンスに関するヒント」461 ページ
- 「Ultra Light page_size 作成パラメータ」151 ページ
- 「Ultra Light データベースへの接続」35 ページ

Ultra Light collation 作成パラメータ

データベースの照合を設定します。

構文

ulinit --collation=*value*

指定可能な値

文字列

デフォルト

932JPN

備考

また、次のコマンドを実行して、Ultra Light でサポートされている照合のリストを表示することもできます。

```
ulinit -Z
```

Sybase Central では、データベースを作成するウィザードで照合を設定できます。**[新しいデータベースの照合と文字セット]** ページで、デフォルトの照合 (932JPN) を選択するか、リストから別の照合をクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light でサポートされている照合」27 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- 「Ultra Light 文字セット」26 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

Ultra Light date_format 作成パラメータ

データベースから取り出した日付の表示形式を設定します。

Android スマートフォンの場合は、代わりに Connection.setOption(OPTION_DATE_FORMAT, value) を使用して、この作成パラメータを設定できます。Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』と Connection.OPTION_DATE_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

ulinit --date_format=*value*

指定可能な値

文字列

デフォルト

YYYY-MM-DD (ISO 日付形式仕様に準拠)

備考

データ型が DATE の値は、date_format 作成パラメータで設定されている形式で表されます。日付の値は、文字列で表すこともできます。値は、文字列に割り当ててから取り出します。

Ultra Light では日付の単位から日付が構築されます。日付の単位には、年、月、日、曜日、通し日数、時、分、秒 (その要素) があります。

デフォルトの日付形式と順序は ISO (YYYY-MM-DD) です。たとえば、「2006 年 1 月 7 日」はこの形式では次のようになります。2006-01-07. デフォルトの ISO 日付形式と順序を使用しない場合は、これらの日付の単位に別の形式と順序を指定します。

表示形式は次の記号を組み合わせた文字列です。

記号	説明
YY	2 桁の年。
YYYY	4 桁の年。
MM	2 桁の月、または 2 桁の分 (hh:mm のように、コロンの後ろに続く場合)
MMM[m...]	月を示す略式文字で、文字数は "m" の数に一致。大文字の M の場合、出力も大文字になります。
D	1 桁の曜日 (0 = 日曜日、6 = 土曜日)。
DD	2 桁の指定月の日。先頭の 0 は必要ありません。
DDD[d...]	曜日を示す略式文字。大文字の D の場合、出力も大文字になります。
HH	2 桁の時間。先頭の 0 は必要ありません。
NN	2 桁の分。先頭の 0 は必要ありません。
SS[.s...]	秒と秒の小数位。
AA	12 時間表記。正午前の時刻は AM で表します。
PP	12 時間表記。正午後の時刻は PM で表します。
JJJ	指定年の日 (1 ～ 366)。

既存のデータベースの日付形式は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

指定可能な値は、上の表に示す記号から構成されます。各記号は、設定された表示形式の日付のデータで置き換えられます。

略式文字については、指定した文字数が数えられます。必要な場合は、A.M. または P.M. のインジケータ (ローカライズ可能) が、指定した文字数に対応するバイト数までトランケートされます。

出力の大文字と小文字の制御　文字データを表す記号 (MMM など) では、出力の文字を次のように制御できます。

○記号を大文字で入力すると、大文字で表記されます。たとえば、MMM と入力すると、JAN と表記されます。

○記号を小文字で入力すると、小文字で表記されます。たとえば、mmm と入力すると、jan と表記されます。

○大文字と小文字を混ぜて入力すると、使用される言語に適切な文字が Ultra Light により選択されます。たとえば、Mmm と入力すると、英語では May、フランス語では mai と表記されます。

0 埋め込みの制御　数値データを表す記号では、記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

○記号をすべて大文字または小文字 (MM や mm など) で入力すると、0 埋め込みが行われます。たとえば、yyyy/mm/dd と入力すると、2002/01/01 と表記されます。

○大文字と小文字を混ぜて入力すると (Mm など)、0 埋め込みは行われません。たとえば、yyyy/Mm/Dd と入力すると、2002/1/1 と表記されます。

Sybase Central では、データベースを作成するウィザードで日付形式を設定できます。**[新しいデータベース作成パラメータ]** ページで **[日付形式]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light date_order 作成パラメータ」145 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

例

次の表は、date_format の設定と、2001 年 5 月 21 日 (木) に実行された SELECT CURRENT DATE 文からの出力を示します。

使用した date_format の構文	返された結果
YYYY/MM/DD/ddd	2001/05/21/thu
JJJ	141
mmm YYYY	may 2001
MM-YYYY	05-2001

Ultra Light date_order 作成パラメータ

日付形式の解釈を制御します。

Android スマートフォンの場合は、代わりに Connection.setOption(OPTION_DATE_ORDER, value) を使用して、この作成パラメータを設定できます。Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』と Connection.OPTION_DATE_ORDER 変数 [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

ulinit --date_order=*value*

指定可能な値

MDY、YMD、DMY

デフォルト

YMD (ISO 日付形式仕様に準拠)

備考

データ型が DATE の値は、date_format 作成パラメータで設定されている形式で表されます。日付の値は、文字列で表すこともできます。値は、文字列に割り当ててから取り出します。

Ultra Light では日付の単位から日付が構築されます。日付の単位には、年、月、日、曜日、通し日数、時、分、秒 (その要素) があります。

デフォルトの日付形式と順序は ISO (YYYY-MM-DD) です。たとえば、「2006 年 1 月 7 日」はこの形式では次のようになります。2006-01-07. デフォルトの ISO 日付形式と順序を使用しない場合は、これらの日付の単位に別の形式と順序を指定します。

既存のデータベースの日付順は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードで日付順を設定できます。**[新しいデータベース作成パラメータ]** ページで **[日付順]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light date_format 作成パラメータ」142 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

例

値によって、日付 10/11/12 の変換方法が異なります。

使用される構文	変換
MDY	1912 年 10 月 11 日
YMD	1910 年 11 月 12 日
DMY	1912 年 11 月 10 日

Ultra Light fips 作成パラメータ

新しいデータベースで AES または AES_FIPS の強力な暗号化方式のどちらを使用するかを制御します。このパラメータは、Ultra Light J または iPhone 用の Ultra Light ではサポートされていません。

構文

{**ulinit -a** | **ulload -c** } **--fips=***value* **--key=***value*

指定可能な値

Yes (AES_FIPS を使用)、No (AES を使用)

デフォルト

Yes

備考

データベースの暗号化タイプを変更するには、fips 作成パラメータまたは obfuscate 作成パラメータを使用してデータベースを作成し直す必要があります。データベースの暗号化キーを変更す

るには、新しい暗号化キーを Connection オブジェクトで指定します。データベースに接続するユーザは、接続するたびにキーを指定する必要があります。

Sybase Central では、データベースを作成するウィザードで暗号化を設定できます。**[新しいデータベースの記憶領域設定]** ページで **[AES FIPS アルゴリズム]** オプションをクリックします。暗号化キーを設定し、確認する必要もあります。

データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとして、このパラメータを設定します。

FIPS オプションを有効にしたデータベースを配備するには、プラットフォームに適切なライブラリをすべてコピーします。

参照

- 「単純暗号化と強力な暗号化」『SQL Anywhere サーバ データベース管理』
- 「データベースセキュリティ」29 ページ
- 「Ultra Light obfuscate 作成パラメータ」150 ページ
- 「Ultra Light DBKEY 接続パラメータ」175 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULChangeEncryptionKey メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- ULConnection.ChangeEncryptionKey メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

Ultra Light max_hash_size 作成パラメータ

デフォルトの最大インデックスハッシュサイズをバイト単位で設定します。

構文

ulinit --max_hash_size=*value*

指定可能な値

0 ～ 32 バイト

デフォルト

4 バイト

備考

ハッシュは、インデックスエントリのオプション部分で、インデックスのページに格納されます。ハッシュによって、インデックスカラムの実際のローの値が、インデックスの順序を保持し

たまま、それに相当する数値 (キー) に変換されます。キーのサイズと、実際の値がハッシュされる長さは、設定するハッシュのサイズで決まります。

Ultra Light では、ロー ID によって、テーブル内の実際のデータのローを見つけることができます。ロー ID は、常にインデックスエントリの一部です。ハッシュサイズを 0 に設定する (インデックスのハッシュを無効にする) と、インデックスエントリにはこのロー ID だけが含まれます。ハッシュサイズを 0 以外に設定すると、そのローの変換されたデータすべてまたはその一部を含むハッシュキーがロー ID とともにインデックスのページに格納されます。Ultra Light で必ずしもデータを検索、ロード、アンパックしてから実際のローの値を比較する必要がないため、これらのインデックスカラムに対するクエリのパフォーマンスを向上できます。

データベースの適切なデフォルトハッシュサイズを決めるには、クエリの効率とデータベースサイズのトレードオフを評価する必要があります。最大ハッシュ値が大きいほど、データベースのサイズが大きくなります。

Ultra Light では、このパラメータで指定されている最大値を上限として、カラムのデータ型に必要なバイト数のみが使用されます。デフォルトのハッシュサイズは、インデックスを作成するときにサイズを設定しなかった場合にだけ使用されます。デフォルトのハッシュサイズを 0 に設定すると、Ultra Light では、ローの値はハッシュされません。

既存のインデックスのハッシュサイズは変更できません。Sybase Central で Ultra Light の**インデックス作成ウィザード**を使用して、または CREATE INDEX 文か CREATE TABLE 文で WITH MAX SIZE 句を使用して新しいインデックスを作成するときにデフォルト値を変更できます。

カラムを DOUBLE、FLOAT、または REAL として宣言した場合は、ハッシュは使用されません。ハッシュサイズは常に無視されます。

Sybase Central では、データベースを作成するウィザードで最大ハッシュサイズを設定できます。**[新しいデータベースの記憶領域設定]** ページで **[インデックスの最大ハッシュサイズ]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light のパフォーマンスに関するヒント」461 ページ
- 「Ultra Light インデックス」56 ページ
- 「最適なハッシュサイズ制限」466 ページ
- 「CREATE INDEX 文 [Ultra Light]」428 ページ
- 「CREATE TABLE 文 [Ultra Light]」432 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

Ultra Light nearest_century 作成パラメータ

文字列から日付への変換で、2 桁の年の解釈を制御します。

Android スマートフォンの場合は、代わりに Connection.setOption(OPTION_NEAREST_CENTURY, value) を使用して、この作成パラメータを設定できます。Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』と Connection.OPTION_NEAREST_CENTURY 変数 [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

ulinit --nearest_century=*value*

指定可能な値

整数 (0 ～ 100)

デフォルト

50

備考

Ultra Light では、期待値が日付の値であれば、年が文字列内で 2 桁だけで表されている場合でも、文字列を日付に自動的に変換します。年が 2 桁の場合は、適切なロールオーバー値を設定する必要があります。この値より小さい 2 桁の年は 20*yy* に変換され、この値以上の年は 19*yy* に変換されます。

適切なロールオーバー値を決定するには、次の点を考慮します。

- **2 桁の年の使用**　年が 2 桁ではない場合は、基準年への変換は適用されません。設定した nearest_century 値よりも小さい 2 桁の年は 20*yy* に変換され、nearest_century 値以上の年は 19*yy* に変換されます。

  4 桁の年を格納して、間違った変換の問題を防ぐことをおすすめします。

- **統合データベースの互換性**　たとえば、従来の SQL Anywhere では、年に 1900 を加算していました。Adaptive Server Enterprise では基準年を使用するので、*yy* が 50 より小さい場合は 20*yy* になります。

- **日付の意味 : 過去または未来**　誕生年は、過去のことなので、小さいロールオーバー値が必要です。したがって、*yy* が 20 未満の年は 20*yy* に設定してください。これに対して、日付が有効期限の場合は、未来のことなので、大きい値の方が理にかなっています。

既存のデータベースの基準年は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードで基準年を設定できます。**[新しいデータベース作成パラメータ]** ページで**[基準年]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

Ultra Light obfuscate 作成パラメータ

データベース内のデータの難読化を制御します。

構文

{**ulinit -a** | **ulload -c** } **obfuscate=***value*

指定可能な値

ブール式

デフォルト

0 (データベースの難読化なし)

備考

難読化は、単純暗号化の一形式です。これにより、第三者がディスクユーティリティを使用してファイルを表示し、データベースのデータを解読することは困難になります。単純暗号化では、データベースの暗号化のためのキーは不要です。

正しい暗号化キーがないとデータベースにアクセスできないようにするには、強力な暗号化方式を使用する必要があります。

Sybase Central では、データベースを作成するウィザードで難読化を設定できます。**[新しいデータベースの記憶領域設定]** ページで **[単純暗号化を使用 (難読化)]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「データベースセキュリティ」29 ページ
- 「Ultra Light fips 作成パラメータ」146 ページ
- 「Ultra Light DBKEY 接続パラメータ」175 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

Ultra Light page_size 作成パラメータ

データベースページサイズをキロバイト単位で定義します。

構文

{ulinit | ulload } -c --page_size=*size***K**

指定可能な値

1K、2K、4K、8K、16K

デフォルト

4K

備考

ページサイズは、数字の後に「K」または「k」を付けて入力するか、またはバイトと等しい数字 (1024、2048、4096、8192、または 16384) を入力します。

Ultra Light データベースはページ単位で保管され、I/O 操作はすべてページ単位で行われます。選択したページサイズが、データベースのパフォーマンスまたはサイズに影響することがあります。

前述の値以外の値を使用すると、サイズはその値より大きい次の値に変更されます。単位を指定しない場合、デフォルトはバイトです。

プラットフォームの動的メモリの制約が大きい場合は、ページサイズを小さくして、同期メモリ要件による影響を低減させることを検討してください。

ページサイズを選択するときは、次のガイドラインに従ってください。

- **データベースサイズ**　大規模なデータベースでは、通常、より大きなページサイズの方が有利です。より大きなページではより多くの情報を保持できるため、特に、ページの半分以上のサイズのローを挿入する場合に領域を効率よく使用できます。ページが大きいほど、必要なページのスワップが少なくなります。

- **ローの数**　ロー (BLOB を除く) はページに収まる必要があるので、ページサイズによって、パックされたローの最大サイズと、各ページに格納できるローの数が決まります。1 ページを読み込んで 1 つのローの値を取得すると、次のいくつかのローの内容をメモリにロードできるという 2 次的な効果がある場合があります。

- **クエリのタイプ**　通常、ページサイズが小さいと、ランダムな場所から比較的少ない数のローを取り出すクエリに有利な傾向があります。これに対して、ページサイズが大きいと、テーブルの逐次スキャンを実行するクエリに有利です。

- **キャッシュサイズ**　ページサイズが大きいと、必要なキャッシュサイズも大きくなる場合があります。ダイナミックキャッシュサイジングを使用すると、Ultra Light で必要に応じてキャッシュ容量が設定されます。

- **インデックスエントリ**　ページサイズは、インデックスにも影響を与えます。データベースのページが大きいほど、保持できるインデックスエントリ数が多くなります。

- **デバイスメモリ**　特に、メモリが限られている小型のデバイスでデータベースを実行する場合に便利です。たとえば、1 MB のメモリには、1 ページを 1 KB とすると 1000 ページ格納できますが、1 ページ 4 KB であれば 250 ページしか格納できません。

既存のデータベースのページサイズは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードでページサイズを設定できます。**[新しいデータベースの記憶領域設定]** ページで、適切なバイト値をクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「ローのパックとテーブル定義」50 ページ
- 「Ultra Light インデックス」56 ページ
- 「Ultra Light case 作成パラメータ」139 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」168 ページ
- 「Ultra Light CACHE_MIN_SIZE 接続パラメータ」167 ページ
- 「UltraLite CACHE_MAX_SIZE 接続パラメータ」166 ページ
- 「Ultra Light RESERVE_SIZE 接続パラメータ」183 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

例

データベースのページサイズを 8 KB に設定するには、**page_size=8k** または **page_size=8192** を指定します。

```
ulinit test.udb -a --page_size=8k
```

Ultra Light precision 作成パラメータ

計算結果が小数の場合の最大桁数を指定します。

Android スマートフォンの場合は、代わりに Connection.setOption(OPTION_PRECISION, value) を使用して、この作成パラメータを設定できます。Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』と Connection.OPTION_PRECISION 変数 [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

ulinit precision=*value*

指定可能な値

整数 (1 ～ 127)

デフォルト

30

備考

小数点の位置は、数値の精度と位取りで決まります。精度は小数点の左右の合計桁数です。位取りは、計算結果が最大精度にトランケートされるときの小数点以下の最小桁数です。

適切な小数点の位置を決定するには、次の点を考慮します。

- **実行する計算のタイプ**　掛け算、割り算、足し算、引き算、集合関数はすべて結果が最大精度を超える可能性があります。

  たとえば、DECIMAL(8,2) と DECIMAL(9,2) を掛けると、結果には DECIMAL(17,4) が必要です。precision が 15 の場合、15 桁のみが結果に保持されます。scale が 4 の場合、結果は DECIMAL(15,4) になります。scale が 2 の場合、結果は DECIMAL(15.2) になります。どちらの場合も、オーバーフローエラーの可能性があります。

- **scale と precision の値の関係**　scale は、小数点以下の桁数を設定し、負の数や precision より大きい値にはできません。

既存のデータベースの精度は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

統合データベースとして Oracle データベースを使用している場合、すべての Ultra Light リモートと Oracle 統合データベースが精度として同じ値を使用する必要があります。

Sybase Central では、データベースを作成するウィザードで精度を設定できます。**[新しいデータベース作成パラメータ]** ページで **[精度]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

Ultra Light scale 作成パラメータ

計算結果が最大精度でトランケートされる場合の、小数点以下の最小桁数を指定します。

Android スマートフォンの場合は、代わりに Connection.setOption(OPTION_SCALE, value) を使用して、この作成パラメータを設定できます。Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』と Connection.OPTION_SCALE 変数 [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

ulinit --scale=*value*

指定可能な値

整数 (0 ～ 127)

デフォルト

6

備考

小数点の位置は、数値の精度と位取りで決まります。精度は小数点の左右の合計桁数です。位取りは、計算結果が最大精度にトランケートされるときの小数点以下の最小桁数です。

適切な小数点の位置を決定するには、次の点を考慮します。

- **実行する計算のタイプ**　掛け算、割り算、足し算、引き算、集合関数はすべて結果が最大精度を超える可能性があります。

  たとえば、DECIMAL(8,2) と DECIMAL(9,2) を掛けると、結果には DECIMAL(17,4) が必要です。precision が 15 の場合、15 桁のみが結果に保持されます。scale が 4 の場合、結果は DECIMAL(15,4) になります。scale が 2 の場合、結果は DECIMAL(15.2) になります。どちらの場合も、オーバーフローエラーの可能性があります。

- **scale と precision の値の関係**　scale は、小数点以下の桁数を設定し、負の数や precision より大きい値にはできません。

既存のデータベースの位取りは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

Sybase Central では、データベースを作成するウィザードで位取りを設定できます。**[新しいデータベース作成パラメータ]** ページで **[位取り]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light precision 作成パラメータ」153 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

例

DECIMAL(8,2) と DECIMAL(9,2) を掛けると、結果には DECIMAL(17,4) が必要です。precision が 15 の場合、15 桁のみが結果に保持されます。scale が 4 の場合、結果は DECIMAL(15,4) になります。scale が 2 の場合、結果は DECIMAL(15.2) になります。どちらの場合も、オーバーフローが発生する可能性があります。

Ultra Light time_format 作成パラメータ

データベースから取り出した時刻の表示形式を設定します。

Android スマートフォンの場合は、代わりに Connection.setOption(OPTION_TIME_FORMAT, value) を使用して、この作成パラメータを設定できます。Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』と Connection.OPTION_TIME_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

ulinit --time_format=*value*

指定可能な値

文字列 (下記の記号の組み合わせ)

デフォルト

HH:NN:SS.SSS

備考

Ultra Light では time_format 作成パラメータで設定した時間の単位を使用して時間が記述されます。時間の単位には、時、分、秒、ミリ秒があります。

時間の値は、文字列で表すこともできます。時間の値は、文字列変数に割り当ててから取り出します。

ISO (HH:MM:SS) がデフォルトの時間形式です。たとえば、真夜中はこの表示形式では 00:00:00 になります。デフォルトの ISO の時間形式を使用しない場合は、これらの時間の単位に別の表示形式と順序を指定します。

表示形式は次の記号を組み合わせた文字列です。

記号	説明
HH	2 桁の時間 (24 時間表記)。
NN	2 桁の分。
MM	2 桁の分 (HH:MM のようにコロンの後ろに続く場合)。
SS.SSS	2 桁の秒と小数。

既存のデータベースの時間形式は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

各記号は、設定された表示形式の時刻のデータで置き換えられます。数字ではなく文字を表す表示形式の記号は、大文字で入力でき、その場合、置き換えられる文字も大文字になります。表示形式の文字列で大文字と小文字を混ぜて使用すると、数字の前に 0 が付きません。

記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

- 記号をすべて大文字または小文字 (HH や hh など) で入力すると、0 埋め込みが行われます。たとえば HH:NN:SS と入力すると、01:01:01 と表記されます。
- 大文字と小文字を混ぜて入力すると (Hh など)、0 埋め込みは行われません。たとえば Hh:Nn:Ss と入力すると、1:1:1 と表記されます。

Sybase Central では、データベースを作成するウィザードで時間形式を設定できます。**[新しいデータベース作成パラメータ]** ページで **[時間形式]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light timestamp_format 作成パラメータ」157 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

例

3:30 PM にトランザクションが実行された場合、デフォルトの time_format 構文 HH:NN:SS.SSS では、結果は次のようになります。

15:30:55.0

Ultra Light timestamp_format 作成パラメータ

データベースから取り出したタイムスタンプの表示形式を設定します。

Android スマートフォンの場合は、代わりに Connection.setOption(OPTION_TIMESTAMP_FORMAT, value) を使用して、この作成パラメータを設定できます。Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』と Connection.OPTION_TIME_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

ulinit **--timestamp_format=***value*

指定可能な値

文字列

デフォルト

YYYY-MM-DD HH:NN:SS.SSS

備考

Ultra Light では、date_format プロパティと time_format 作成パラメータで設定した日付と時間の単位からタイムスタンプが作成されます。日付と時間には合計 7 つの単位があります (年、月、日、時、分、秒、ミリ秒)。

タイムスタンプの値は、文字列で表すこともできます。タイムスタンプの値は、文字列変数に割り当ててから取り出します。

一般に、タイムスタンプカラムによって、統合データベースと同期させるときにデータの整合性が維持されます。タイムスタンプを使用して、各ユーザが最後に同期した時刻を追跡することで、複数のリモートデータベース間でデータの同時更新がいつ発生したかを特定できます。

ヒント
タイムスタンプとその増分は、統合データベースと Ultra Light リモートデータベースで同じ値になるようにしてください。これらの作成パラメータを統合データベースと一致するように設定すると、タイムスタンプの不一致を回避できます。

表示形式は次の記号を組み合わせた文字列です。

記号	説明
YY	2 桁の年。
YYYY	4 桁の年。
MM	2 桁の月、または 2 桁の分 (HH:MM のように、コロンの後ろに続く場合)
MMM[m...]	月を示す略式文字−"m" の数と同数の文字。大文字の M の場合、出力も大文字になります。
D	1 桁の曜日 (0 = 日曜日、6 = 土曜日)。
DD	2 桁の指定月の日。先頭の 0 は必要ありません。
DDD[d...]	曜日を示す略式文字。大文字の D の場合、出力も大文字になります。
HH	2 桁の時間。先頭の 0 は必要ありません。
NN	2 桁の分。先頭の 0 は必要ありません。
SS.SSS	秒と秒の小数位。
AA	12 時間表記。正午前の時刻は AM で表します。
PP	12 時間表記。正午後の時刻は PM で表します。
JJJ	指定年の日 (1 ～ 366)。

既存のデータベースのタイムスタンプ形式は変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

指定可能な値は、上の表に示す記号から構成されます。各記号は、設定された表示形式の日付のデータで置き換えられます。

略式文字については、指定した文字数が数えられます。必要な場合は、A.M. または P.M. のインジケータ (ローカライズ可能) が、指定した文字数に対応するバイト数までトランケートされます。

文字データを表す記号 (MMM など) では、出力の文字を次のように制御できます。

- 記号をすべて大文字で入力すると、すべて大文字で表記されます。たとえば、MMM と入力すると、JAN と表記されます。
- 記号をすべて小文字で入力すると、すべて小文字で表記されます。たとえば、mmm と入力すると、jan と表記されます。
- 大文字と小文字を混ぜて入力すると、使用される言語に適切な文字が Ultra Light により選択されます。たとえば、Mmm と入力すると、英語では May、フランス語では mai と表記されます。

数値データを表す記号では、記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

- 記号をすべて大文字または小文字 (MM や mm など) で入力すると、0 埋め込みが行われます。たとえば、yyyy/mm/dd と入力すると、2002/01/01 と表記されます。
- 大文字と小文字を混ぜて入力すると (Mm など)、0 埋め込みは行われません。たとえば、yyyy/Mm/Dd と入力すると、2002/1/1 と表記されます。

Sybase Central では、データベースを作成するウィザードでタイムスタンプ形式を設定できます。**[新しいデータベース作成パラメータ]** ページで **[タイムスタンプ形式]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light timestamp_increment 作成パラメータ」160 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ
- 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバ管理』
- 「Ultra Light の同時実行性」475 ページ

例

2006 年 5 月 12 日 (金) の 3:30 PM にトランザクションが実行された場合、デフォルトの timestamp_format 構文 YYYY-MM-DD HH:NN:SS.SSS では、結果は次のようになります。

2006-05-12 15:30:55.0

Ultra Light timestamp_increment 作成パラメータ

タイムスタンプ値の精度を制限します。データベースにタイムスタンプが挿入されると、この増分に合わせてタイムスタンプはトランケートされます。

Android スマートフォンの場合は、代わりに Connection.setOption(OPTION_TIMESTAMP_INCREMENT, value) を使用して、この作成パラメータを設定できます。Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』と Connection.OPTION_TIMESTAMP_INCREMENT 変数 [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

ulinit timestamp_increment=*value*

指定可能な値

1 ～ 60000000 マイクロ秒

デフォルト

1 マイクロ秒

備考

1000000 マイクロ秒は 1 秒です。

既存のデータベースのタイムスタンプインクリメントは変更できません。変更する必要がある場合は、新しいデータベースを作成してください。

DEFAULT TIMESTAMP カラムがプライマリキーまたはロー識別子として使用されている場合、このインクリメントが役に立ちます。

Sybase Central では、データベースを作成するウィザードでタイムスタンプインクリメントを設定できます。**[新しいデータベース作成パラメータ]** ページで **[タイムスタンプインクリメント]** オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light timestamp_format 作成パラメータ」157 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULCreateDatabase メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ
- 「タイムスタンプベースのダウンロードの実装」『Mobile Link サーバ管理』
- 「Ultra Light の同時実行性」475 ページ

例

'2000/12/05 10:50:53:700' などの値を格納するには、この作成パラメータを 100000 に設定します。この値に設定すると、秒の部分は小数点以下 1 桁の後でトランケートされます。

Ultra Light timestamp_with_time_zone_format 作成パラメータ

データベースから取り出した TIMESTAMP WITH TIME ZONE 値の表示形式を設定します。

構文

ulinit --timestamp_with_time_zone_format=*value*

指定可能な値

文字列 (下記の記号の組み合わせ)

デフォルト

YYYY-MM-DD HH:NN:SS.SSS+HH:NN

備考

表示形式は次の記号を組み合わせた文字列です。

記号	説明
YY	2 桁の年
YYYY	4 桁の年
MM	2 桁の月、または 2 桁の分 (HH:MM のように、コロンの後ろに続く場合)
MMM[m...]	月を略した文字、"m" の数だけ文字を使用

記号	説明
DD	2 桁の日
DDD[d...]	曜日を略した文字
HH	2 桁の時間
NN	2 桁の分
SS.SSSSSS	秒と小数点第 6 位までの秒の小数。小数点以下 6 桁のタイムスタンプをサポートしないプラットフォームもあります。
AA	A.M. または P.M. (12 時間表記) - 24 時間表記の場合、AA と PP は省略します。
PP	P.M. (必要に応じて) (12 時間表記) − 24 時間表記の場合、AA と PP は省略します。
HH	2 桁の時間 (タイムゾーンオフセット)。
NN	2 桁の分 (タイムゾーンオフセット)。

各記号は、設定された表示形式の日付のデータで置き換えられます。

文字データを表す記号 (MMM など) では、出力の文字を次のように制御できます。

- 記号をすべて大文字で入力すると、すべて大文字で表記されます。たとえば、MMM と入力すると、JAN と表記されます。
- 記号をすべて小文字で入力すると、すべて小文字で表記されます。たとえば、mmm と入力すると、jan と表記されます。
- 大文字と小文字を混ぜて入力すると、使用される言語に適切な文字が Ultra Light により選択されます。たとえば、Mmm と入力すると、英語では May、フランス語では mai と表記されます。

文字データがマルチバイトである場合、各記号の長さはバイト数ではなく文字数を表します。たとえば MMM は 3 文字の月名を示しています。

数値データを表す記号では、記号に大文字を使用するか小文字を使用するかで 0 埋め込みを制御できます。

- 記号をすべて大文字または小文字 (MM や mm など) で入力すると、0 埋め込みが行われます。たとえば、yyyy/mm/dd と入力すると、2002/01/01 と表記されます。
- 大文字と小文字を混ぜて入力すると (Mm など)、0 埋め込みは行われません。たとえば、yyyy/Mm/Dd と入力すると、2002/1/1 と表記されます。

> **注意**
> 日付形式の順序を変更するように timestamp_with_time_zone_format オプションの設定を変更する場合は、同じ変更を反映するように date_order オプションも変更し、同様に date_order を変更する場合は timestamp_with_time_zone_format も変更してください。「date_order オプション」『SQL Anywhere サーバ データベース管理』を参照してください。

参照

- 「TIMESTAMP WITH TIME ZONE データ型」『SQL Anywhere サーバ SQL リファレンス』

Ultra Light utf8_encoding 作成パラメータ

Unicode 用の 8 ビットマルチバイトエンコードである UTF-8 フォーマットでデータをエンコードします。

構文

ulload - c --utf8_encoding=*value*

値

ブール式

デフォルト

1 (データベースの UTF-8 エンコードあり)

備考

UTF-8 文字は 1 ～ 4 バイトで表されます。他のマルチバイト照合の場合、1 または 2 バイトが使用されます。すべてのマルチバイト照合で、2 バイト以上の文字はアルファベットであると見なされます。これらの文字は、二重引用符を使用しないで識別子で使用できます。

Ultra Light データベース内の文字は、選択した照合において暗黙的に決定されるコードページ内の文字であるか、または UTF-8 でエンコードされています。UTF8BIN 照合を使用する Ultra Light データベースは、自動的に UTF-8 でエンコードされます。Ultra Light アプリケーションを配備するオペレーティングシステムで (ほとんどの Linux ディストリビューション、Windows Mobile、iPhone のように) UTF-8 または Unicode が使用される場合、またはデータベースに複数の言語の文字を格納する予定がある場合は、UTF-8 エンコードを使用してデータベースを作成してください。UTF-8 でエンコードされている文字を、Unicode がサポートされていない統合テーブルに同期しようとすると、ユーザエラーがレポートされます。

Sybase Central では、データベースを作成するウィザードで UTF-8 エンコーディングを選択できます。**[新しいデータベースの照合と文字セット]**ページで**[データベース内の文字に UTF-8 エンコードを使用する]**オプションをクリックします。

クライアントアプリケーションから、データベースマネージャクラスに対するデータベース作成メソッドの作成パラメータの 1 つとしてこのプロパティを設定します。

参照

- 「Ultra Light での文字セットのエンコードに関するプラットフォーム要件」27 ページ
- 「Ultra Light 文字セット」26 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.CreateDatabase メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「作成パラメータの値へのアクセス」25 ページ

Ultra Light 接続パラメータ

ここでは、Ultra Light データベースへの接続時に使用できる Ultra Light 接続パラメータについて説明します。

メンバー

Ultra Light は以下の接続パラメータに対応しています。

パラメータ名	説明
CACHE_MAX_SIZE, CACHE_MIN_SIZE, CACHE_SIZE	データベースキャッシュのサイズを定義します。 「Ultra Light CACHE_SIZE 接続パラメータ」168 ページ、「Ultra Light CACHE_MIN_SIZE 接続パラメータ」167 ページ、「UltraLite CACHE_MAX_SIZE 接続パラメータ」166 ページを参照してください。
COMMIT_FLUSH	コミットされたトランザクションが、コミット呼び出しの後にいつ記憶領域にフラッシュされるかを規定します。「Ultra Light COMMIT_FLUSH 接続パラメータ」171 ページを参照してください。
CON	現在の接続の名前を指定します。 「Ultra Light CON 接続パラメータ」172 ページを参照してください。

パラメータ名	説明
CE_FILE、DBF、desktop、device、および NT_FILE	データベースの作成時には、これらのパラメータはデータベースのロケーションを設定します。 その後の接続では、ファイルの場所を指定します。 単一プラットフォームのアプリケーションを作成している場合や、Ultra Light 管理ツールに接続している場合は、DBF を使用できます。さまざまなプラットフォーム固有データベースに接続する Ultra Light クライアントをプログラミングしている場合は、その他のプラットフォーム固有バージョンを使用します。 次の項を参照してください。 ● 「Ultra Light DBF 接続パラメータ」173 ページ ● 「Ultra Light CE_FILE 接続パラメータ」170 ページ ● 「Ultra Light desktop 接続パラメータ」177 ページ ● 「Ultra Light device 接続パラメータ」178 ページ ● 「Ultra Light NT_FILE 接続パラメータ」181 ページ
DBKEY	データベースの作成時には、このパラメータでデータベースの暗号化に使用する暗号化キーを設定します。 その後の接続では、データベースの暗号化に使用する暗号化キーを指定します。 「Ultra Light DBKEY 接続パラメータ」175 ページを参照してください。
DBN	ファイル名ではなく名前で実行中のデータベースを指定します。「Ultra Light DBN 接続パラメータ」176 ページを参照してください。
MIRROR_FILE	データベースミラーファイルの名前を指定します。 「Ultra Light MIRROR_FILE 接続パラメータ」179 ページを参照してください。
PWD	データベースの作成時に、このパラメータでユーザの初期パスワードを設定します。 その後の接続では、ユーザ ID のパスワードを指定します。 「Ultra Light PWD 接続パラメータ」182 ページを参照してください。
RESERVE_SIZE	実際にデータを挿入することなく、Ultra Light データベースが必要とするファイルシステム領域を事前に割り付けます。「Ultra Light RESERVE_SIZE 接続パラメータ」183 ページを参照してください。

パラメータ名	説明
START	Ultra Light エンジンの実行可能ファイルのロケーションを指定します。 「Ultra Light START 接続パラメータ」184 ページを参照してください。
TEMP_DIR	Ultra Light でテンポラリファイル (データベース名から導出される名前を持つ) を配置するフォルダ (すでに存在している必要がある) の名前を指定します。「Ultra Light TEMP_DIR 接続パラメータ」185 ページを参照してください。
UID	データベースの作成時には、このパラメータで初期ユーザ ID を設定します。 その後の接続では、データベースに対してユーザを識別します。 ユーザ ID は、Ultra Light データベースに格納されている最大 4 つのユーザ ID のうちの 1 つである必要があります。 「Ultra Light UID 接続パラメータ」186 ページを参照してください。

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ

UltraLite CACHE_MAX_SIZE 接続パラメータ

データベースキャッシュの最大サイズを定義します。Ultra Light ではキャッシュサイズは自動的に管理されるため、このパラメータを設定することは必須ではありません。

構文

CACHE_MAX_SIZE=*number*{ **k** | **m** }

デフォルト

デフォルトの最大キャッシュサイズはデバイスでは 20 MB、デスクトップでは 50 MB です。

備考

cache_max_size 接続パラメータで、ファイルキャッシュに割り当てる最大メモリ容量を指定します。デフォルトのサイズはバイト単位です。キロバイトまたはメガバイトの単位を指定するには、それぞれ k、m を使用します。

最大キャッシュサイズを超える場合、代わりにプラットフォームのキャッシュサイズの上限値が使用されます。Ultra Light では、データベースの実ファイルサイズを超える値にキャッシュサイズを拡大することはできません。

キャッシュ サイズの上限値としてデータベースのサイズを超える値を指定すると、ローのキャッシュに余分なスペースが使用されることがあります。

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

参照

- 「Ultra Light CACHE_MIN_SIZE 接続パラメータ」167 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」168 ページ
- 「Ultra Light cache_allocation オプション」191 ページ
- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「Ultra Light page_size 作成パラメータ」151 ページ
- 「Ultra Light RESERVE_SIZE 接続パラメータ」183 ページ
- Ultra Light for C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「Ultra Light データベースのキャッシュサイズの調整」461 ページ

例

次の接続文字列フラグメントによって、最大キャッシュサイズが 100 MB に設定されます。

```
"CACHE_MAX_SIZE=100m"
```

Ultra Light CACHE_MIN_SIZE 接続パラメータ

データベースキャッシュの最小サイズを定義します。Ultra Light ではキャッシュサイズは自動的に管理されるため、このパラメータを設定することは必須ではありません。

構文

CACHE_MIN_SIZE=*number*{ **k** | **m** }

デフォルト

デバイスのデフォルトのキャッシュ サイズは 256 KB です。デスクトップのデフォルトのキャッシュ サイズは 512 KB です。

備考

cache_max_size 接続パラメータで、ファイルキャッシュに割り当てる最小メモリ容量を指定します。デフォルトのサイズはバイト単位です。キロバイトまたはメガバイトの単位を指定するには、それぞれ k、m を使用します。

最小キャッシュサイズが最大キャッシュサイズよりも大きくなるように設定した場合、UltraLite からエラーメッセージが返され、接続が失敗します。

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

参照

- 「UltraLite CACHE_MAX_SIZE 接続パラメータ」166 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」168 ページ
- 「Ultra Light cache_allocation オプション」191 ページ
- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「Ultra Light page_size 作成パラメータ」151 ページ
- 「Ultra Light RESERVE_SIZE 接続パラメータ」183 ページ
- Ultra Light for C/C++：「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL：「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET：「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の接続文字列フラグメントによって、最小キャッシュサイズが 1 MB に設定されます。

```
"CACHE_MIN_SIZE=1m"
```

Ultra Light CACHE_SIZE 接続パラメータ

データベースキャッシュの初期サイズを定義します。Ultra Light ではキャッシュサイズは自動的に管理されるため、このパラメータを設定することは必須ではありません。

Android のスマートフォンの場合、代わりに Configuration.setPageSize を使用して、この接続パラメータを設定できます。Configuration.setPageSize メソッド [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

CACHE_SIZE=*number*{ **k** | **m** }

デフォルト

デフォルトの初期キャッシュサイズは、システムで使用可能なメモリ容量とデータベースの容量によって決まります。

備考

cache_max_size 接続パラメータで、ファイルキャッシュに割り当てる初期メモリ容量を指定します。このキャッシュは、データベースから最近使用したページをメモリに保持し、次に必要なときにこれらのページにすぐにアクセスできるようするために使用され、またページへの複数の変更を記憶領域に書き込む前に収集するためにも使用されます。キャッシュからのページへのアクセスは、記憶領域から読み込むよりも何倍も速くなります。さらに、記憶領域への書き込みは

コストがかさむため、複数の変更を 1 つの書き込みにグループ化することがパフォーマンスのためには重要になります。暗号化されたデータベースにはキャッシュによる利点もあります。復号化はページがキャッシュにロードされた場合にのみ発生し、暗号化はページが記憶領域に書き込まれる前に発生するからです。キャッシュが十分に大きい場合は、暗号化によるオーバヘッドは無視できる程度のものになります。

キャッシュの使用例として、同期を考えてみてください。Ultra Light がダウンロードを受け取っている間、データベースに行が挿入され、参照整合性検査が実行されます。挿入されると、行にはインデックスが付けられ、テーブル上の各インデックスに追加されます。このため、同期が行われている間、キャッシュは新しい行が保存されているページと現在のテーブルのインデックスページを保持する傾向があります。同期のパフォーマンスは、キャッシュのサイズが同期中のテーブルのページの適切な「ワーキングセット」を含むのに十分な大きさかどうかによって大きく左右されます。キャッシュが小さすぎると、行を挿入するために記憶領域から繰り返しインデックスページを読み込む必要が生じる場合があり、必要なインデックスページがキャッシュに収まる場合に比べて明らかにパフォーマンスが低下します。

デフォルトのサイズはバイト単位です。キロバイトまたはメガバイトの単位を指定するには、それぞれ k、m を使用します。

許容される最大キャッシュサイズを超える値を指定すると、自動的にプラットフォームのキャッシュサイズ上限が設定されます。

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

参照

- 「UltraLite CACHE_MAX_SIZE 接続パラメータ」166 ページ
- 「Ultra Light CACHE_MIN_SIZE 接続パラメータ」167 ページ
- 「Ultra Light cache_allocation オプション」191 ページ
- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「Ultra Light page_size 作成パラメータ」151 ページ
- 「Ultra Light RESERVE_SIZE 接続パラメータ」183 ページ
- Ultra Light for C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の接続文字列フラグメントによって、キャッシュサイズが 20 MB に設定されます。

```
"CACHE_SIZE=20m"
```

Ultra Light CE_FILE 接続パラメータ

新しい Ultra Light データベースの作成時に、この接続パラメータで新しいデータベースファイルの名前を指定します。

既存のデータベースへの接続を確立する場合は、このパラメータでデータベースを識別します。

Android スマートフォンの場合、代わりに Configuration.setDatabaseName を使用して、この接続パラメータを設定できます。Configuration.setDatabaseName メソッド [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

CE_FILE=*path¥ce-db*

デフォルト

DBF 接続パラメータ。

動作

1. DBF が指定されている場合、一致するデータベース (同一ファイル名) を探して、見つかったら接続し、見つからない場合は自動起動に進みます。
2. DBF が指定されている場合、必要であれば、データベースは自動起動されます。

備考

Microsoft Windows Mobile デバイスへの接続にも、他のプラットフォームへの接続にも同じ接続文字列を使用する Ultra Light クライアントアプリケーションの場合は、CE_FILE 接続パラメータを使用してください。

CE_FILE 接続パラメータは、DBF 接続パラメータより優先されます。Ultra Light 管理ツールから接続する場合、または接続オブジェクトが Windows Mobile データベースに接続する場合は、DBF 接続パラメータを使用します。

CE_FILE の値は、Windows Mobile のファイル名の要件を満たしている必要があります。データベースへの絶対パスを含める場合は、フォルダをすべて作成してから、このファイルのパスを設定する必要があります。フォルダは自動的には作成されません。

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

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「接続パラメータでの Ultra Light ファイルパスの指定」37 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」36 ページ
- 「Ultra Light DBF 接続パラメータ」173 ページ
- Ultra Light for C/C++： 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET： 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の例では、新しい接続を作成し、Windows デスクトップおよび Windows Mobile プラットフォームに異なるデータベースファイルを指定します。

```
Set Connection = DatabaseMgr.OpenConnection("DBF=d:¥Dbfile.udb;CE_FILE=¥myapp¥MyDB.udb")
```

Ultra Light COMMIT_FLUSH 接続パラメータ

コミットされたトランザクションが、コミット呼び出しの後にいつ記憶領域にフラッシュされるかを規定します。Ultra Light アプリケーションからコミットが呼び出されないと、フラッシュされません。

構文

COMMIT_FLUSH={ immediate | grouped | on_checkpoint }

デフォルト

immediate

備考

この接続パラメータは、ハードウェア障害またはクラッシュの後に、どのトランザクションをリカバリするかを定義します。論理オートコミット操作を単一のリカバリポイントとしてグループ化できます。

これらの操作をグループ化することで、Ultra Light のパフォーマンスを向上させることができますが、データのリカバリ性が低下します。トランザクションがコミットされた後であっても、トランザクションが記憶領域にフラッシュされる前にハードウェア障害またはクラッシュが発生すると、トランザクションが失われる可能性がわずかながらあります。

次のパラメータがサポートされます。

- **immediate** コミットされたトランザクションは、コミット操作が完了する前でも、コミット呼び出しが実行されるとすぐに記憶領域にフラッシュされます。

- **grouped**　コミットされたトランザクションは、設定したスレッショルドに達している場合にかぎり、コミット呼び出しがあった時点で記憶領域にフラッシュされます。トランザクションカウントスレッショルドを commit_flush_count データベースオプションを使用して設定するか、時間ベースのスレッショルドを commit_flush_timeout データベースオプションを使用して設定できます。

  設定された場合、commit_flush_count オプションと commit_flush_timeout オプションはどちらもコミットフラッシュのトリガとして動作し、先にスレッショルドに達した方がフラッシュをトリガします。フラッシュが実行されると、Ultra Light によってカウンタとタイマーはどちらも 0 に戻されます。その後、カウンタとタイマーのどちらか一方が再度スレッショルドに達するまで、両方がモニタリングされます。

- **on_checkpoint**　コミットされたトランザクションは、チェックポイント操作の時点で記憶領域にフラッシュされます。次のいずれかを使用してチェックポイントを実行できます。

  - CHECKPOINT 文。チェックポイントメソッドのない API では、この SQL 文を使用する必要があります。
  - Ultra Light Embedded SQL 用の ULCheckpoint 関数。
  - C++ コンポーネントの接続オブジェクトの Checkpoint メソッド。

参照

- 「単一のトランザクションまたはグループ化されたトランザクションのフラッシュ」478 ページ
- 「Ultra Light commit_flush_count オプション [テンポラリ]」192 ページ
- 「Ultra Light commit_flush_timeout オプション [テンポラリ]」193 ページ
- 「CHECKPOINT 文 [Ultra Light]」426 ページ
- ULCheckpoint メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.Checkpoint メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』

Ultra Light CON 接続パラメータ

接続に名前を付け、マルチ接続アプリケーションで簡単に切り替えができるようにします。

構文

CON=*name*

デフォルト

接続名なし

備考

CON 接続パラメータは、アプリケーション全体に対して設定されます。

複数の同時接続を確立し、その接続間で切り替えを行わない場合は、この接続パラメータは使用しないでください。

接続名はデータベース名とは異なります。

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

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「Ultra Light DBN 接続パラメータ」176 ページ
- Ultra Light for C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の接続文字列フラグメントによって、最初の接続名が MyFirstCon に設定されます。

```
"CON=MyFirstCon"
```

Ultra Light DBF 接続パラメータ

新しい Ultra Light データベースの作成時に、この接続パラメータで新しいデータベースファイルの名前を指定します。

既存のデータベースへの接続を確立する場合は、ロードして接続するデータベースファイルを指定します。

構文

DBF=*ul-db*

動作

1. 接続時に、データベースがすでに実行されているか確認します。DBN が指定されている場合、一致するデータベースを探して、見つかったら接続し、見つからない場合は自動起動に進みます。

2. DBF が指定されている場合、一致するデータベース (同一ファイル名) を探して、見つかったら接続し、見つからない場合は自動起動に進みます。

3. DBN も DBF も指定されておらず、1 つのデータベースが実行されている場合、そのデータベースに接続します。

4. DBF が指定されている場合、必要であれば、データベースは自動起動されます。DBN も指定されている場合、DBN が実行するデータベースの名前になり、それ以外の場合、基本のファイル名から名前が生成されます。

備考

これらのパラメータはエイリアスなので、DBF を同時に指定した場合は、最後に指定した方が優先します。

単一の接続文字列から、異なるデバイスにある複数のデータベースに接続する場合は、次のパラメータを使用して、プラットフォーム固有の代替パラメータを指定できます。

- CE_FILE
- desktop
- device
- NT_FILE

プラットフォーム固有の接続パラメータを指定した場合は、そのパラメータが DBF より優先されます。

DBF の値は、プラットフォームのファイル名の要件を満たしている必要があります。

Windows Mobile　Windows Mobile デバイスに配備する場合は、Ultra Light のユーティリティやウィザードを使用して、接続している Windows Mobile デバイス上の Ultra Light データベースを管理できます。Windows Mobile デバイス上のファイルを指定するには、絶対パスを指定し、**wce:¥** プレフィクスを使用してください。

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

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「接続パラメータでの Ultra Light ファイルパスの指定」37 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」36 ページ
- 「Ultra Light DBN 接続パラメータ」176 ページ
- 「Ultra Light CE_FILE 接続パラメータ」170 ページ
- 「Ultra Light NT_FILE 接続パラメータ」181 ページ
- Ultra Light for C/C++：「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL：「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET：「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

デスクトップのフォルダ *c:¥mydb* にインストールされているデータベース *MyULdb.udb* に接続するには、次の接続文字列を使用します。

```
"DBF=c:¥mydb¥MyULdb.udb"
```

接続している Windows Mobile デバイスの *UltraLite* フォルダに配備されている同じデータベースに接続するには、次の接続文字列を使用します。

```
"DBF=wce:¥UltraLite¥MyULdb.udb"
```

Ultra Light DBKEY 接続パラメータ

新しい Ultra Light データベースの作成時に、この接続パラメータでデータベースの暗号化キーを設定します。

既存のデータベースへの接続を確立する場合は、この接続パラメータで、データベースの暗号化キーを指定します。

構文

DBKEY=*string*

デフォルト

キーは指定されません。

備考

データベースの正しい暗号化キーを指定しないと、接続が失敗します。

暗号化キーを使用してデータベースを作成すると、データベースファイルは AES 256 ビットアルゴリズムまたは AES FIPS 認定アルゴリズムのどちらかを使用して、強力に暗号化されます。強力な暗号化を使用すると、巧妙で容赦ないデータへのアクセス試行に対抗するセキュリティが提供されます。ただし、強力な暗号化を使用すると、パフォーマンスに大きな影響を与える可能性があります。

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

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「データベースセキュリティ」29 ページ
- 「Ultra Light obfuscate 作成パラメータ」150 ページ
- Ultra Light for C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

Ultra Light DBN 接続パラメータ

アプリケーションが複数のデータベースに接続した場合に、データベースを名前で識別します。

構文

DBN=*db-name*

デフォルト

なし

動作

1. 接続時に、データベースがすでに実行されているか確認します。DBN が指定されている場合、一致するデータベースを探して、見つかったら接続し、見つからない場合は自動起動に進みます。

2. DBF が指定されている場合、一致するデータベース (同一ファイル名) を探して、見つかったら接続し、見つからない場合は自動起動に進みます。

3. DBN も DBF も指定されておらず、1 つのデータベースが実行されている場合、そのデータベースに接続します。

4. DBF が指定されている場合、必要であれば、データベースは自動起動されます。DBN も指定されている場合、DBN が実行するデータベースの名前になり、それ以外の場合、基本のファイル名から名前が生成されます。

備考

Ultra Light では、データベースを開いてからデータベース名が設定されます。データベースが開いたら、クライアントアプリケーションでファイルではなく名前を使用してデータベースに接続できます。

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

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「Ultra Light DBF 接続パラメータ」173 ページ
- Ultra Light for C/C++： 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET： 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次のパラメータを使用して Kitchener という実行中の Ultra Light データベースに接続することもできます。

```
DBN=Kitchener;DBF=cities.udb
```

Ultra Light desktop 接続パラメータ

新しい Ultra Light データベースの作成時に、この接続パラメータで新しいデータベースファイルの名前を指定します。

既存のデータベースへの接続を確立する場合は、このパラメータでデータベースを識別します。

構文

desktop:DBF=*path¥db* | **temp_dir=¥***Temp*

備考

Microsoft Windows または Mac デバイスへの接続にも同じ接続文字列を使用する Ultra Light クライアントアプリケーションの場合は、デスクトップ接続パラメータを使用してください。

アンダースコア「_」のほかに、コロン「:」が区切り文字と見なされます。プレフィクスのあるオプションは、プレフィクスのないオプションよりも優先されます。

temp_dir 接続パラメータを使用できるようになりました。このパラメータはディレクトリ (すでに存在する) を指定する必要があります。UL は、データベースファイルの横 (デフォルトの以前の動作) ではなく、指定されたディレクトリにテンポラリファイル (まだデータベースから取った名前を持つ) を置きます。高速 I/O 特性を持つテンポラリフォルダを指定すると、キャッシュサイズよりも大きなテンポラリテーブルなどのパフォーマンスが向上することがあります。また、実行時間が長いトランザクションは、テンポラリファイル内のかなりの領域を消費する可能性があります。

desktop の値は、Windows または Mac のファイル名の要件を満たしている必要があります。データベースへの絶対パスを含める場合は、フォルダをすべて作成してから、このファイルのパスを設定する必要があります。フォルダは自動的には作成されません。

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

動作

1. DBF が指定されている場合、一致するデータベース (同一ファイル名) を探して、見つかったら接続し、見つからない場合は自動起動に進みます。

2. DBF が指定されている場合、必要であれば、データベースは自動起動されます。

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「接続パラメータでの Ultra Light ファイルパスの指定」37 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」36 ページ
- 「Ultra Light DBF 接続パラメータ」173 ページ
- Ultra Light for C/C++： 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET： 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の例では、新しい接続を作成し、Windows デスクトップおよび Windows Mobile プラットフォームに異なるデータベースファイルを指定します。

```
"desktop:DBF=C:¥dir¥db.udb; device:DBF=¥SD Card¥db.udb; device:temp_dir=¥Temp;
device:cache_size=4M"
```

Ultra Light device 接続パラメータ

新しい Ultra Light データベースの作成時に、この接続パラメータで新しいデータベースファイルの名前を指定します。

既存のデータベースへの接続を確立する場合は、このパラメータでデータベースを識別します。

構文

device:DBF=*path¥db* | **temp_dir=¥***Temp*

備考

Microsoft Windows CE または iPhone デバイスへの接続にも同じ接続文字列を使用する Ultra Light クライアントアプリケーションの場合は、device 接続パラメータを使用してください。

アンダースコア「_」のほかに、コロン「:」が区切り文字と見なされます。プレフィクスのあるオプションは、プレフィクスのないオプションよりも優先されます。

temp_dir 接続パラメータを使用できるようになりました。このパラメータはディレクトリ (すでに存在する) を指定する必要があります。UL は、データベースファイルの横 (デフォルトの以前の動作) ではなく、指定されたディレクトリにテンポラリファイル (まだデータベースから取った名前を持つ) を置きます。高速 I/O 特性を持つテンポラリフォルダを指定すると、キャッシュサイズよりも大きなテンポラリテーブルなどのパフォーマンスが向上することがあります。また、実行時間が長いトランザクションは、テンポラリファイル内のかなりの領域を消費する可能性があります。

device の値は、Windows CE または iPhone のファイル名の要件を満たしている必要があります。データベースへの絶対パスを含める場合は、フォルダをすべて作成してから、このファイルのパスを設定する必要があります。フォルダは自動的には作成されません。

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

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「接続パラメータでの Ultra Light ファイルパスの指定」37 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」36 ページ
- 「Ultra Light DBF 接続パラメータ」173 ページ
- Ultra Light for C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の例では、新しい接続を作成し、Windows デスクトップおよび Windows Mobile プラットフォームに異なるデータベースファイルを指定します。

```
"desktop:DBF=C:¥dir¥db.udb; device:DBF=¥SD Card¥db.udb; device:temp_dir=¥Temp;
device:cache_size=4M"
```

Ultra Light MIRROR_FILE 接続パラメータ

(メインデータベースファイルへの書き込みが発行されると同時に) データベースへのすべての書き込みが発行される、データベースミラーファイルの名前を指定します。

構文

MIRROR_FILE=*path¥mirrorfile-db*

デフォルト

なし

備考

Ultra Light には、潜在的に信頼性が低いストレージシステムのフォールトトレランスを改善する基本的なデータベースファイルミラーリング機能があります。ミラーリングには、ミラーファイルが使用されます。データベースへのすべての書き込みは、メインデータベースファイルに対して発行されると同時にミラーファイルにも発行されます。したがって書き込みのオーバヘッドは 2 倍になり、読み込みのオーバヘッドには影響ありません。データベースファイルから破損したページが読み込まれた場合、ミラーファイルから読み込むことでページをリカバリできます。

ミラーリングは、ファイルベースのストアを使用するすべてのプラットフォームでサポートされています。

データベースの起動時に **mirror_file=** オプションが指定された場合、Ultra Light は指定されたファイルを開き、メインデータベースファイルと一致することを確認してから、処理を続行します。ミラーファイルがない場合は、この時点でメインファイルをコピーして作成されます。ミラーがデータベースファイルではないか破損していると、エラーが報告されます。この場合、ファイルが削除されるか、別のミラーが指定されるまで、データベースは起動しません。ミラーがデータベースと一致しない場合は、SQLE_MIRROR_FILE_MISMATCH が生成され、データベースは起動しません。破損したページがリカバリされると、警告 SQLE_CORRUPT_PAGE_READ_RETRY が生成されます。ミラーリングを使用していないか、ミラーファイルも破損している場合は、エラー SQLE_DEVICE_ERROR が生成され、データベースが停止します。

メディア障害から効果的に保護するために、ミラーファイルを使用しているときはページチェックサムを有効にしてください。ミラーリングを行っているかどうかに関係なく、ページチェックサムによって、ページのロード後すぐに Ultra Light でページの破損が検出可能になり、破損したデータの参照を回避できます。チェックサムを有効にするには checksum_level データベース作成オプションを指定します。ミラーファイルを使用しているときにチェックサムが有効になっていなかった場合、警告 SQLE_MIRROR_FILE_REQUIRES_CHECKSUMS が生成されます。

ミラーは、データベースファイルとまったく同じコピーなので、データベースとして直接起動できます。破損したページは ulvalid ユーティリティによって報告されます。

参照

- 「Ultra Light checksum_level 作成パラメータ」140 ページ
- 「Ultra Light データベース検証ユーティリティ (ulvalid)」235 ページ
- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「接続パラメータでの Ultra Light ファイルパスの指定」37 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」36 ページ
- Ultra Light for C/C++： 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET： 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の例では、新しい接続を作成し、ミラーファイルを作成します。

```
Connection = DatabaseMgr.OpenConnection("DBF=c:¥Dbfile.udb; UID=JDoe;PWD=ULdb;
MIRROR_FILE=c:¥test¥MyMirrorDB.udb")
```

Ultra Light NT_FILE 接続パラメータ

新しい Ultra Light データベースの作成時に、この接続パラメータで新しいデータベースファイルの名前を指定します。

既存のデータベースへの接続を確立する場合は、このパラメータでデータベースを識別します。

Android スマートフォンの場合、代わりに Configuration.setDatabaseName を使用して、この接続パラメータを設定できます。Configuration.setDatabaseName メソッド [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

NT_FILE=*path*¥*nt-db*

デフォルト

DBF 接続パラメータ。

備考

デスクトップデータベースへの接続にも、他のプラットフォームにあるデータベースへの接続にも同じ接続文字列を使用する Ultra Light クライアントアプリケーションの場合は、NT_FILE 接続パラメータを使用してください。

この接続パラメータは DBF パラメータより優先されます。Ultra Light 管理ツールから接続する場合、または接続オブジェクトがデスクトップデータベースに接続する場合は、DBF 接続パラメータを使用します。

NT_FILE の値は、Windows デスクトッププラットフォームのファイル名の要件を満たしている必要があります。

パスは絶対パスまたは相対パスのどちらでもかまいません。ファイル名にフォルダを含める場合は、フォルダをすべて作成してから、このファイルのパスを設定する必要があります。フォルダは自動的には作成されません。

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

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「接続パラメータでの Ultra Light ファイルパスの指定」37 ページ
- 「Ultra Light 管理ツールの接続パラメータの優先度」36 ページ
- 「Ultra Light DBF 接続パラメータ」173 ページ
- Ultra Light for C/C++： 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET： 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の例では、新しい接続を作成し、デスクトップおよび Windows Mobile プラットフォームに異なるデータベースファイルを指定します。

```
Connection = DatabaseMgr.OpenConnection("UID=JDoe;PWD=ULdb;
NT_FILE=c:¥test¥MyTestDB.udb;CE_FILE=¥database¥MyCEDB.udb")
```

Ultra Light PWD 接続パラメータ

新しい Ultra Light データベースの作成時に、この接続パラメータでデフォルトユーザの暗号化キーを設定します。

既存のデータベースに接続する場合は、認証に使用するユーザ ID のパスワードを識別します。

Android のスマートフォンの場合、代わりに Configuration.setPassword を使用して、この接続パラメータを設定できます。Configuration.setPassword メソッド [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

構文

PWD=*password*

デフォルト

UID と PWD の両方を設定しなかった場合、Ultra Light では **UID=DBA** と **PWD=sql** で接続が確立されます。

備考

すべてのデータベースユーザにはパスワードがあります。Ultra Light では、最大で 4 つのユーザ ID とパスワードの組み合わせがサポートされています。

パスワードは NULL または空の文字列に設定できます。

新しいユーザが作成されるか、または既存のユーザが各自のパスワードを変更すると、ランダムな 4 バイトのソルト値が生成されます。ソルト値は、パスワードハッシュを計算するときにユー

ザのパスワードに追加され、ハッシュとともにデータベースに格納されます。ソルト値を適用することによって、辞書攻撃に対する脆弱性が低下するだけでなく、同じパスワードを使用するユーザのパスワードハッシュが確実に異なるようになります。

この接続パラメータは暗号化されません。ただし、Ultra Light では、パスワードがハッシュされてから保存されるため、パスワードは Sybase Central を使用してだけ変更できます。

参照

- 「ユーザ」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「Ultra Light ユーザ」61 ページ
- 「Ultra Light UID 接続パラメータ」186 ページ
- Ultra Light for C/C++ : 「ユーザ認証」『Ultra Light C/C++ プログラミング』
- Ultra Light for C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「ユーザ認証」『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の部分的な接続文字列は、ユーザ ID DBA とパスワード sql を指定します。

```
"UID=DBA;PWD=sql"
```

次の部分的な接続文字列は、ユーザ ID DBA と空のパスワードを指定します。

```
"UID=DBA;PWD=""
```

Ultra Light RESERVE_SIZE 接続パラメータ

実際にデータを挿入することなく、Ultra Light データベースが必要とするファイルシステム領域を事前に割り付けます。ファイルシステム領域を予約すると、領域が他のファイルによって使用されることはありません。

構文

RESERVE_SIZE= *number*{ **k** | **m** | **g** }

デフォルト

0 (予約サイズなし)。

備考

0 からデータベースの最大サイズまでの任意の値を指定できます。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれ k、m、g のいずれかを使用してください。単位を指定しない場合、デフォルトはバイトです。

データベースをテストデータを使用して実行することによってデータベースサイズを確認し、Ultra Light の配備に適した予約サイズを選択してください。

RESERVE_SIZE の値がデータベースのサイズよりも小さい場合、このパラメータは無視されます。

ファイルシステム領域を予約すると、次の理由によりパフォーマンスが多少向上します。

●段階的なサイズの増加と比較して、断片化レベルが低減される。

●メモリ不足障害を防ぐ。

Ultra Light データベースはデータとメタデータで構成されているので、データベースサイズが大きくなるのは必要がある場合 (アプリケーションがデータベースを更新する場合) のみです。

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」168 ページ
- 「Ultra Light page_size 作成パラメータ」151 ページ
- Ultra Light for C/C++： 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET： 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の接続文字列フラグメントによって、予約サイズが 128 KB に設定され、起動時にシステムによってデータベース用にこのシステム領域が予約されるようになります。

```
"RESERVE_SIZE=128K"
```

Ultra Light START 接続パラメータ

Ultra Light エンジンの実行プログラムを起動します。このパラメータは、Android 用 Ultra Light ではサポートされていません。このパラメータは、エンジンが予測されるロケーションにない場合にのみ必要です。

構文

START=*path*¥uleng16.exe

備考

現在実行中でないエンジンに接続する場合は、StartLine (START) 接続パラメータだけを指定します。

スペースが含まれているパスには、引用符が必要です。パスにスペースが含まれていない場合、クライアントは SQLE_UNABLE_TO_CONNECT_OR_START を返します。

参照

- 「Ultra Light エンジンの起動」124 ページ
- 「Ultra Light エンジンユーティリティ (uleng16)」207 ページ
- 「Windows Mobile 用の Ultra Light データ管理コンポーネント」18 ページ
- Ultra Light for C/C++： 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET： 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次のコマンドは、*Program Files* フォルダの Ultra Light エンジンを起動します。

```
Start=¥"Program Files¥uleng16.exe"
```

または、文字列全体を一重引用符で囲んでこのパスを定義することもできます。

```
Start='"¥Program Files¥uleng16.exe"'
```

Ultra Light TEMP_DIR 接続パラメータ

Ultra Light でテンポラリファイル (データベース名から導出される名前を持つ) を配置するフォルダ (すでに存在している必要がある) の名前を指定します。

構文

TEMP_DIR=¥*path*

備考

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

デフォルトでは、テンポラリファイルは Ultra Light データベース自体と同じフォルダ (存在する場合) で管理されます。テンポラリファイルのファイル名はデータベースと同じになりますが、ファイルベースのプラットフォームの場合、ファイルの拡張子にチルダが含まれます。たとえば、サンプルデータベース *CustDB.udb* を実行すると、このデータベースファイルと同じフォルダに *CustDB.~db* というテンポラリファイルが作成されます。

高速 I/O 特性を持つテンポラリフォルダを指定すると、キャッシュサイズよりも大きなテンポラリテーブルなどのパフォーマンスが向上することがあります。また、実行時間が長いトランザクションは、テンポラリファイル内のかなりの領域を消費する可能性があります。

スペースが含まれているパスには、引用符が必要です。パスにスペースが含まれていない場合、クライアントは SQLE_UNABLE_TO_CONNECT_OR_START を返します。

参照

- Ultra Light for C/C++： 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL： 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET： 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の接続文字列フラグメントは、テンポラリファイルを *¥Temp* フォルダに配置します。

```
temp_dir=¥Temp;
```

Ultra Light UID 接続パラメータ

新しい Ultra Light データベースの作成時に、この接続パラメータでデータベースのデフォルトのユーザ ID を設定します。

既存のデータベースに接続する場合は、そのデータベースに接続に使用するユーザ ID を指定します。値には、データベースで認証されたユーザを指定します。

構文

UID=*user*

デフォルト

接続時に UID と PWD を設定しなかった場合、Ultra Light では **UID=DBA** と **PWD=sql** で接続が確立されます。

備考

すべてのデータベースユーザにはユーザ ID があります。Ultra Light は、ユーザ ID とパスワードの組み合わせを最大 4 つまでサポートします。

Ultra Light のユーザ ID は、Mobile Link のユーザ名やその他の SQL Anywhere のユーザ ID とは別です。ユーザ ID は一度作成すると変更できません。変更する必要がある場合は、ユーザ ID を削除し、新しい ID を追加してください。

UID は NULL または空の文字列に設定できません。ユーザ ID の最大長は 31 文字です。ユーザ ID の大文字と小文字は区別されません。

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

参照

- 「Ultra Light 接続文字列とパラメータ」35 ページ
- 「Ultra Light ユーザ」61 ページ
- Ultra Light for C/C++ : 「ユーザ認証」『Ultra Light C/C++ プログラミング』
- Ultra Light for C/C++ : 「Ultra Light データベースへの接続」『Ultra Light C/C++ プログラミング』
- ULDatabaseManager.OpenConnection メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「ユーザ認証」『Ultra Light C/C++ プログラミング』
- Ultra Light for Embedded SQL : 「Embedded SQL を使用した Ultra Light データベース接続」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「Ultra Light データベースの接続の設定」『Ultra Light .NET プログラミング』
- ULConnection.Open メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の接続文字列フラグメントは、データベースのユーザ ID DBA とパスワード sql を指定します。

```
"UID=DBA;PWD=sql"
```

Ultra Light データベースプロパティ

Ultra Light データベースのプロパティの値は、データベースの最初の作成時に定義されます。これらの値は、Ultra Light データベースを再作成するか、または対応するデータベースオプションを使用できる場合にはオプションを編集することによって、変更できます。

Ultra Light では、次のデータベースプロパティがサポートされます。

プロパティ	説明
cache_allocation	現在のキャッシュサイズを最小および最大設定のパーセンテージとして返します。「Ultra Light cache_allocation オプション」191 ページを参照してください。
CaseSensitive	大文字と小文字の区別のステータスを返します。データベースで大文字と小文字が区別される場合は、On を返します。それ以外の場合は、Off を返します。「Ultra Light case 作成パラメータ」139 ページを参照してください。

プロパティ	説明
CharSet	データベースの CHAR 文字セットを返します。データベースで使用される文字セットは、データベースの照合順と、データが UTF-8 コード化されているかどうかで決まります。 参照： ● 「Ultra Light utf8_encoding 作成パラメータ」163 ページ ● 「Ultra Light collation 作成パラメータ」141 ページ
ChecksumLevel	データベースのチェックサム検証のレベルを返します。値は、0 (チェックサムを追加しない)、1 (重要なページにチェックサムを追加する)、または 2 (すべてのページにチェックサムを追加する) のいずれかです。 「Ultra Light checksum_level 作成パラメータ」140 ページを参照してください。
Collation	データベースの照合順の名前を返します。 「Ultra Light collation 作成パラメータ」141 ページを参照してください。
commit_flush_count	コミットカウントスレッショルドを設定する commit_flush_count オプションの値を返します。 「Ultra Light commit_flush_count オプション [テンポラリ]」192 ページを参照してください。
commit_flush_timeout	時間間隔スレッショルドを設定する commit_flush_timeout オプションの値を返します。 「Ultra Light commit_flush_timeout オプション [テンポラリ]」193 ページを参照してください。
ConnCount	データベースとの接続の数を返します。値は動的です。現在存在する接続の数によって値は変化する可能性があります。Ultra Light では、最大 14 の同時データベース接続をサポートできます。
date_format	データベースで文字列変換に使用される日付形式を返します。 「Ultra Light date_format 作成パラメータ」142 ページを参照してください。
date_order	データベースで文字列変換に使用される日付順を返します。 「Ultra Light date_order 作成パラメータ」145 ページを参照してください。

プロパティ	説明
Encryption	データベース暗号化のタイプを返します。値は、None、Simple、AES、AES_FIPS のいずれかです。 データベースで使用される暗号化は、強力な暗号化 (AES か AES_FIPS) と DBKEY 作成パラメータ、または難読化 (単純暗号化) のどちらが設定されているかによって判断されます。 このプロパティを変更できるのは、最初の値が None (fips も難読化も使用されていない) の場合だけです。その場合は、使用している API の適切な関数またはメソッドを呼び出して、Connection オブジェクトに新しい暗号化キーを指定することで、暗号化キーを変更できます。この場合、値は AES に変更されます。これは、データベースの作成後に fips 作成パラメータを設定することができないためです。 次の項を参照してください。 ● ULChangeEncryptionKey メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』 ● ULConnection.ChangeEncryptionKey メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』 ●「データベースセキュリティ」29 ページ ●「Ultra Light fips 作成パラメータ」146 ページ ●「Ultra Light obfuscate 作成パラメータ」150 ページ ●「Ultra Light DBKEY 接続パラメータ」175 ページ
File	現在の接続のデータベースルートファイル名をパスを含めて返します。この値は、DBF 接続パラメータ値で指定されている値です。 次の項を参照してください。 ●「Ultra Light DBF 接続パラメータ」173 ページ
global_database_id	グローバルオートインクリメントカラムに使用される global_database_id オプションの値を返します。 「Ultra Light global_database_id オプション」194 ページを参照してください。
isolation_level	データベースの現在の独立性レベルを返します。この値は **read_committed** または **read_uncommitted** のいずれかになります。 「独立性レベル」43 ページを参照してください。
MaxHashSize	インデックスのハッシュに使用する、デフォルトの最大バイト数を返します。このプロパティは、インデックス単位で指定できます。 「Ultra Light max_hash_size 作成パラメータ」147 ページを参照してください。

プロパティ	説明
ml_remote_id	Mobile Link 同期のデータベースをユニークに識別する ml_remote_id オプションの値を返します。 「Ultra Light ml_remote_id オプション」195 ページを参照してください。
Name	現在の接続におけるデータベースの名前 (またはエイリアス) を返します。返される名前は、DBN 接続パラメータ値と一致します。DBN 接続パラメータを使用しなかった場合、返される名前はパスと拡張子のないデータベースファイルになります。 参照： ● 「Ultra Light DBN 接続パラメータ」176 ページ ● 「Ultra Light DBF 接続パラメータ」173 ページ
nearest_century	データベースで文字列変換に使用される基準年を返します。 「Ultra Light nearest_century 作成パラメータ」149 ページを参照してください。
PageSize	データベースのページサイズ (バイト単位) を返します。「Ultra Light page_size 作成パラメータ」151 ページを参照してください。
precision	データベースで文字列変換に使用される浮動小数点の精度を返します。 「Ultra Light precision 作成パラメータ」153 ページを参照してください。
scale	データベースによる文字列変換中に、計算結果が最大精度にトランケートされるときの小数点以下の最大桁数を返します。 「Ultra Light scale 作成パラメータ」154 ページを参照してください。
time_format	データベースで文字列変換に使用される時間形式を返します。 「Ultra Light time_format 作成パラメータ」155 ページを参照してください。
timestamp_format	データベースで文字列変換に使用されるタイムスタンプ形式を返します。「Ultra Light timestamp_format 作成パラメータ」157 ページを参照してください。
timestamp_increment	2 つのユニークなタイムスタンプ間のマイクロ秒単位の最小差を返します。 「Ultra Light timestamp_increment 作成パラメータ」160 ページを参照してください。

プロパティ	説明
timestamp_with_time_zone	TIMESTAMP WITH TIME ZONE 値のタイムスタンプ形式を返します。「Ultra Light timestamp_with_time_zone_format 作成パラメータ」161 ページを参照してください。

参照

- 「データベースプロパティの読み取り」39 ページ
- 「データベースオプションへのアクセス」39 ページ
- 「Ultra Light 作成パラメータ」137 ページ
- 「Ultra Light データベースオプション」191 ページ

Ultra Light データベースオプション

この項では、使用可能な Ultra Light データベースのオプションについて説明します。

Ultra Light データベースのオプション値は、データベースの最初の作成時に定義されますが、データベースへの接続中に変更できます。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「Ultra Light 作成パラメータ」137 ページ

Ultra Light cache_allocation オプション

明示的にキャッシュのサイズを変更します。この値は、最小から最大までの範囲のパーセンテージです。値 0 は最小サイズを表し、値 100 は最大サイズを表します。

指定可能な値

整数

デフォルト

なし

備考

cache_allocation プロパティは、現在のキャッシュサイズを最小キャッシュサイズと最大キャッシュサイズのパーセンテージ値として返します。

参照

- 「Ultra Light データベースプロパティ」187 ページ
- 「データベースオプションへのアクセス」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」168 ページ
- 「Ultra Light CACHE_MIN_SIZE 接続パラメータ」167 ページ
- 「UltraLite CACHE_MAX_SIZE 接続パラメータ」166 ページ
- ULConnection.SetDatabaseOptionInt メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULSetDatabaseOptionULong メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseSchema.SetDatabaseOption メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

Ultra Light commit_flush_count オプション [テンポラリ]

コミットカウントスレッショルドを設定します。このスレッショルドに達すると、コミットフラッシュが実行されます。

指定可能な値

整数

デフォルト

10

備考

トランザクションカウントを無効にする場合は、0 を使用します。トランザクションカウントを無効にすると、フラッシュがトリガされた際のコミット数に制限がなくなります。

commit_flush_count と commit_flush_timeout は、どちらも一時的なデータベースオプションです。これらのオプションは、データベースを起動するたびに設定する必要があります。オプションは、データベースが実行され続けている間は保持されます。これらは、COMMIT_FLUSH=grouped が接続文字列の一部として設定される場合にのみ必要です。

このオプションを設定し、接続文字列にグループ化されるように COMMIT_FLUSH 接続パラメータを設定すると、どちらかのスレッショルドに達するとフラッシュがトリガされます。フラッシュが実行されると、Ultra Light によってカウンタとタイマーは**どちらも** 0 に戻されます。その後、どちらか一方がスレッショルドに達するまで、両方ともモニタリングされます。

コミットフラッシュオプションを設定するうえで重要な考慮事項は、コミットされたトランザクションがフラッシュされるまでの遅延によって、データのリカバリ性はどれ程のリスクを受けるかということです。トランザクションがコミットされた後であっても、トランザクションが失われる可能性がわずかながら存在します。コミット後に深刻なハードウェア障害が発生した場合、それがトランザクションが記憶領域にフラッシュされる前ならば、トランザクションはリカバリ時にロールバックされます。遅延を長くするほど、Ultra Light のパフォーマンスは向上します。適切なカウントのスレッショルドの選択は慎重に行ってください。

クライアントアプリケーションから commit_flush_count オプションを設定するには、オプションを設定するときに、使用しているプログラミングインタフェースのデータベースオプション設定機能を使用するか、SET OPTION SQL 文を使用します。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- 「単一のトランザクションまたはグループ化されたトランザクションのフラッシュ」478 ページ
- 「Ultra Light commit_flush_timeout オプション [テンポラリ]」193 ページ
- 「Ultra Light COMMIT_FLUSH 接続パラメータ」171 ページ
- ULConnection.SetDatabaseOptionInt メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULSetDatabaseOptionULong メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseSchema.SetDatabaseOption メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

Ultra Light commit_flush_timeout オプション [テンポラリ]

時間間隔スレッショルドを設定します。このスレッショルドに達すると、グループ化されたコミットフラッシュが実行されます。

指定可能な値

整数 (ミリ秒単位)

デフォルト

10000 ミリ秒

備考

時間スレッショルドを無効にする場合は、0 を使用します。

commit_flush_count と commit_flush_timeout は、どちらも一時的なデータベースオプションです。これらのオプションは、データベースを起動するたびに設定する必要があります。オプションは、データベースが実行され続けている間は保持されます。これらは、COMMIT_FLUSH=grouped が接続文字列の一部として設定される場合にのみ必要です。

このオプションを commit_flush_timeout オプションとともに設定し、COMMIT_FLUSH 接続パラメータを grouped に設定した場合は、どちらかのスレッショルドに達するとフラッシュがトリガされます。フラッシュが実行されると、Ultra Light によってカウンタとタイマーは**どちらも** 0 に戻されます。その後、どちらか一方がスレッショルドに達するまで、両方ともモニタリングされます。

コミットフラッシュオプションを設定するうえで重要な考慮事項は、コミットされたトランザクションがフラッシュされるまでの遅延によって、データのリカバリ性はどれ程のリスクを受けるかということです。トランザクションがコミットされた後であっても、トランザクションが失われる可能性がわずかながら存在します。コミット後に深刻なハードウェア障害が発生した場合、

それがトランザクションが記憶領域にフラッシュされる前ならば、トランザクションはリカバリ時にロールバックされます。遅延を長くするほど、Ultra Light のパフォーマンスは向上します。適切なタイムアウトのスレッショルドの選択は慎重に行ってください。

クライアントアプリケーションから commit_flush_timeout オプションを設定するには、オプションを設定するときに、使用しているプログラミングインタフェースのデータベースオプション設定機能を使用するか、SET OPTION SQL 文を使用します。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- 「Ultra Light commit_flush_count オプション [テンポラリ]」192 ページ
- 「Ultra Light COMMIT_FLUSH 接続パラメータ」171 ページ
- ULConnection.SetDatabaseOptionInt メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULSetDatabaseOptionULong メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseSchema.SetDatabaseOption メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

Ultra Light global_database_id オプション

データベースの ID 番号を設定します。

指定可能な値

負でない一意の整数

デフォルト

特定のグローバルオートインクリメントカラムのデフォルト値の範囲は、$pn + 1 \sim p(n + 1)$ です。ここで、p はカラムの分割サイズを、n はグローバルデータベース ID 番号を示します。

備考

グローバル ID は、Mobile Link サーバとの同期時にプライマリキーの一意性を維持するために GLOBAL AUTOINCREMENT カラムの開始値を設定します。グローバル ID を設定してからデフォルト値を割り当ててください。テーブルにローが追加され、まだ値が設定されていない場合は、Ultra Light によって、global_database_id の値と分割サイズを組み合わせてカラムの値が生成されます。

このオプションが設定されると、Ultra Light はコミット操作を実行します。

アプリケーションを配備するときには、Mobile Link サーバとの同期のために、各データベースに異なる ID 番号を割り当てる必要があります。既存のデータベースのグローバル ID はいつでも変更できます。

クライアントアプリケーションから global_database_id オプションを設定するには、使用しているプログラミングインタフェースのデータベースオプション設定機能を使用するか、SET OPTION SQL 文を使用します。

参照

- 「GLOBAL AUTOINCREMENT」『Mobile Link サーバ管理』
- 「データベースオプションへのアクセス」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- 「Ultra Light の GLOBAL AUTOINCREMENT カラム」71 ページ
- ULConnection.SetDatabaseOptionInt メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULSetDatabaseID メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.DatabaseID プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.setDatabaseId メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_DATABASE_ID 変数 [BlackBerry] [Ultra Light J]『Ultra Light Java プログラミング』

例

Ultra Light データベースのカラムを 3001 から 4000 までオートインクリメントするには、グローバル ID を 3 に設定します。

```
SET OPTION global_database_id=3
```

Ultra Light isolation_level オプション

独立性レベルは、あるトランザクションの操作が、同時に処理されている別のトランザクションの操作からどの程度参照できるかを定義します。Ultra Light では、オートコミットモードでの接続には、デフォルトの独立性レベルである read_committed (コミットされた呼び出し) が使用されます。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- 「独立性レベル」43 ページ
- ULConnection.SetDatabaseOption メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULConnection.BeginTransaction メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- 「BlackBerry スマートフォンでのデータ同期」『Ultra Light Java プログラミング』

Ultra Light ml_remote_id オプション

「リモート ID」 は、Ultra Light データベースのユニークな識別子で、Mobile Link で同期用のデータベースを識別するために使用されます。リモート ID は、すべてのリモート Mobile Link クライアント間でユニークであれば、意味のある任意の文字列にできます。この ID は NULL に設定することもできます (NULL は初期値です)。同期中、リモート ID が NULL の場合、Ultra Light はそれを生成された GUID に割り当てます。

複数のデバイスへの分散を目的として同期を使用して Ultra Light データベースに事前に値を格納する場合、各データベースのリモート ID をユニークな ID にするために、分散前にリモート ID を NULL にリセットする必要があります。分散時に、新しいユニークなリモート ID を明示的

に設定できます。また、リモート ID を NULL のままにしておくと、Ultra Light が新しいユニークな値を自動的に生成します。

指定可能な値

Mobile Link による同期時にデータベースをユニークに識別する任意の値

デフォルト

NULL

備考

Mobile Link ではリモート ID を使用して、リモートデータベースの同期情報を保存できます。リモート ID が設定されると、Mobile Link ユーザ名は一意である必要がなくなります。リモート ID は、複数の Mobile Link ユーザが同じ Ultra Light データベースを同期する場合に特に便利です。この場合、同期スクリプトでは、ユーザ名だけでなく、リモート ID を参照してください。

このオプションが設定されると、Ultra Light はコミット操作を実行します。

クライアントアプリケーションから ml_remote_id オプションを設定するには、使用しているプログラミングインタフェースのデータベースオプション設定機能を使用するか、SET OPTION SQL 文を使用して設定します。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- 「リモート ID」『Mobile Link クライアント管理』
- 「User Name 同期パラメータ」110 ページ
- ULConnection.SetDatabaseOption メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULSetDatabaseOptionString メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULDatabaseSchema.SetDatabaseOption メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_ML_REMOTE_ID 変数 [BlackBerry] [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light ユーティリティ

Ultra Light には、コマンドプロンプトでデータベースの基本的な管理作業を行うことができる一連のユーティリティがあります。これらのユーティリティの多くは、SQL Anywhere サーバのユーティリティと機能が似ています。ただし、オプションの使用方法が異なる場合があります。Ultra Light でのオプションの実装方法については、必ず Ultra Light のリファレンスを参照してください。

注意
この項に示すユーティリティのオプションでは、特に明記されていないかぎり、大文字と小文字が区別されます。オプションはここに示すとおり**正確に**入力してください。

サポートされている終了コード

ulload、ulsync、ulunload の各ユーティリティは、処理が正常に終了したかどうかを示す終了コードを返します。0 は処理が成功したことを示します。他の値は処理が失敗したことを示します。

終了コード	ステータス	説明
0	EXIT_OKAY	処理は成功した。
1	EXIT_FAIL	処理は失敗した。
3	EXIT_FILE_ERROR	データベースが見つからない。
4	EXIT_OUT_OF_MEMORY	デバイスの動的メモリが足りない。
6	EXIT_COMMUNICATIONS_FAIL	Ultra Light エンジンとの通信中に通信エラーが発生した。
9	EXIT_UNABLE_TO_CONNECT	指定された UID または PWD が無効であるため、データベースに接続できない。
12	EXIT_BAD_ENCRYPT_KEY	暗号化キーがないか、無効である。
13	EXIT_DB_VER_NEWER	データベースのバージョンに互換性がないことが検出された。データベースを新しいバージョンにアップグレードする必要がある。
255	EXIT_USAGE	コマンドラインオプションが無効である。

Ultra Light 用 Interactive SQL ユーティリティ (dbisql)

SQL 文を実行し、データベースに対してスクリプトファイルを実行します。

構文

dbisql -c "*connection-string*" [*options*] [*dbisql-statement* | *dbisql-script-file*]

dbisql -c "*connection-string*" **-ul** [*options*] [*dbisql-statement* | *dbisql-script-file*]

dbisql-statement:1 つの SQL 文、またはコマンドデリミタによって区切られる一連の SQL 文。

オプション	説明
@*data*	指定された環境変数または設定ファイルからオプションを読み込みます。 同じ名前の環境変数と設定ファイルが存在する場合は、環境変数が使用されます。「設定ファイル」『SQL Anywhere サーバ データベース管理』を参照してください。 設定ファイル内のパスワードなどの情報を保護する場合は、ファイル難読化ユーティリティを使用して、設定ファイルの内容を難読化します。「ファイル非表示ユーティリティ (dbfhide)」『SQL Anywhere サーバ データベース管理』を参照してください。
-c *"keyword=value; ..."*	接続パラメータを指定します。Interactive SQL が接続できない場合は、接続パラメータを入力するウィンドウが表示されます。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。「接続パラメータ」『SQL Anywhere サーバ データベース管理』を参照してください。
-d *delimiter*	コマンドデリミタを指定します。デリミタを囲む引用符は省略可能ですが、コマンドシェル自体がデリミタを特別な方法で解釈するときは必ず指定します。 このオプションは、command_delimiter オプションの設定を上書きします。「command_delimiter オプション [Interactive SQL]」『SQL Anywhere サーバ データベース管理』を参照してください。

オプション	説明
-d1	ユーザが明示的に実行するすべての文をコマンドウィンドウ (STDOUT) にエコーします。これによって、SQL スクリプトのデバッグ、または Interactive SQL が長文の SQL スクリプトを処理しているときに有用なフィードバックが提供されます。(最後の文字は数値の 1 であり、L の小文字ではありません)。このオプションは、Interactive SQL をコマンドラインプログラムとして実行している場合のみ使用できます。
-datasource *DSN-name*	接続先の ODBC データソースを指定します。
-f *filename*	[SQL 文] ウィンドウ枠で *filename* というファイルを (実行しないで) 開きます。 -f オプションが指定されている場合、-c オプションは無視されます。つまり、データベースへの接続は確立されません。 ファイル名は引用符で囲んでも囲まなくても構いませんが、ファイル名にスペースが含まれている場合は必ず引用符で囲む必要があります。 そのファイルが存在しない場合、またはファイルではなく実際にはディレクトリである場合は、Interactive SQL がエラーメッセージを出力して終了します。 ファイル名に完全なドライブとパスの仕様が含まれていないときは、現在のディレクトリが基準として想定されます。 このオプションは、Interactive SQL をウィンドウベースのアプリケーションとして実行している場合のみサポートされます。
-host *hostname*	データベースサーバを実行するコンピュータの*ホスト名*または IP アドレスを指定します。現在のコンピュータを表す localhost という名前を使用できます。

オプション	説明
-nogui	Interactive SQL をコンソールアプリケーションとして実行します。このとき、ウィンドウベースのユーザインタフェースは使用しません。これは、バッチ処理に便利です。 *dbisql-statement* または *dbisql-script-file* のいずれかを指定する場合、-nogui が使用されます。 このモードのとき、Interactive SQL は、処理の成功または失敗をプログラム終了コードを設定することで示します。Windows オペレーティングシステムでは、プログラム終了コードに対して環境変数 ERRORLEVEL が設定されます。「ソフトウェアコンポーネントの終了コード」『SQL Anywhere サーバ プログラミング』を参照してください。
-onerror { **continue** | **exit** }	スクリプトファイルから文を読み出し中にエラーが起こった場合の事象を制御します。これは、Interactive SQL をバッチ処理で使用するときに便利です。このオプションは、on_error 設定を上書きします。「on_error オプション [Interactive SQL]」『SQL Anywhere サーバ データベース管理』を参照してください。 *behavior* には次のいずれかの値を定義できます。 ● **Continue**　エラーは無視され、Interactive SQL は文の実行を継続します。 ● **Exit**　Interactive SQL は終了します。
-q	出力メッセージを表示しません。ユーティリティをクワイエットモードで実行するように設定します。これは、文またはスクリプトファイルを使って Interactive SQL を起動したときにのみ便利です。このオプションを指定した場合、エラーメッセージは表示されますが、次の情報は表示されません。 ●警告およびその他の致命的でないメッセージ ●結果セットの出力

オプション	説明
-ul	デフォルトとして Ultra Light データベースを指定します。Interactive SQL では、接続するデータベースのタイプに応じて使用可能なオプションがカスタマイズされます。 Interactive SQL では、デフォルトで SQL Anywhere データベースに接続すると見なされます。-ul オプションを指定すると、デフォルトが Ultra Light データベースに変更されます。デフォルトに設定されているデータベースのタイプに関係なく、**[接続]** ウィンドウの **[データベースタイプを変更]** ドロップダウンリストからデータベースのタイプを選択することで、SQL Anywhere または Ultra Light のデータベースに接続できます。 Interactive SQL から Ultra Light データベースへの接続については、「Ultra Light 用 Interactive SQL ユーティリティ (dbisql)」197 ページを参照してください。
-version	Interactive SQL のバージョン番号を表示します。Interactive SQL 内から **[ヘルプ]** → **[Interactive SQL について]** をクリックして、バージョン番号を参照することもできます。
-x	文をスキャンしますが、実行しません。長いスクリプトファイルの構文エラーをチェックする場合に有用です。 SQL 文の詳細については、「SQL 文」『SQL Anywhere サーバ SQL リファレンス』を参照してください。
dbisql-statement | *dbisql-script-file*	SQL 文または指定された *dbisql-script-file* を実行します。 *dbisql-statement* または *dbisql-script-file* を指定しなかった場合、Interactive SQL は対話型モードになり、コマンドをコマンドウィンドウに入力できます。

備考

Interactive SQL を使用して、データベースのブラウズ、SQL 文の実行、およびスクリプトファイルの実行を行うことができます。また、次の項目に関するフィードバックも提供します。

- 影響を受けたローの数
- 各文に必要な時間
- クエリの実行プラン
- エラーメッセージ

Interactive SQL を使用して、SQL Anywhere データベース、Ultra Light データベース、SAP Sybase IQ データベース、SAP HANA データベース、または汎用の ODBC データベースに接続できます。

Windows の場合、実行プログラムが 2 つあります。

1. バッチスクリプトでは、*dbisql.exe* ではなく、*dbisql* または *dbisql.com* を呼び出す必要があります。実行プログラム *dbisql.com* は、コンソールアプリケーションとしてリンクされています。

2. 実行プログラム *dbisql.exe* はウィンドウアプリケーションとしてリンクされており、起動元のコマンドシェルがブロックされません。*dbisql.exe* をバッチファイルから実行した場合、標準出力または標準のエラーファイルに出力が送信されません。

Interactive SQL のデフォルトのエンコードは、default_isql_encoding オプションでも一時的に設定できます。「default_isql_encoding オプション [Interactive SQL]」『SQL Anywhere サーバ データベース管理』を参照してください。

INPUT、OUTPUT、または READ 文の ENCODING 句を使用して、ファイルの読み込みまたは書き込みをするときに使用するエンコードを指定できます。次の項を参照してください。

- 「INPUT 文 [Interactive SQL]」『SQL Anywhere サーバ SQL リファレンス』
- 「OUTPUT 文 [Interactive SQL]」『SQL Anywhere サーバ SQL リファレンス』
- 「READ 文 [Interactive SQL]」『SQL Anywhere サーバ SQL リファレンス』

終了コードは、0 (成功) または 0 以外の値 (失敗) です。0 以外の終了コードが設定されるのは、(SQL 文またはスクリプトファイル名を指定したコマンドラインによって) Interactive SQL をバッチモードで実行した場合だけです。「ソフトウェアコンポーネントの終了コード」『SQL Anywhere サーバ プログラミング』を参照してください。

コマンドプロンプトモードのとき、Interactive SQL は、処理の成功または失敗を、プログラム終了コードを設定することで示します。Windows オペレーティングシステムでは、プログラム終了コードに対して環境変数 ERRORLEVEL が設定されます。

Interactive SQL で *reload.sql* ファイルを実行しており、データベースが暗号化されている場合は、パラメータとして暗号化キーを指定する必要があります。READ 文にキーを指定していない場合、Interactive SQL でキーの入力を要求するプロンプトが表示されます。

次の方法で Interactive SQL を開始できます。

- Sybase Central で **[ファイル]** » **[Interactive SQL を開く]** をクリックする。
- **[スタート]** メニューから、**[スタート]** » **[プログラム]** » **[SQL Anywhere 16]** » **[管理ツール]** » **[Interactive SQL]** をクリックする。
- コマンドプロンプトで dbisql コマンドを使用する。

参照

- 「Interactive SQL の SQL 文」『SQL Anywhere サーバ データベース管理』
- 「Interactive SQL」『SQL Anywhere サーバ データベース管理』
- 「設定ファイル」『SQL Anywhere サーバ データベース管理』
- 「ファイル非表示ユーティリティ (dbfhide)」『SQL Anywhere サーバ データベース管理』
- 「INPUT 文 [Interactive SQL]」『SQL Anywhere サーバ SQL リファレンス』
- 「OUTPUT 文 [Interactive SQL]」『SQL Anywhere サーバ SQL リファレンス』
- 「READ 文 [Interactive SQL]」『SQL Anywhere サーバ SQL リファレンス』
- 「Ultra Light 接続パラメータ」164 ページ
- 「サポートされている終了コード」197 ページ

例

次のコマンドは、Ultra Light の *CustDB.udb* データベースに対してスクリプトファイル *mycom.sql* を実行します。ユーザ ID とパスワードが定義されていないため、デフォルトのユーザ ID **DBA** とパスワード **sql** が使用されます。-onerror オプションは終了として定義されています。このため、スクリプトファイルにエラーがあった場合、処理は終了します。

```
dbisql -ul -c DBF=CustDB.udb -onerror exit mycom.sql
```

Ultra Light SQL プリプロセッサユーティリティ (sqlpp)

Embedded SQL (ESQL) を含む C/C++ のプログラムをプリプロセッサで処理し、コンパイラを実行する前に、このプログラムに必要なコードを生成できるようにします。次の表では、万全を期すためにすべてのオプションについて説明していますが、Ultra Light に関連するオプションは **-eu** と **-wu** のみです。

構文

sqlpp -u [*options*] *esql-filename* [*output-filename*]

オプション	説明
-d	データ領域サイズを減らし、コードサイズを増加させるコードを生成します。データ構造体を再利用し、実行時に初期化してから使用します。

オプション	説明
-e *flag*	このオプションは、指定した規格に含まれない静的 Embedded SQL をエラーとして通知します。*level* 値は、使用する規格を表します。たとえば、sqlpp -e c03 ... は、コア SQL/2003 規格に含まれない構文を通知します。 *level* として使用される値は次のとおりです。 ● **c03**　コア SQL/2003 構文でない構文を通知します。 ● **p03**　上級レベル SQL/2003 構文でない構文を通知します。 ● **c99**　コア SQL/1999 構文でない構文を通知します。 ● **p99**　上級レベル SQL/1999 構文でない構文を通知します。 ● **e92**　初級レベル SQL/1992 構文でない構文を通知します。 ● **i92**　中級レベル SQL/1992 構文でない構文を通知します。 ● **f92**　上級レベル SQL/1992 構文でない構文を通知します。 ● **t**　標準ではないホスト変数型を通知します。 ● **u**　Ultra Light がサポートしていない構文を通知します。 以前のバージョンの SQL Anywhere と互換性を保つために、e、I、f を指定することもできます。これらはそれぞれ e92、i92、f92 に対応します。
-h *width*	sqlpp によって出力される *.c* ファイル内の分割された行の最大長を *width* に制限します。分割された行が C コンパイラで 1 行として解析されるように、分割された行の末尾には円記号が追加されます。デフォルト値はありません (出力の行はデフォルトで分割されません)。

オプション	説明
-k	コンパイルされるプログラムが SQLCODE のユーザ宣言をインクルードすることをプリプロセッサに通知します。
-m *mode*	カーソルの更新可能性モード。 **HISTORICAL** または **READONLY**。
-n	コード内の適切な場所で #line ディレクティブを使用して、C ファイル内に行番号情報を生成します。 このオプションを使用して、ソースのエラーをレポートし、また *output-filename* ファイルではなく、*esql-filename* ファイル内の行番号にあるソースをデバッグできます。
-o *O/S spec*	Ultra Light には不適用。
-q	ユーティリティをクワイエットモードで実行するように設定します。情報のバナー、バージョン番号、ステータスメッセージが非表示になります。エラーメッセージは引き続き表示されます。
-r-	Ultra Light には不適用。
-s *string-length*	プリプロセッサが C ファイルに出力する文字列の最大サイズを設定します。この値より長い文字列は、文字のリスト (*'a'*、*'b'*、*'c'* など) を使用して初期化されます。ほとんどの C コンパイラには、処理できる文字列リテラルのサイズに制限があります。このオプションを使用して上限を設定します。デフォルト値は 500 です。
-u	Ultra Light で必須です。Ultra Light データベースに必要な出力を生成します。

オプション	説明
-w *level*	SQL 構文に準拠しないものを警告として通知します。*level* 値は、使用する規格を表します。たとえば sqlpp -w c03 ... は、コア SQL/2003 構文に含まれない SQL 構文を通知します。 *level* として使用される値は次のとおりです。 ● **c03**　コア SQL/2003 構文でない構文を通知します。 ● **p03**　上級レベル SQL/2003 構文でない構文を通知します。 ● **c99**　コア SQL/1999 構文でない構文を通知します。 ● **p99**　上級レベル SQL/1999 構文でない構文を通知します。 ● **e92**　初級レベル SQL/1992 構文でない構文を通知します。 ● **i92**　中級レベル SQL/1992 構文でない構文を通知します。 ● **f92**　上級レベル SQL/1992 構文でない構文を通知します。 ● **t**　標準ではないホスト変数型を通知します。 ● **u**　Ultra Light がサポートしていない構文を通知します。 以前のバージョンの SQL Anywhere と互換性を保つために、e、I、f を指定することもできます。これらはそれぞれ e92、i92、f92 に対応します。
-x	マルチバイト文字列をエスケープシーケンスに変更し、コンパイラをパススルーできるようにします。
-z *collation-sequence*	照合順を指定します。

備考

このプリプロセッサは、入力ファイル内の SQL 文を C/C++ に変換します。結果は *output-filename* に書き込みます。Embedded SQL を含むソースプログラムの拡張子は通常は *sqc* です。デフォルトの *output-filename* は拡張子 *c* が付いた *esql-filename* ベースの名前です。ただし、*esql-filename* に拡張子 *.c* がすでに付いている場合、デフォルトの出力ファイル拡張子は *.cc* になります。

照合順は、プリプロセッサにプログラムのソースコードで使用されている文字を理解させるために使用します。たとえば、識別子に使用できるアルファベット文字の識別などに使用されます。Ultra Light では、照合にコードページとソート順が含まれます。**-z** を指定しなかった場合は、プリプロセッサが、オペレーティングシステムに基づいて、使用する合理的な照合順を決定しようとします。

コマンド **ulinit -Z** を実行すると、サポートされている照合と対応するコードページのリストが表示されます。

ヒント
SQL プリプロセッサ (sqlpp) には、Embedded SQL アプリケーション内の静的 SQL 文をコンパイル時に通知する機能があります。この機能は、Ultra Light アプリケーションを開発するときに特に便利です。この機能で、SQL 文に Ultra Light との互換性があることを確認できます。SQL Anywhere アプリケーションと Ultra Light アプリケーションの両方に対する SQL の互換性をテストするには、-e オプションか -w オプションまたはその両方を使用します。 SQL Flagger の詳細については、「SQL Flagger を使用した SQL 準拠のテスト」『SQL Anywhere サーバ SQL の使用法』を参照してください。

参照

- 「Embedded SQL」『SQL Anywhere サーバ プログラミング』
- 「Ultra Light 文字セット」26 ページ

例

次のコマンドは、Ultra Light アプリケーション用に Embedded SQL ファイル *srcfile.sqc* をクワイエットモードで前処理します。

```
sqlpp -u -q MyEsqlFile.sqc
```

Ultra Light エンジンユーティリティ (uleng16)

アプリケーションからの Ultra Light データベースへの同時接続を管理し、Ultra Light エンジンが -ud オプションを使用するデーモンとして稼動できるようにします。

構文

uleng16 [*-ud*] [*db-file-name*]

オプション	説明
-ud *db-file-name*	エンジンを実行して、現在のユーザセッションが終了しても実行を継続できるようにします。-ud オプションを使用してデーモンを直接起動したときは、uleng16 コマンドがデーモンプロセスを作成し、(終了して次のコマンドを実行できるように) すぐに返します。その後、デーモンがそれ自体を初期化するか、コマンドで指定されたデータベースを開こうとします。

備考

Ultra Light エンジンでは、起動時にメッセージウィンドウが表示されません。

Ultra Light エンジンは、複数のプロセスで同じデータベースに同時にアクセスする場合にアプリケーションで使用されます。Ultra Light デスクトップ管理ツールがこのエンジンを使用してデータベースに接続するため、エンジンは SQL Anywhere の *bin32* または *bin64* ディレクトリにインストールされます。

-ud オプションを使用して、Ultra Light エンジンを実行し、コンピュータをログオフするときに、データベースエンジンが動作し続けるようにします。(通常、コンピュータからログオフすると、そのセッションに関連するすべてのアプリケーションは停止します。)

参照

- 「Ultra Light の配備」 113 ページ
- 「Ultra Light C++ アプリケーションの構築と配備」『Ultra Light C/C++ プログラミング』
- 「Windows Mobile 用の Ultra Light データ管理コンポーネント」 18 ページ
- 「Ultra Light エンジン停止ユーティリティ (ulstop)」 208 ページ
- 「Ultra Light START 接続パラメータ」 184 ページ

Ultra Light エンジン停止ユーティリティ (ulstop)

Ultra Light エンジンの停止

構文

ulstop

備考

開発中にエンジンを手動でシャットダウンする場合は ulstop を使用してください。本配備で ulstop を使用する必要は一般的にはありません。

参照

- 「Windows Mobile 用の Ultra Light データ管理コンポーネント」 18 ページ
- 「Ultra Light エンジンユーティリティ (uleng16)」 207 ページ

Ultra Light ユーティリティ消去 (ulerase)

Ultra Light データベースを消去します。

構文

ulerase [*options*] [*db-file-name*]

オプション	説明
@*data*	これを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。
-k *key* または **--ek**=*key*	暗号化されたデータベースで使用する暗号化キーを指定します。
-p または **--ep**	暗号化キーの入力を求めるプロンプトを表示するよう指定します。
--log	指定したファイルに操作のログを記録します。
-q または **--quiet**	ユーティリティをクワイエットモードで実行するように設定します。情報のバナー、バージョン番号、ステータスメッセージが非表示になります。エラーメッセージは引き続き表示されます。
-u *uid*,*pwd* または **--dba**=*uid*,*pwd*	データベースへのアクセスに必要な userid と password を指定します。
-? または **--help**	ユーティリティの使用法を表示し、終了します。
db-file-name	指定したデータベースを消去します。

備考

データベースにアクセスできる必要があります。ユーザ ID とパスワードの組み合わせを使用して接続が許可される必要があります。そうでない場合、データベースは消去されません。

暗号化されたデータベースでは、接続文字列に指定されるキーが必要です。つまり、**-k** *key* または **-p** のどちらかを使用します。

参照

- 「設定ファイル」『SQL Anywhere サーバ データベース管理』

Ultra Light 情報ユーティリティ (ulinfo)

Ultra Light データベースに関する情報を表示する。

構文

ulinfo -c *options*

オプション	説明
@data	これを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。
-c "*connection-string*" または **--connect**="*connection-string*"	データベース接続パラメータを指定します。必須。
-q または **--quiet**	ユーティリティをクワイエットモードで実行するように設定します。情報のバナー、バージョン番号、ステータスメッセージが非表示になります。エラーメッセージは引き続き表示されます。
--log=*filename*	指定したファイルに操作のログを記録します。
-? または **--help**	ユーティリティの使用法を表示し、終了します。

備考

Ultra Light データベースを開くときに生成される警告メッセージは、-q オプションを使用しないかぎり、常に表示されます。

参照

- 「設定ファイル」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light 接続パラメータ」164 ページ
- 「Ultra Light global_database_id オプション」194 ページ
- 「Ultra Light ml_remote_id オプション」195 ページ

例

すでに同期されている *cv_dbattr.udb* というファイルの基本的なデータベース内部構成を表示します。

```
ulinfo -c DBF=cv_dbattr.udb
```

Ultra Light データベース初期化ユーティリティ (ulinit)

新しい Ultra Light データベースを作成します。

このユーティリティは、次のいずれかのモードで機能します。

- **空モード**　コマンドライン引数で指定された特性を持つ空のデータベースを作成します。
- **抽出モード**　SQL Anywhere データベースに基づいてデータベースを作成します。

  SQL Anywhere 参照データベースのテーブルおよびインデックスと一致する初期スキーマが作成されます。多くの参照データベース特性が抽出され、新しい Ultra Light データベースに使用されます。

構文

ulinit *options dbname*

オプション	説明
@data	これを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。 「設定ファイル」『SQL Anywhere サーバ データベース管理』を参照してください。

オプション	説明
-a "*keyword=value;...*" または **--SAconnect**="*keyword=value;...*"	ユーティリティを「抽出モード」に設定して、指定された接続パラメータを使用して、既存のデータベースに接続します。 このオプションがない場合は、ユーティリティは指定された接続パラメータ (「空モード」) を使用して、新しいデータベースを作成します。
-c または **--case**	空モード。 すべての文字列比較で大文字と小文字の区別を強制します。
-d または **--datacopy**	抽出モード。 新しい Ultra Light データベースの各テーブルに、SQL Anywhere データベースの対応するテーブルからデータをコピーします。このオプションを使用しない場合、新しいデータベースは最初は空になります。 デフォルトでは、その後の同期時にこのデータはアップロードされません。次の同期時にデータをアップロードするには、**-i** と同時に **-d** を使用します。
--date_format=*format*	空モード。 データベースから取り出した日付の表示形式を設定します。「Ultra Light date_format 作成パラメータ」 142 ページを参照してください。
--date_order=*date-format-interpretation*	空モード。 日付形式の解釈を設定します。「Ultra Light date_order 作成パラメータ」 145 ページを参照してください。

オプション	説明
-e *value* または **--fips**=*value*	空モード。 On または Off、1 または 0 など。このオプションは、Certicom 認定暗号化アルゴリズムを使用した AES FIPS 認定暗号化を制御します。「データベースセキュリティ」29 ページと「Ultra Light fips 作成パラメータ」146 ページを参照してください。
-f または **--exactschema**	抽出モード。 同じスキーマが Ultra Light でサポートされていない場合は失敗します。スキーマが異なる場合は、警告が表示されます。
-g *id* または **--databaseid**=*id*	初期データベース ID を、指定する INTEGER 値に設定します。この初期値と分割サイズが、グローバルオートインクリメントカラムがある新しいローに使用されます。アプリケーションを配備するときには、Mobile Link サーバとの同期のために、各データベースに対して異なる ID 番号の範囲を割り当てます。「Ultra Light global_database_id オプション」194 ページを参照してください。
-i または **--insertforupload**	抽出モード。 **-d** とともに使用します。挿入されたローを次の同期時にアップロードします。デフォルトでは、このユーティリティによって挿入されたローは、同期時にアップロードされません。
--identity-file=*file*	クライアントの TLS ID を含むファイルを指定します。「identity」『Mobile Link クライアント管理』を参照してください。
--identity-password=*password*	クライアント TLS ID のパスワードを指定します。 「identity_password」『Mobile Link クライアント管理』を参照してください。
-k *key* または **--key**=*key*	抽出モード。 暗号化されたデータベースで使用する暗号化キーを指定します。

オプション	説明
-K または **--prompt**	空モード。 暗号化キーの入力を求めるプロンプトを表示するよう指定します。
-l *filename* または **--sql**=*filename*	抽出モード。 実行時に、指定されたファイルに DDL データベーススキーマ作成の SQL 文のログを取ります。
--log=*filename*	空モード。 指定したファイルに操作のログを記録します。
-m *filename* または **--mirror_file**=*filename*	抽出モード。 データベースミラーファイルを指定します。「Ultra Light MIRROR_FILE 接続パラメータ」179 ページを参照してください。
--max_hash_size=*size*	空モード。 デフォルトのインデックスハッシュサイズをバイト単位で設定します。「Ultra Light max_hash_size 作成パラメータ」147 ページを参照してください。
-n *pubname* または **--publication**=*pubname*	抽出モード。 必須。テーブルを Ultra Light データベーススキーマに追加します。 *pubname* は、リファレンスデータベース内のパブリケーションを指定します。パブリケーション内のテーブルは、Ultra Light データベースに追加されます。 複数のパブリケーションからのテーブルを Ultra Light データベースに追加するには、オプションを複数回指定します。リファレンスデータベース内のすべてのテーブルを Ultra Light データベースに追加するには、**-n*** を指定します。

オプション	説明
--nearest_century=*yy*	空モード。 文字列から日付への変換で、2 桁の年の解釈を制御します。「Ultra Light nearest_century 作成パラメータ」149 ページを参照してください。
-o *value* または **--obfuscate**=*value*	空モード。 On または Off、1 または 0 など。データベースのデータを難読化するかどうか制御します。難読化は単純暗号化です。「データベースセキュリティ」29 ページと「Ultra Light obfuscate 作成パラメータ」150 ページを参照してください。
-p *size* または **--page_size**=*size*	空モード。 データベースページサイズを指定します。
--precision=*precision*	空モード。 計算結果が小数の場合の最大桁数を指定します。「Ultra Light precision 作成パラメータ」153 ページを参照してください。
-q または **--quiet**	ユーティリティをクワイエットモードで実行するように設定します。情報のバナー、バージョン番号、ステータスメッセージが非表示になります。エラーメッセージと警告は引き続き表示されます。
-r *size* または **--reserve_size**	データベースの接続のみ。 サイズを予約します。「Ultra Light RESERVE_SIZE 接続パラメータ」183 ページを参照してください。

オプション	説明
-s *pubname* または **--sync_publication**	抽出モード。 リファレンスデータベースの *pubname* と同じ定義を持つパブリケーションを Ultra Light データベースに作成します。パブリケーションは、同期の設定に使用されます。複数の同期パブリケーションを指定するには、複数の -s オプションを指定します。 このパブリケーション内のテーブルが -n オプションでリストされたパブリケーションに含まれている必要があります。 -s を指定しないと、Ultra Light リモートデータベースにパブリケーションが指定されません。 Mobile Link との同期用のパブリケーションを作成する方法については、「Ultra Light でのデータのパブリッシュ」78 ページを参照してください。
-S *checksum_level* または **--checksum_level**=*checksum_level*	空モード。 0、1、または 2。データベースページでチェックサムレベルの検証を指定します。「Ultra Light checksum_level 作成パラメータ」140 ページを参照してください。
--scale=*scale*	空モード。 計算結果が最大精度でトランケートされる場合の、小数点以下の最小桁数を指定します。「Ultra Light scale 作成パラメータ」154 ページを参照してください。
-t *file* または **--rootcert**=*file*	信頼できるルート証明書を含むファイルを指定します。この証明書は、サーバ認証に必要です。
--time_format=*format*	空モード。 データベースから取り出した時刻の表示形式を設定します。「Ultra Light time_format 作成パラメータ」155 ページを参照してください。

オプション	説明
--timestamp_format=*format*	空モード。 データベースから取り出したタイムスタンプの表示形式を設定します。「Ultra Light timestamp_format 作成パラメータ」157 ページを参照してください。
--timestamp_increment=*increment*	空モード。 Ultra Light でのタイムスタンプのトランケート方法を指定します。「Ultra Light timestamp_increment 作成パラメータ」160 ページを参照してください。
--timestamp_with_time_zone_format= *format*	空モード。 このオプションでは、データベースから取り出した TIMESTAMP WITH TIME ZONE 値の表示形式を設定します。「Ultra Light timestamp_with_time_zone_format 作成パラメータ」161 ページを参照してください。
-u <*uid*>,<*pwd*> または **--dba**=<*uid*>,<*pwd*>	データベースの接続のみ。 ユーザ ID とパスワードを指定します。
--utf8_encoding=*value*	空モード。 On または Off、1 または 0 など。Unicode 用の 8 ビットマルチバイトエンコードである UTF-8 フォーマットでデータをエンコードします。「Ultra Light 文字セット」26 ページと「Ultra Light utf8_encoding 作成パラメータ」163 ページを参照してください。
-w または **--nowarnings**	抽出モード。 警告を表示しません。

オプション	説明
-x *table* または **--exclude**	抽出モード。 リスト内に指定されたテーブルを除外します。
-y または **--overwrite**	既存のデータベースファイルを上書きします。
-z *collation-sequence* または **--collation**=*collation-sequence*	空モード。 照合順を指定します。
-Z または **--listcollation**	空モード。 使用可能な照合順をリストして終了します。
-? または **--help**	ユーティリティの使用法を表示し、終了します。

注意
いずれのモードでも説明に指定されていないオプションは使用できません。

備考

抽出モードで実行する場合、ULINIT は、SQL Anywhere データベースとできるだけ厳密に一致する Ultra Light データベースを作成しようとします。たとえば、SQL Anywhere データベースのカラムに含まれる句が Ultra Light ではサポートされていない場合、デフォルト値は無視され、代わりに Ultra Light のデフォルト値が使用されます。警告が生成されて、作成は続行されます。これは、SQL Anywhere テーブルを変更できなくても、Ultra Light の妥当な代替手段を使用できるような場合をサポートしています。スキーマの完全一致を強制するには、**-f** オプションを使用します。スキーマが Ultra Light の妥当な代替値をサポートしていない場合、ulinit ユーティリティは失敗します。

参照

- 「SQL Anywhere データベースから Ultra Light データベースへの変換」33 ページ
- 「同期モデル」『Mobile Link クイックスタート』
- 「Ultra Light 接続パラメータ」164 ページ

例

TestPublication で定義されているテーブルを含む *customer.udb* というファイルを作成します。

```
ulinit -a "DSN=MySADb;UID=JimmyB;PWD=secret" -n TestPublication -k mykey customer.udb
```

この例では、*MySADb* データソースで定義された SQL Anywhere データベースに接続します。これにより、そのデータベースからのすべてのデータベースオプションと *TestPublication* パブリケーションに含まれるすべてのテーブルを含む Ultra Light データベースが作成されます。新しい Ultra Light データベースは *customer.udb* と呼ばれ、mykey で暗号化されます。

2 つの異なるパブリケーションを含む *customer.udb* というファイルを作成します。Pub1 には、優先的に同期するデータのサブセットが含まれ、Pub2 には残りのデータが含まれます。

```
ulinit -a "DSN=MySADb;UID=JimmyB;PWD=secret" --exactschema -n Pub1 -n Pub2 -s Pub1 -s Pub2
customer.udb
```

この例では、*MySADb* データソースで定義された SQL Anywhere データベースに接続します。これにより、そのデータベースからのすべてのデータベースオプションと *Pub1* および *Pub2* パブリケーションに含まれるすべてのテーブルを含む Ultra Light データベースが作成されます。また、*Pub1* および *Pub2* パブリケーションで新しい Ultra Light データベースも作成されます。--exactschema オプションが設定されているので、ulinit は正確なスキーマをすべて抽出できないと失敗します。

別の *customer.udb* ファイルがあれば、それを上書きする新しい空のデータベースを作成します。新しいデータベースにはスキーマがなく、データベース オプションはすべてデフォルト値に設定されています。

```
ulinit -y customer.udb
```

Ultra Light データベースへの XML のロードユーティリティ (ulload)

XML ファイルから新しいデータベースまたは既存のデータベースにデータをロードします。

構文

ulload -c "*connection-string*" [*options*] *xml-file*

オプション	説明
@data	これを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。 「設定ファイル」『SQL Anywhere サーバ データベース管理』を参照してください。
-a または **--append**	データとスキーマの定義を既存のデータベースに追加します。
-c "*connection-string*" または **--connect**="*connection-string*"	データベース接続パラメータを指定します。
-d または **--dataonly**	データだけをロードし、XML ファイル入力内のメタデータは無視します。 -d または --dataonly スイッチは -a が指定されている場合にだけ使用できます (データのみをロードするため、データをロードする UDB がロードされるデータをサポートするスキーマと共に存在する必要があります)。
-e *value* または **--fips**=*value*	On または Off、1 または 0 などを指定します。このオプションは、Certicom 認定暗号化アルゴリズムを使用して、AES FIPS 認定暗号化を制御します。 「データベースセキュリティ」29 ページと「Ultra Light fips 作成パラメータ」146 ページを参照してください。

オプション	説明
-E *behavior* または **--onerror**=*behavior*	XML ファイルからデータを読み出し中にエラーが起こった場合の事象を制御します。*behavior* には次のいずれかの値を指定できます。 ● **continue**　エラーは無視され、XML のロードが継続されます。 ● **prompt**　続行するように要求されます。 ● **quit**　XML のロードが停止し、エラーで終了します。この動作はデフォルトの動作です。 ● **exit**　ulload が終了します。
-f *directory* または **--filedir**=*directory*	ロードする追加データを含むファイルが格納されているフォルダを設定します。 「Ultra Light データベースのアンロードユーティリティ (ulunload)」232 ページを参照してください。
-g *ID* または **--databaseid**=*ID*	初期データベース ID を、指定する INTEGER 値に設定します。この初期値と分割サイズが、グローバルオートインクリメントカラムがある新しいローに使用されます。アプリケーションを配備するときには、Mobile Link サーバとの同期のために、各データベースに対して異なる ID 番号の範囲を割り当てます。「Ultra Light global_database_id オプション」194 ページを参照してください。
-i または **--insertforsync**	挿入されたローを次の同期時にアップロードします。デフォルトでは、このユーティリティによって挿入されたローは、同期時にアップロードされません。
--identity-file = *file*	クライアントの TLS ID を含むファイルを指定します。 「identity」『Mobile Link クライアント管理』を参照してください。
--identity-password = *password*	クライアント TLS ID のパスワードを指定します。 「identity_password」『Mobile Link クライアント管理』を参照してください。

オプション	説明
-l *filename* または **--log**=*filename*	指定したファイルに操作のログを記録します。
-n または **--schemaonly**	メタデータだけをロードし、XML 入力ファイル内のデータは無視します。
-o *value* または **--obfuscate**=*value*	On または Off、1 または 0 など。データベースのデータを難読化するかどうか制御します。難読化は、単純な暗号化の一種です。. 「データベースセキュリティ」29 ページと「Ultra Light obfuscate 作成パラメータ」150 ページを参照してください。
-p *page-size* または **--page_size**=*page-size*	データベースページサイズを定義します。「Ultra Light page_size 作成パラメータ」151 ページを参照してください。
-q または **--quiet**	ユーティリティをクワイエットモードで実行するように設定します。情報のバナー、バージョン番号、ステータスメッセージが非表示になります。エラーメッセージは引き続き表示されます。
-s *file* または **--sql**=*file*	データベースのロードに使用された SQL 文を、指定する *file* に記録します。
-t *file* または **--rootcert**=*file*	信頼できるルート証明書を含むファイルを指定します。この証明書は、サーバ認証に必要です。

オプション	説明
--utf8_encoding=*value*	On または Off、1 または 0 など。Unicode 用の 8 ビットマルチバイトエンコードである UTF-8 フォーマットでデータをエンコードします。 「Ultra Light 文字セット」26 ページと「Ultra Light utf8_encoding 作成パラメータ」163 ページを参照してください。
-v または **--verbose**	冗長メッセージを表示します。
-y または **--overwrite**	確認メッセージを表示しないでデータベースファイルを上書きします。このオプションは、ulload を使用して新しいデータベースを作成する場合にのみ適用されます。
-? または **--help**	ユーティリティの使用法を表示し、終了します。

備考

ulload ユーティリティは、ulunload、ulunloadold (SQL Anywhere 10 で提供)、または ulxml (Ultra Light バージョン 8 と 9) で生成された入力 XML ファイルを取得します。ulunload とこのユーティリティを同時に使用すると、データベースを再構築できます。データベースを再構築するもう 1 つの手段としては、ulunload を使用して SQL 文を生成し、DBISQL を使用して新しいデータベースに読み込む方法があります。

XML ファイルには、スキーマのメタデータかデータベースデータのメタデータまたはその両方が含まれている可能性があります。**-d** を使用すると、スキーマのメタデータは無視され、データのみが *.udb* ファイルに追加されます。**-n** を使用すると、データとメタデータが無視され、スキーマのみが *.udb* ファイルに追加されます。

コマンドラインでオプションを設定するか、証明書を指定すると、ulload で処理される *xml-file* 内の設定より優先されます。

ulload ユーティリティは、XML の読み込み時にデータベースに同期プロファイルをリストアします。

このユーティリティはエラーコードを返します。0 以外の値は処理に失敗したことを示します。

参照

- 「Ultra Light 接続パラメータ」164 ページ
- 「Ultra Light データベースのアンロードユーティリティ (ulunload)」232 ページ
- 「サポートされている終了コード」197 ページ
- 「Ultra Light global_database_id オプション」194 ページ

例

新しい Ultra Light データベースファイル *sample.udb* を作成し、*sample.xml* 内のデータをロードします。

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

sample.xml から既存のデータベース *sample.udb* にデータをロードし、エラーが発生した場合はプロンプトを表示します。

```
ulload -d -c DBF=sample.udb --onerror=prompt sample.xml
```

sample.udb データベースの *test_data.xml* に保存されるスキーマとデータを作成します。-a スイッチが指定されているため、このコマンドを実行する前に *sample.udb* が存在する必要があります。さらに、*sample.udb* にすでにあるスキーマやデータと競合するスキーマやデータがあると、ULLOAD コマンドが失敗します。

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

Ultra Light 同期ユーティリティ (ulsync)

Ultra Light データベースを Mobile Link サーバと同期させます。このツールを使用して、アプリケーション開発中に同期をテストできます。

構文

ulsync -c [*options*] [*synchronization parameters*]

オプション	説明
@data	これを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。「設定ファイル」『SQL Anywhere サーバ データベース管理』を参照してください。
-c "*connection-string*" または **--connect**="*connection-string*"	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。

オプション	説明
-p *profile-name* または **--profile=profile**	指定した同期プロファイルを使用して同期します。次の構文と同義です。 SYNCHRONIZE profileName MERGE syncOptions ここでは、同期オプションは後続の ulsync オプションから取得されます。次に例を示します。 ulsync -p profileName "MobiLinkUid=ml;ScriptVersion=Version001...syncOptions" 「同期プロファイルオプション」227 ページを参照してください。
-q または **--quiet**	ユーティリティをクワイエットモードで実行するように設定します。情報のバナー、バージョン番号、ステータスメッセージが非表示になります。エラーメッセージは引き続き表示されます。
-r または **--result**	前回の同期の結果を表示し、終了します。
-v または **--verbose**	同期の進行状況のメッセージを表示します。C++ API または SQL SYNCHRONIZE PROFILE 文を使用して、同期の進捗状況を表示するかどうかも指定します。 「CREATE SYNCHRONIZATION PROFILE 文 [Ultra Light]」431 ページを参照してください。
--log *filename*	指定したファイルに操作のログを記録します。
-? または **--help**	ユーティリティの使用法を表示し、終了します。

備考

trusted_certificate または **e2ee_public_key** オプションで証明書ファイルが作成されると、Ultra Light ランタイムはすべての iPhone アプリケーション配備パッケージの一部であるメインリ

ソースでのみこれらのファイルを検索します。Xcode プロジェクトの */Resources* フォルダに項目を含めることによって、項目をバンドルに追加します。これは、Ultra Light データベースに保存される証明書には適用できず、Mac OS クライアント (iPhone のみ) には影響を与えません。「trusted_certificates」『Mobile Link クライアント管理』を参照してください。

バージョン 10 以前に有効であった次のオプションは今はサポートされていません。-a *authenticate-parameters*、-e *sync-parms*、-k *stream-type*、-n (no sync)、および -x *protocol options*。-e *keyword=value* は同期パラメータ文字列の一部となり、-k および -x は **Stream=** *stream*{*stream-parms*} 同期パラメータ文字列の一部になっています。

用途によって、ulsync は次の SQL 文のいずれかと同義と考えることができます。

```
ulsync -p profile "parms"
```

この文は次の文と同義です。

```
SYNCHRONIZE PROFILE profile MERGE
parms
```

および

```
ulsync "parms"
```

この文は次の文と同義です。

```
SYNCHRONIZE USING <parms>
```

安全な同期のために、Ultra Light アプリケーションはパブリック証明書にアクセスする必要があります。次の方法で証明書を参照できます。

- ulinit または ulload で **-t** *file* オプションを使用して Ultra Light データベースの作成時に証明書情報を組み込む。
- **trusted_certificate=***file* ストリームオプションを使用して同期時に外部の証明書ファイルを参照する。

このユーティリティはエラーコードを返します。0 以外の値は処理に失敗したことを示します。

参照

- 「同期プロファイルオプション」227 ページ
- 「エンドツーエンド暗号化」『SQL Anywhere サーバ データベース管理』
- 「trusted_certificates」『Mobile Link クライアント管理』
- 「Ultra Light 接続パラメータ」164 ページ
- 「Ultra Light クライアント」69 ページ
- 「サポートされている終了コード」197 ページ
- 「Mobile Link ファイル転送ユーティリティ (mlfiletransfer)」『Mobile Link クライアント管理』

例

次のコマンドは、**remoteA** という Mobile Link ユーザの *myuldb.udb* というデータベースファイルを同期させます。

```
ulsync -c DBF=myuldb.udb "MobiLinkUid=remoteA;Stream=http;ScriptVersion=2"
```

次のコマンドは、*myuldb.udb* というデータベースファイルを HTTPS を通じ *C:¥Users¥Public¥Documents¥SQL Anywhere 16¥Samples¥Certificates¥rsaroot.crt* 証明書を使用して同期させます。データベースの作成時に信頼できる証明書ファイルがデータベースに追加されなかったため、**trusted_certificate=***file* オプションを使用する必要があります。さらに、Mobile Link ユーザ名は **remoteB** です。

```
ulsync -c DBF=myuldb.udb "Stream=https{trusted_certificate=C:¥Users¥Public¥Documents¥SQL
Anywhere 16¥Samples¥Certificates¥rsaroot.crt};
MobiLinkUid=remoteB;ScriptVersion=2;UploadOnly=ON"
```

次のコマンドは、*synced.udb* というデータベースファイルについて、前回の同期の結果を表示します。

```
ulsync -r -c dbf=synced.udb
```

前回の同期の結果は、次のようにリストされます。

```
SQL Anywhere UltraLite Database Synchronize Utility Version XX.X
Results of last synchronization:
Succeeded
Download timestamp: 2006-07-25 16:39:36.708000
Upload OK
No ignored rows
Partial download retained
Authentication value:1000 (0x3e8)
```

次の例は、CustDB データベースをユーザ名 50 で TCP/IP 接続によってポート 2439 で同期させる場合に使用するコマンドラインを示します。

```
ulsync -c "dbf=C:¥Users¥Public¥Documents¥SQL Anywhere 16¥Samples¥UltraLite¥custdb.udb"
"MobiLinkUid=50;ScriptVersion=custdb 12.0;Stream=tcpip{port=2439}"
```

次のコマンドは、E2EE とともに TLS 暗号化を使用する方法を示します。

```
ulsync -c "uid=dba;pwd=sql;dbf=myudb.db"
"MobiLinkUid=rem1;MobiLinkPwd=password;ScriptVersion=v1;Stream=tls{host=myServer;port=2439;tru
sted_certificate=c:¥clientcert.pem;e2ee_public_key=c:¥e2eepublic.pem}"
```

同期プロファイルオプション

同期プロファイルオプションは、ulsync ユーティリティのコマンドラインで、使用する他のすべてのコマンドラインオプションを定義した後に指定します。キーワードは大文字と小文字が区別されません。

同期プロファイルオプション	有効な値	説明
AllowDownloadDupRows	ブール式	このオプションを指定すると、プライマリキーの同じローが複数ダウンロードされたときにエラーが発生しません。このオプションによって、同期を失敗させることなく、不整合なデータを同期できます。デフォルト値は "no" です。「Additional Parameters 同期パラメータ」90 ページを参照してください。
AuthParms	文字列 (カンマ区切り)	Mobile Link サーバに送信される認証パラメータのリストを指定します。認証パラメータを使用すると、Mobile Link スクリプトでカスタム認証を実行できます。「Authentication Parameters 同期パラメータ」92 ページを参照してください。
CheckpointStore	ブール式	同期中にデータベースのチェックポイントを追加して、同期プロセス中のデータベースの拡張を制限します。「Additional Parameters 同期パラメータ」90 ページを参照してください。
ContinueDownload	ブール式	以前に失敗したダウンロードを再起動します。ダウンロードを続行すると、失敗した同期でダウンロードするように選択されていた変更のみを受信します。デフォルトでは、Ultra Light はダウンロードを続行しません。「ダウンロードの失敗の再開」『Mobile Link サーバ管理』を参照してください。

同期プロファイルオプション	有効な値	説明
DisableConcurrency	ブール式	この同期中は他のスレッドからデータベースへのアクセスを禁止します。 「Additional Parameters 同期パラメータ」90 ページを参照してください。
DownloadOnly	ブール式	ダウンロード専用同期を実行します。 「Download Only 同期パラメータ」94 ページを参照してください。
KeepPartialDownload	ブール式	接続エラーが発生した場合に、部分的なダウンロードを維持するかどうかを制御します。デフォルトでは、Ultra Light は、部分的にダウンロードされた変更をロールバックしません。「Keep Partial Download 同期パラメータ」96 ページを参照してください。
MobiLinkPwd	文字列	ユーザ名に対する既存の Mobile Link パスワードを指定します。 「MobiLinkPwd (mp) 拡張オプション」『Mobile Link クライアント管理』を参照してください。
MobiLinkUid	文字列	Mobile Link ユーザ名を指定します。 「-u dbmlsync オプション (旧式)」『Mobile Link クライアント管理』と「-mn dbmlsync オプション」『Mobile Link クライアント管理』を参照してください。

同期プロファイルオプション	有効な値	説明
NewMobiLinkPwd	文字列	Mobile Link ユーザの新しいパスワードを指定します。このオプションは、既存のパスワードを変更する場合に使用します。「-mn dbmlsync オプション」『Mobile Link クライアント管理』を参照してください。
Ping	ブール式	サーバとの通信の確認のみを行います。同期は実行しません。「Ping 同期パラメータ」100 ページを参照してください。
Publications	文字列 (カンマ区切り)	同期させるパブリケーションを指定します。パブリケーションによって、同期に関与するリモートデータベース上のテーブルが決定されます。このパラメータが空白 (デフォルト) の場合は、すべてのテーブルが同期されます。このパラメータがアスタリスク (*) の場合は、すべてのパブリケーションが同期されます。「Ultra Light でのデータのパブリッシュ」78 ページを参照してください。
ScriptVersion	文字列	Mobile Link スクリプトのバージョンを指定します。スクリプトバージョンは、同期中に統合データベースの Mobile Link によって実行されるスクリプトを決定します。スクリプトバージョンを指定しない場合、「デフォルト」が使用されます。「ScriptVersion (sv) 拡張オプション」『Mobile Link クライアント管理』を参照してください。

同期プロファイルオプション	有効な値	説明
SendDownloadACK	ブール式	クライアントからサーバにダウンロード確認が送信されるように指定します。デフォルトでは、Mobile Link サーバはダウンロード確認を提供しません。 「Send Download Acknowledgement 同期パラメータ」103 ページを参照してください。
Stream	文字列 (サブリスト付き)	Mobile Link のネットワーク同期プロトコルを指定します。 「Stream Type 同期パラメータ」105 ページを参照してください。
TableOrder	文字列 (カンマ区切り)	アップロードでのテーブルの順序を指定します。デフォルトでは、Ultra Light は、外部キーの関係に基づいて順序を選択します。. 「Additional Parameters 同期パラメータ」90 ページを参照してください。
UploadOnly	文字列	同期にはアップロードのみが含まれ、ダウンロードは発生されないよう指定します。 「Upload Only 同期パラメータ」108 ページを参照してください。

ブール値に指定できるのは、Yes/No、1/0、True/False、On/Off です。ブール値の場合はどれであっても、デフォルトは No です。他の値の場合、デフォルトは未指定になります。

参照

- 「ALTER SYNCHRONIZATION PROFILE 文 [Ultra Light]」420 ページ
- 「DROP SYNCHRONIZATION PROFILE 文 [Ultra Light]」441 ページ
- 「SYNCHRONIZE 文 [Ultra Light]」455 ページ
- 「Ultra Light 作成パラメータ」137 ページ

Ultra Light データベースのアンロードユーティリティ (ulunload)

使用するオプションによって、次のいずれかをアンロードします。

- Ultra Light データベース全体を XML または SQL にアンロードする。
- Ultra Light データのすべてまたは一部だけを XML または SQL にアンロードする。

構文

ulunload -c "*connection-string*" [*options*] *output-file*

オプション	説明
@data	これを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。
-b *max-size* または **--maxblob**=*max-size*	XML ファイルに格納するカラムデータの最大サイズを設定します。デフォルトは 10 KB です。すべてのデータを XML ファイルに保存するには (最大サイズなし)、**-b -1** を使用します。
-c "*connection-string*" または **--connect**="*connection-string*"	必須。*connection-string* の DBF パラメータまたは file_name パラメータで指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-d または **--dataonly**	データベースのデータのみを出力ファイルにアンロードします。スキーマ情報はアンロードしません。
-e *table,...* または **--exclude**=*table,...*	データベースのアンロード時に指定した *table* を除外します。複数のテーブルをカンマで区切って指定できます。次に例を示します。 -e mydbtable1,mydbtable5
-f *directory* または **--filedir**=*directory*	-b で指定した最大サイズよりも大きいデータを格納するフォルダを設定します。デフォルトは、出力ファイルと同じフォルダです。

オプション	説明
-l *filename* または **--log**=*filename*	指定したファイルに操作のログを記録します。
-n または **--schemaonly**	スキーマだけをアンロードし、データベース内のデータは無視します。
-q または **--quiet**	ユーティリティをクワイエットモードで実行するように設定します。情報のバナー、バージョン番号、ステータスメッセージが非表示になります。エラーメッセージは引き続き表示されます。
-s または **--sql**	SQL Anywhere 互換の SQL 文としてアンロードします。SQL ファイルの出力は、DBISQL. を使用して Ultra Light または SQL Anywhere で読み込むことができます。
-t *table*,... または **--include**=*table*,...	指定した *table* 内のデータだけをアンロードします。複数のテーブルをカンマで区切って指定できます。次に例を示します。 -t mydbtable2,mydbtable6
-v または **--verbose**	冗長メッセージを表示します。
-x *owner* または **--owner**=*owner*	テーブルが特定のユーザ ID の所有になるように出力します。このオプションは、-s オプションと同時に使用できます。
-y または **--overwrite**	確認メッセージを表示しないで *output-file* を上書きします。

オプション	説明
-? または **--help**	ユーティリティの使用法を表示し、終了します。
output-file	必須。データベースのアンロード先のファイルの名前を設定します。-s オプションを使用すると、データベースは SQL 文としてアンロードされます。-s オプションを使用しなかった場合は、データベースは XML としてアンロードされます。

備考

デフォルトでは、データベース内のスキーマとデータを表す XML が出力されます。出力は、アーカイブに使用するか、Ultra Light データベースをすべてのリリース間で移植するために使用できます。

同期プロファイルがあるデータベースを保存すると、以前のバージョンの Ultra Light ユーティリティと互換性がない XML になります。対処方法として、XML を編集し、次のようにマーク付けされたテキストセクションを削除します。

```
<syncprofiles>...</syncprofiles>
```

データベースのアンロード時に次のデータは保持されません。

- 同期ステータス、同期数、ロー削除。アンロードの前にデータベースを同期させる必要があります。
- Ultra Light ユーザのエントリ。

保持されたデータベースオプションまたはプロパティを確認するには、ulload ユーティリティでデータベースを再ロードした後に ulinfo を実行します。

カラムデータが、-b で指定した最大サイズを超えていた場合、オーバーフローは **.bin* ファイルに保存されます。このファイルは、次のいずれかのフォルダにあります。

- XML ファイルと同じフォルダ
- -f で指定したフォルダ

ファイルは次の命名規則に従います。

```
tablename-columnname-rownumber.bin
```

-x オプションで Ultra Light テーブルに所有権を割り当てることができます。テーブルに所有者を割り当てるのは、出力の SQL 文を、SQL Anywhere データベースの作成または変更に使用する場合だけです。Ultra Light で読み込まれるとき、所有者名は通知されることなく無視されます。

このユーティリティはエラーコードを返します。0 以外の値は処理に失敗したことを示します。

このユーティリティを使用して Windows Mobile デバイス上で直接データベースをアンロードすると、Ultra Light はアンロードまたは操作の実行前にデータベースをバックアップできません。したがって、これらの操作を手動で行ってから、これらのウィザードを実行してください。

参照

- 「設定ファイル」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light 接続パラメータ」164 ページ
- 「サポートされている終了コード」197 ページ
- 「Ultra Light データベースへの XML のロードユーティリティ (ulload)」219 ページ
- 「Ultra Light 情報ユーティリティ (ulinfo)」210 ページ

例

sample.udb データベースを *sample.xml* ファイルにアンロードします。

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

sample.udb データベースのデータを *sample1.sql* という SQL ファイルにアンロードします。SQL ファイルが存在する場合は、上書きします。

```
ulunload -c DBF=sample.udb -d -y -s sample.sql
```

Ultra Light データベース検証ユーティリティ (ulvalid)

Ultra Light データベースのフル (通常の) 検証を実行します。

構文

ulvalid -c "*connection-string*" [*options*]

オプション	説明
@*data*	これを使用すると、指定された環境変数または設定ファイルからオプションを読み込むことができます。同じ名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。
-c "*connection-string*" または **--connect**="*connection-string*"	必須。*connection-string* で指定するデータベースに接続します。ユーザ ID とパスワードの両方を指定しなかった場合は、デフォルトの UID **DBA** と PWD **sql** が使用されます。
-e または **--express**	エクスプレス検証。テーブルの検証のみを実行します。このオプションを使用すると、通常の検証よりも速く検証できます。

オプション	説明
-q または **--quiet**	ユーティリティをクワイエットモードで実行するように設定します。情報のバナー、バージョン番号、ステータスメッセージが非表示になります。エラーメッセージは引き続き表示されます。
-v または **--verbose**	冗長メッセージを表示します。
--log=*filename*	指定したファイルに操作のログを記録します。
-? または **--help**	ユーティリティの使用法を表示し、終了します。

備考

データベースの検証によってテーブルメタデータの精度を検証し、ファイルが破損していないことを確認します。

検証には次のものがあります。

- **データベースページ**　チェックサム (有効な場合) を使用してすべてのデータベース ページを検証します。特定の重要なページには必ずチェックサムが含まれており、チェックサムのないページにも基本的な妥当性検査が実行されます。
- **テーブル**　テーブルのロー数が各インデックスの数と一致していることをチェックしてテーブルを検証します。
- **インデックス**　エントリが有効なローを参照することをチェックしてインデックスを検証します。ulvalid -e では、テーブルの検証のみを行うエクスプレスチェックを実行します。

参照

- 「Ultra Light checksum_level 作成パラメータ」 140 ページ
- 「設定ファイル」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light データベースの検証」 45 ページ

例

データベース *sample.udb* のエクスプレス検証の例は、クワイエットモードで動作します。

```
ulvalid -c DBF=sample.udb -e -q
```

Ultra Light のシステムテーブル

Ultra Light データベースのスキーマは独自フォーマットで格納されます。以前のバージョンの Ultra Light データベースは、複数のシステムテーブルに格納されていました。これらのシステムテーブルは下位互換性のために引き続き問い合わせ可能になっていますが (これらは実質的にはシステムビューです)、その中にはユーザスキーマ (テーブル、カラム、インデックスなど) に関する情報のみが含まれており、システムスキーマに関する情報は含まれていません。たとえば、systable に対する問い合わせによって systable 自体のプロパティを検索することはできません。systable に対する問い合わせでは、ユーザが作成したテーブルのプロパティのみを検索できます。

各 Ultra Light プログラミング API では、データベースのスキーマに関する問い合わせに使用できるオブジェクトとメソッドがサポートされています。スキーマについて調べる場合は、システムビューを問い合わせるのではなく、これらのオブジェクトと API を使用することをおすすめします。

これらのシステムビューに対して実行するすべてのクエリは、完全なテーブルスキャンに相当します。これらのシステムビューでは、インデックススキャンはサポートされていません。

sysarticle システムテーブル

sysarticle システムテーブルの各ローは、パブリケーションに属するテーブルを示します。

カラム名	カラム型	説明
publication _id	UNSIGNED INT	このアーティクルが属するパブリケーションの識別子。
table_id	UNSIGNED INT	パブリケーションに属するテーブルの識別子。
where_expr	VARCHAR(256)	ローをフィルタする述部 (オプション)。

制約

PRIMARY KEY (publication_id, table_id)

FOREIGN KEY (publication_id) REFERENCES syspublication (publication_id)

FOREIGN KEY (table_id) REFERENCES systable (object_id)

参照

- 「syspublication システムテーブル」240 ページ

syscolumn システムテーブル

syscolumn システムテーブルの各ローは、カラム 1 つを示します。

カラム名	カラム型	説明
column_name	VARCHAR(128)	カラムの名前。
default	VARCHAR(128)	このカラムのデフォルト値。例：autoincrement。
domain	UNSIGNED INT	カラムのドメインを示す列挙値。
domain_info	UNSIGNED INT	サイズが可変のドメインで使用されます。
nulls	VARCHAR(1)	カラムのデフォルトに NULL が許可されるかどうかを決定します。
object_id	UNSIGNED INT	カラムのユニークな識別子。
table_id	UNSIGNED INT	カラムが属するテーブルの識別子。

制約

PRIMARY KEY (table_id, object_id)

FOREIGN KEY (table_id) REFERENCES systable (object_id)

sysindex システムテーブル

sysindex システムテーブルの各ローは、データベース内のインデックス 1 つを示します。

カラム名	カラム型	説明
check_on_commit	BIT	すべての外部キーに一致するプライマリローがあることを確認するために参照整合性をいつチェックするかを示します。type が **foreign** の場合にのみ必要です。
index_name	VARCHAR(128)	インデックスの名前。
ixcol_count	UNSIGNED INT	インデックス内のカラム数。
nullable	BIT	type が **foreign** の場合にのみ必要です。NULL が許可されるかどうかを示します。

カラム名	カラム型	説明
object_id	UNSIGNED INT	インデックスのユニークな識別子。
primary_index_id	UNSIGNED INT	type が **foreign** の場合にのみ必要です。プライマリインデックスの識別子をリストします。
primary_table_id	UNSIGNED INT	type が **foreign** の場合にのみ必要です。プライマリテーブルの識別子をリストします。
root_handle	UNSIGNED INT	内部でのみ使用。
table_id	UNSIGNED INT	インデックスが適用されるテーブルのユニークな識別子。
type	VARCHAR(10)	インデックスの型。次のいずれかです。 ● **primary** ● **foreign** ● **key** ● **unique** ● **index**
hash_size	UNSIGNED SHORTINT	インデックスのハッシュに使用されるハッシュサイズ。

制約

PRIMARY KEY (table_id, object_id)

FOREIGN KEY (table_id) REFERENCES systable (object_id)

参照

● 「sysixcol システムテーブル」 239 ページ

sysixcol システムテーブル

sysixcol システムテーブルの各ローは、sysindex にリストされているインデックスのカラム 1 つを示します。

カラム名	カラム型	説明
column_id	UNSIGNED INT	インデックス対象カラムのユニークな識別子。
index_id	UNSIGNED INT	このインデックスカラムが属するインデックスのユニークな識別子。
order	VARCHAR(1)	インデックス内のカラムが昇順 (A) か、降順 (D) かを示します。
sequence	UNSIGNED INT	インデックス内のカラムの順序。
table_id	UNSIGNED INT	インデックスが適用されるテーブルのユニークな識別子。

制約

PRIMARY KEY (table_id, index_id, sequence)

FOREIGN KEY (table_id, index_id) REFERENCES sysindex (table_id, object_id)

FOREIGN KEY (table_id, column_id) REFERENCES syscolumn (table_id, object_id)

参照

- 「sysindex システムテーブル」238 ページ

syspublication システムテーブル

syspublication システムテーブルの各ローは、パブリケーションを示します。

カラム名	カラム型	説明
download_timestamp	TIMESTAMP	最後のダウンロードの時刻。
last_sync	UNSIGNED BIGINT	アップロードの進行状況を追跡するために使用します。
publication_id	UNSIGNED INT	パブリケーションのユニークな識別子。
publication_name	VARCHAR(128)	パブリケーションの名前。

制約

PRIMARY KEY (publication_id)

参照

- 「sysarticle システムテーブル」237 ページ

syssyncresult システムテーブル

syssyncresult システムテーブルローにはいずれも最新の同期についての情報が含まれています。

カラム名	カラム型	説明
sql_code	INTEGER	最後の同期の SQL コード。
error_string	CHAR(200)	前回の同期でのエラーメッセージ。
stream_error_code	SMALLINT	特定のストリームエラー。想定される値については、ss_error_code 列挙を参照してください。
system_error_code	INTEGER	システム固有のエラーコード。エラーコードの詳細については、プラットフォームのマニュアルを参照してください。
stream_error_string	CHAR(80)	stream_error_code 値のための文字列です。利用可能な場合は、追加情報が含まれます。
upload_ok	BIT	アップロードが成功した場合は true、それ以外の場合は false。
ignored_rows	BIT	アップロードされたローが無視された場合は true、それ以外の場合は false。
auth_status	UNSIGNED SMALLINT	同期認証ステータス。
auth_value	INTEGER	Mobile Link サーバが auth_status の結果を判断するために使用する値。
auth_info	CHAR(1024)	Mobile Link ユーザ認証スクリプトから返された認証メッセージ。
partial_download_retained	BIT	部分的なダウンロードが保持されたことを通知する値。

カラム名	カラム型	説明
timestamp	TIMESTAMP	最後の同期の日付と時刻。
sent_bytes	UNSIGNED INT	アップロードでこれまでに送信されたバイト数。
sent_inserts	UNSIGNED INT	アップロードでこれまでに挿入されたローの数。
sent_updates	UNSIGNED INT	アップロードでこれまでに送信された更新済みローの数。
sent_deletes	UNSIGNED INT	アップロードでこれまでに送信された削除済みローの数。
received_bytes	UNSIGNED INT	ダウンロードでこれまでに送信されたバイト数。
received_inserts	UNSIGNED INT	ダウンロードでこれまでに挿入されたローの数。
received_updates	UNSIGNED INT	ダウンロードでこれまでに送信された更新済みローの数。
received_ignored_updates	UNSIGNED INT	ダウンロードで受信された重複ローの数。
received_deletes	UNSIGNED INT	ダウンロードでこれまでに送信された削除済みローの数。
received_ignored_deletes	UNSIGNED INT	すでに削除されているローのダウンロードで受信された削除済みローの数。
received_truncate_deletes	UNSIGNED INT	ダウンロード時にトランケート操作によって削除されたローの数。

参照

- 「Keep Partial Download 同期パラメータ」96 ページ

systable システムテーブル

systable システムテーブルの各ローは、データベース内のテーブル 1 つを示します。

カラム名	カラム型	説明
column_count	UNSIGNED INT	テーブル内のカラム数。
index_count	UNSIGNED INT	テーブル内のインデックス数。
ixcol_count	UNSIGNED INT	テーブル内のすべてのインデックスのカラムの合計数。
table_name	VARCHAR(128)	テーブルの名前。
object_id	UNSIGNED INT	テーブルのユニークな識別子。
sync_type	VARCHAR(32)	Mobile Link による同期に使用されます。**no_sync** (非同期)、**all_sync** (すべてのローの同期)、**normal_sync** (変更されたローのみ同期) のいずれか。
table_type	VARCHAR(32)	**user** (ユーザが作成したテーブルを示す)。

制約

PRIMARY KEY (object_id)

Ultra Light Java Edition データベースのプロパティ

Ultra Light Java Edition データベースのプロパティの値は、データベースの最初の作成時に定義されます。これらの値は、Ultra Light Java Edition のデータベースを再作成するか、または対応するデータベースオプションを使用できる場合にはオプションを編集することによって、変更できます。

Ultra Light Java Edition では、次のデータベースのプロパティがサポートされます。

プロパティ	説明
blob_file_base_dir	データベースで BLOB ファイルの検索に使用されるベースディレクトリを返します。 参照： ● 「Ultra Light Java Edition の blob_file_base_dir オプション」246 ページ ● 「Ultra Light Java Edition データベースロードユーティリティ (uljload)」252 ページ
date_format	データベースで文字列変換に使用される日付形式を返します。 「Ultra Light date_format 作成パラメータ」142 ページを参照してください。
date_order	データベースで文字列変換に使用される日付順を返します。 「Ultra Light date_order 作成パラメータ」145 ページを参照してください。
global_database_id	グローバルオートインクリメントカラムに使用される global_database_id オプションの値を返します。 「Ultra Light global_database_id オプション」194 ページを参照してください。
ml_remote_id	Mobile Link 同期のデータベースをユニークに識別する ml_remote_id オプションの値を返します。 「Ultra Light ml_remote_id オプション」195 ページを参照してください。
database_name	現在の接続におけるデータベースの名前 (またはエイリアス) を返します。返される名前は、DBN 接続パラメータ値と一致します。DBN 接続パラメータを使用しなかった場合、返される名前はパスと拡張子のないデータベースファイルになります。 参照： ● 「Ultra Light DBN 接続パラメータ」176 ページ ● 「Ultra Light DBF 接続パラメータ」173 ページ
nearest_century	データベースで文字列変換に使用される基準年を返します。 「Ultra Light nearest_century 作成パラメータ」149 ページを参照してください。
page_size	データベースのページサイズ (バイト単位) を返します。 「Ultra Light page_size 作成パラメータ」151 ページを参照してください。

プロパティ	説明
precision	データベースで文字列変換に使用される浮動小数点の精度を返します。 「Ultra Light precision 作成パラメータ」153 ページを参照してください。
scale	データベースによる文字列変換中に、計算結果が最大精度にトランケートされるときの小数点以下の最大桁数を返します。 「Ultra Light scale 作成パラメータ」154 ページを参照してください。
time_format	データベースで文字列変換に使用される時間形式を返します。 「Ultra Light time_format 作成パラメータ」155 ページを参照してください。
timestamp_format	データベースで文字列変換に使用されるタイムスタンプ形式を返します。「Ultra Light timestamp_format 作成パラメータ」157 ページを参照してください。
timestamp_increment	2 つのユニークなタイムスタンプ間のマイクロ秒単位の最小差を返します。 「Ultra Light timestamp_increment 作成パラメータ」160 ページを参照してください。
timestamp_with_time_zone	TIMESTAMP WITH TIME ZONE 値のタイムスタンプ形式を返します。 「Ultra Light timestamp_with_time_zone_format 作成パラメータ」161 ページを参照してください。

参照

- 「データベースプロパティの読み取り」39 ページ
- 「データベースオプションへのアクセス」39 ページ
- 「Ultra Light 作成パラメータ」137 ページ
- 「Ultra Light Java Edition データベースのオプション」245 ページ

Ultra Light Java Edition データベースのオプション

Ultra Light Java Edition データベースのオプションの値は、データベースの最初の作成時に定義されます。これらの値は、データベースへの接続中に変更できます。

この項では、使用可能な Ultra Light データベースのオプションについて説明します。これらのオプションに加え、Ultra Light Java Edition データベースでは、次の Ultra Light データベースオプションがサポートされます。

- global_database_id
- ml_remote_id

参照

- 「Ultra Light global_database_id オプション」194 ページ
- 「Ultra Light ml_remote_id オプション」195 ページ
- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「Ultra Light 作成パラメータ」137 ページ

Ultra Light Java Edition の blob_file_base_dir オプション

データベースで BLOB ファイルの検索に使用されるベースディレクトリを設定します。

指定可能な値

文字列

デフォルト

BlackBerry スマートフォンのデフォルト値は "*file:///SDCard/*" です。その他のプラットフォームのデフォルト値は "" です。

備考

ファイル名の保管に使用したテーブルカラムは、文字列 **<base_dir>/<reference_to_external_file>** として格納されます。**<base_dir>** は、blobfile が置かれているベースディレクトリです。**<reference_to_external_file>** は、XML ファイルのコンテンツカラムで検索された正確な文字列です。

ベースディレクトリは、Ultra Light Java Edition データベースのロードユーティリティの -f オプションを使用して初期設定できます。関連ファイル名は、blob_file_base_dir オプションの値によって解決されます。ファイル名が **file:**// のプレフィクスで始まっていない場合、Ultra Light はファイルを開く前に、ファイル名の前にこのオプションの値を付加します。

参照

- 「Ultra Light Java Edition データベースロードユーティリティ (uljload)」252 ページ
- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_BLOB_FILE_BASE_DIR 変数 [BlackBerry] [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の date_format オプション

データベースで文字列変換に使用される日付形式を設定します。

指定可能な値

文字列

デフォルト

"YYYY-MM-DD"

備考

詳細については、「Ultra Light date_format 作成パラメータ」142 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_DATE_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の date_order オプション

データベースで文字列変換に使用される日付順を設定します。

指定可能な値

文字列

デフォルト

"YMD"

備考

詳細については、「Ultra Light date_order 作成パラメータ」145 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_DATE_ORDER 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の nearest_century オプション

データベースで文字列変換に使用される基準年を設定します。

指定可能な値

文字列

デフォルト

"50"

備考

詳細については、「Ultra Light nearest_century 作成パラメータ」149 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_NEAREST_CENTURY 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の precision オプション

データベースで文字列変換に使用される浮動小数点の精度を設定します。

指定可能な値

文字列

デフォルト

"30"

備考

詳細については、「Ultra Light precision 作成パラメータ」153 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_PRECISION 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の scale オプション

データベースによる文字列変換中に、計算結果が最大精度にトランケートされるときの小数点以下の最大桁数を設定します。

指定可能な値

文字列

デフォルト

"6"

備考

詳細については、「Ultra Light scale 作成パラメータ」154 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_SCALE 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の time_format オプション

データベースで文字列変換に使用される時間形式を設定します。

指定可能な値

文字列

デフォルト

"HH:NN:SS.SSS"

備考

詳細については、「Ultra Light time_format 作成パラメータ」155 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_TIME_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の timestamp_format オプション

データベースで文字列変換に使用されるタイムスタンプ形式を設定します。

指定可能な値

文字列

デフォルト

"YYYY-MM-DD HH:NN:SS.SSS"

備考

詳細については、「Ultra Light timestamp_format 作成パラメータ」157 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_TIMESTAMP_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の timestamp_increment オプション

2 つのユニークなタイムスタンプ間のマイクロ秒単位の最小差を設定します。

指定可能な値

文字列

デフォルト

"1"

備考

詳細については、「Ultra Light timestamp_increment 作成パラメータ」160 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_TIMESTAMP_INCREMENT 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition の timestamp_with_time_zone_format オプション

TIMESTAMP WITH TIME ZONE 値のタイムスタンプ形式を設定します。

指定可能な値

文字列

デフォルト

"YYYY-MM-DD HH:NN:SS.SSS+HH:NN"

備考

詳細については、「Ultra Light timestamp_with_time_zone_format 作成パラメータ」161 ページを参照してください。

参照

- 「データベースオプションへのアクセス」39 ページ
- 「データベースプロパティの読み取り」39 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- Connection.getDatabaseProperty メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.setOption メソッド [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_TIMESTAMP_WITH_TIME_ZONE_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light Java Edition のユーティリティ

Ultra Light Java Edition データベース上で保守と管理を実行するための Ultra Light ユーティリティが提供されています。

Ultra Light Java Edition データベース情報ユーティリティ (uljinfo)

既存の Ultra Light Java Edition データベースに関する情報を表示します。

構文

uljinfo -c *filename* **-p** *password* [*options*]

オプション	説明
-c *filename*	必須。検査する Ultra Light Java Edition データベースのファイル名を指定します。
-ek *key*	Ultra Light Java Edition データベースへのアクセスに必要な暗号化キーを指定します。
-p *password*	Ultra Light Java Edition データベースに接続するためのパスワードを指定します。デフォルトは **sql** です。
-v	冗長メッセージを表示します。
-?	コマンドラインの使用法を表示します。

例

次に、uljinfo プログラムからの出力例を示します。

```
C:¥ULj¥bin>uljinfo.cmd -c ..¥Samples¥Demo1.ulj -p sql
SQL Anywhere UltraLite J Database Information Utility
Database name:..¥Samples¥Demo1.ulj
Disk file:'..¥Samples¥Demo1.ulj'
Database ID: 0 0
Page size: 1024 1024
0 rows for next upload
Date format:YYYY-MM-DD
Date order:YMD
Nearest century: 50 50
Numeric precision: 30 30
Numeric scale: 6 6
Time format:HH:NN:SS.SSS
Timestamp format:YYYY-MM-DD HH:NN:SS.SSS
Timestamp increment: 1 1
Number of tables: 1 1
Number of columns: 2 2
Number of publications: 0 0
Number of tables that will always be uploaded: 0 0
Number of tables that are never synchronized: 0 0
Number of primary keys: 1 1
Number of foreign keys: 0 0
Number of indexes: 0 0
Last download occurred on Thu Jul 05 11:31:05 EDT 2007
Upload OK:true
```

Ultra Light Java Edition データベースロードユーティリティ (uljload)

XML ソースファイルから Ultra Light Java Edition データベースをロードする機能を提供します。XML ファイルは通常は uljunload ユーティリティで作成し、またカスタマイズが可能です。

注意
データベース設定パラメータは、コマンドラインでは設定できません。新しいデータベースを作成する場合、uljload ではデフォルト設定が使用されます。異なるデータベース設定パラメータ (異なるページサイズなど) を使用してデータベースを作成するには、データベースを Java で作成してから、**uljload -a** を使用して、作成したデータベースに XML コンテンツをロードします。

構文

uljload -c *filename* **-p** *password* [*options*] *inputfile*

オプション	説明
-a	XML ファイルから既存のデータベースに情報を追加します。このオプションを指定しないと、新しいデータベースが作成されます。
-c *filename*	必須。データベースファイルの名前を指定します。
-d	データのみをロードし、スキーマ情報を無視します。
-ek *key*	Ultra Light Java Edition データベースへのアクセスに必要な暗号化キーを指定します。

オプション	説明
-f *directory*	blob の最大サイズ (uljunload 操作の -b オプションによって指定) よりも大きなカラムのデータを取り出すディレクトリを指定します。
-i	アップロード同期用のローを挿入します。
-n	スキーマ情報だけをロードし、ローデータを無視します。
-p *password*	省略可。パスワードを指定します。デフォルトのパスワードは **sql** です。
-q	ユーティリティをクワイエットモードで実行します (メッセージを表示しません)。
-v	冗長メッセージを表示します。
-y	出力ファイルが存在する場合、それを上書きします (-a オプションは指定しません)。
-z *pagesize*	作成するデータベースのページサイズ (バイト単位) を指定します。
-?	コマンドラインの使用法を表示します。
inputfile	XML 文を含む入力ファイルを指定します。

blobfile 型のロード例

この例は、次の SQL 文を Ultra Light Java Edition データベースに適用し、Ultra Light Java Edition データベースのアンロードユーティリティを使用してファイルをアンロードしたことを前提としています。

```
CREATE TABLE blobfile_example
   file_name CHAR(size) DEFAULT AUTOFILENAME( prefix, extension ),
   file_contents LONG BINARY STORE AS FILE( file_name ) CASCADE DELETE
```

ulload ユーティリティでは、uljunload ユーティリティによってエクスポートされたファイルが検出されると、file_name カラムが CHAR カラムとして処理され、file_contents カラムが LONG BINARY カラムとして処理されます。

uljload ユーティリティで同じ XML ファイルが検出されると、blobfile 型が再構築されます。file_name カラムには、文字列 **<base_dir>/<reference_to_external_file>** が格納されます。**<base_dir>** は、-f オプションで指定された文字列であり、**<reference_to_external_file>** は、XML ファイルの file_contents カラムで検出された文字列です。新しく再構築されたデータベースには、外部ファイルに対する有効な参照が含まれます。

生成されたファイル名が長すぎるとロードに失敗し、エラーメッセージが表示され、file_name カラムのサイズを増加するように要求されます。

参照

- 「Ultra Light Java Edition データベースアンロードユーティリティ (uljunload)」254 ページ

例

-? オプションは、コマンドライン情報を表示します。

```
uljload -?
```

uljload ユーティリティコマンドラインに **-?** オプションを指定すると、次の使用法が表示されます。

```
SQL Anywhere Ultra Light J データベースロードユーティリティ
使用法：uljload [options] <XML ファイル>
新しい Ultra Light J データベースを作成し、<XML ファイル> からデータをロードします。

オプション：
-a     既存のデータベースに追加します。
-c     <ファイル> データベースファイル。
-d     データのみ -- スキーマは無視します。
-ek    <キー> 暗号化キー。
-f     <ディレクトリ>
       ロードするディレクトリを指定します。ファイルパス。
-i     アップロード同期のローを挿入します。
-n     スキーマのみ -- データは無視します。
-p     データベースに接続するためのパスワード。
-q     クワイエット：メッセージを出力しません。
-v     冗長メッセージ。
-y     ファイルがすでに存在する場合は上書きします。
-z     <ページサイズ>
       データベース作成のページサイズ。
```

Ultra Light Java Edition データベースアンロードユーティリティ (uljunload)

Ultra Light Java Edition データベース (データとスキーマのいずれか、またはその両方) を XML ファイルにアンロードする機能を提供します。

構文

uljunload -c *filename* **-p** *password* [*options*] *outputfile*

オプション	説明
-b *max-blob-size*	XML に出力される blob または char データの最大サイズ (バイト単位) を指定します。
-c *filename*	必須。アンロードするデータベースファイルの名前を指定します。
-d	データのみをアンロードし、スキーマ情報は出力しません。
-e *table, ...*	リスト内に指定されたテーブルのデータを除外します。

オプション	説明
-ek *key*	Ultra Light Java Edition データベースへのアクセスに必要な暗号化キーを指定します。
-f *directory*	blob の最大サイズ (-b オプションによって指定) よりも大きなカラムのデータを格納するディレクトリを指定します。
-n	スキーマ情報のみをアンロードし、データは出力しません。
-p *password*	データベースへの接続に使用するパスワードを指定します。デフォルトは **sql** です。
-q	クワイエットモードで実行します (メッセージは表示されません)。
-t *table, ...*	リストで名前が指定されたテーブルのデータのみを出力します。
-v	冗長メッセージを表示します。
-y	出力ファイルがすでに存在する場合、それを上書きします。
-?	オプションの使用法とヘルプ情報を表示します。
outputfile	出力ファイルの名前です (このファイルにはデータベースの内容を記述する XML 文が含まれています)。

blobfile 型のアンロード例

blobfile 型をアンロードするには、次のサンプル SQL 文をデータベースに適用します。

```
CREATE TABLE blobfile_example
   file_name CHAR(size) DEFAULT AUTOFILENAME( prefix, extension ),
   file_contents LONG BINARY STORE AS FILE( file_name ) CASCADE DELETE
```

この例を適用して ULjUnload ユーティリティを使用すると、file_name カラムは通常の CHAR としてエクスポートされますが、プレフィクスと拡張子の文字列を 'prefix', 'extension' の形式で格納する属性 default_autofilename が追加されます。file_contents カラムは LONG BINARY としてエクスポートされますが、参照される file_name カラムの名前を格納する属性 filename_col が追加されます。blobfile の各カラムを含むローでは、file_name カラムの内容に変更はありません。file_contents カラムは、外部的に格納された blob カラムのように動作し、次のような形式になります。

```
file_contents.File="tablename-columnname-rownumber.bin"
```

file_contents カラムの内容は、-f オプションで指定されたロケーションに *.bin* ファイルとして保存されます。

参照

- 「Ultra Light Java Edition データベースアンロードユーティリティ (uljunload)」254 ページ

XML ファイルの内容の例


```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<ul:ulschema xmlns:ul="urn:ultralite">
 <collation name="1252LATIN1" case_sensitive="no"/>
 <options>
  <option name="dateformat" value="YYYY-MM-DD"/>
  <option name="dateorder" value="YMD"/>
  <option name="nearestcentury" value="50"/>
  <option name="precision" value="30"/>
  <option name="scale" value="6"/>
  <option name="timeformat" value="HH:NN:SS.SSS"/>
  <option name="timestampformat" value="YYYY-MM-DD HH:NN:SS.SSS"/>
  <option name="timestampincrement" value="1"/>
 </options>
 <tables>
  <table name="ULCustomer" sync="changes">
   <columns>
    <column name="cust_id" type="integer" null="no"/>
    <column name="cust_name" type="char(30)" null="yes"/>
   </columns>
   <primarykey>
    <primarycolumn name="cust_id" direction="asc"/>
   </primarykey>
   <indexes/>
  </table>
 </tables>
 <uldata>
  <table name="ULCustomer">
   <row cust_id="2000" cust_name="Apple St. Builders"/>
   <row cust_id="2001" cust_name="Art's Renovations"/>
   <row cust_id="2002" cust_name="Awnings R Us"/>
   <row cust_id="2003" cust_name="Al's Interior Design"/>
   <row cust_id="2004" cust_name="Alpha Hardware"/>
   <row cust_id="2005" cust_name="Ace Properties"/>
   <row cust_id="2006" cust_name="A1 Contracting"/>
   <row cust_id="2007" cust_name="Archibald Inc."/>
   <row cust_id="2008" cust_name="Acme Construction"/>
   <row cust_id="2009" cust_name="ABCXYZ Inc."/>
   <row cust_id="2010" cust_name="Buy It Co."/>
   <row cust_id="2011" cust_name="Bill's Cages"/>
   <row cust_id="2012" cust_name="Build-It Co."/>
   <row cust_id="2013" cust_name="Bass Interiors"/>
   <row cust_id="2014" cust_name="Burger Franchise"/>
   <row cust_id="2015" cust_name="Big City Builders"/>
   <row cust_id="2016" cust_name="Bob's Renovations"/>
   <row cust_id="2017" cust_name="Basements R Us"/>
   <row cust_id="2018" cust_name="BB Interior Design"/>
   <row cust_id="2019" cust_name="Bond Hardware"/>
   <row cust_id="2020" cust_name="Cat Properties"/>
   <row cust_id="2021" cust_name="C &amp;amp; C Contracting"/>
   <row cust_id="2022" cust_name="Classy Inc."/>
   <row cust_id="2023" cust_name="Cooper Construction"/>
   <row cust_id="2024" cust_name="City Schools"/>
   <row cust_id="2025" cust_name="Can Do It Co."/>
   <row cust_id="2026" cust_name="City Corrections"/>
   <row cust_id="2027" cust_name="City Sports Arenas"/>
   <row cust_id="2028" cust_name="Cantaloupe Interiors"/>
   <row cust_id="2029" cust_name="Chicken Franchise"/>
  </table>
 </uldata>
</ul:ulschema>
```

Ultra Light Java Edition データベース転送ユーティリティ

ULjDbT ユーティリティは、Ultra Light データベースを BlackBerry スマートフォンから、デスクトップ、ラップトップ、サーバなどの外部デバイスに転送する機能を提供します。また、データベースの削除、データベース情報の表示、データベース転送ログの表示や電子メール送信も実行できます。このユーティリティは、Ultra Light Java Edition データベース転送デスクトップアプリケーション (ULjDbT) と BlackBerry スマートフォンクライアントアプリケーション (*ULjDatabaseTransfer.cod*) という同時に実行する必要のある 2 つのアプリケーションで構成されています。

このデスクトップアプリケーションは、USB 接続または HTTP 接続を使用して Ultra Light Java Edition データベースを受信します。サーバアプリケーションを起動すると、サーバアプリケーションは、クライアントアプリケーションとの指定された接続を通じて BlackBerry スマートフォンからデータベースが転送されるのを待機します。接続は、アプリケーションがタイムアウトになったときにアプリケーションインタフェースを使用して手動で切断されるか、または転送が完了したときに切断されます。

BlackBerry スマートフォンクライアントアプリケーションは、USB ケーブル、またはデスクトップアプリケーションに対して指定された TCP ポートを使用して、Ultra Light Java Edition データベースを送信します。

クライアントアプリケーションの起動

uljdbtserv ユーティリティを使用して、Ultra Light Java Edition データベースの削除、表示、外部デバイスへの転送、または電子メール送信を行います。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

クライアントアプリケーションは、SQL Anywhere インストールディレクトリの *UltraLite¥UltraLiteJ¥BlackBerry4.2* ディレクトリにある署名付きファイルです。

♦ タスク

1. SQL Anywhere インストールディレクトリの *UltraLite¥UltraLiteJ¥BlackBerry4.2* ディレクトリから *ULjDatabaseTransfer.cod* をロードします。

   クライアントアプリケーションのアイコンがアプリケーションのリストに表示されます。

2. アプリケーションを起動し、トラックホイールを押します。

3. **[データベース接続]** 画面で、次のフィールドを完成させます。

   - **[データベース名]**　外部デバイスに転送するデータベースの名前。名前が *file://* (大文字と小文字が区別される) で始まる場合は、クライアントアプリケーションによってデータベースがファイルシステムで検索されます。そうでない場合は、データベースがオブジェクトストアで検索されます。

- **[データベースのパスワード]**　データ転送を許可するデータベースパスワード。**[データベースのパスワード]** を空白にすると、デフォルトのパスワードが使用されます。
- **[暗号化キー]**　データベースの暗号化に使用される暗号化キー。このオプションが必要なのは、データベースを暗号化する場合だけです。

4. **[次へ]** をクリックします。

結果

[アクション] 画面が表示されます。この画面からすべてのクライアントアプリケーション機能にアクセスできます。

Ultra Light Java Edition のデータベースの転送

BlackBerry スマートフォンクライアントアプリケーションを使用してデータベースを転送します。

前提条件

この作業を実行するための前提条件はありません。

♦ タスク

1. BlackBerry スマートフォンクライアントで、*%SQLANY16%¥Bin32¥uljdbtserv.cmd* にある uljdbt ユーティリティを実行します。
2. BlackBerry の **[アクション]** 画面で、目的の接続方法 (USB または HTTP) を選択します。
3. USB 転送の場合は、**[USB を使用してデータベースをサーバに転送します]** をクリックします。HTTP 転送の場合は、次の手順に進みます。
4. 指示に従って、データベース転送デスクトップアプリケーションを起動します。

> **注意**
> データベースを正常に転送するには、デバイスまたはシミュレータが BlackBerry Device Manager に接続されていることを確認してください。シミュレータの場合は、**[USB Cable Connected]** を使用して USB 接続がシミュレートされていることを確認してください。

   a. クライアントアプリケーションで **[次へ]** をクリックします。
   b. デスクトップアプリケーションで、**[USB]** が選択されていることを確認し、**[開始]** をクリックします。

      BlackBerry スマートフォンで、外部デバイスへのデータベースの転送が開始されます。進行状況はデスクトップアプリケーションに表示されます。
   c. クライアントアプリケーションとデスクトップアプリケーションの両方で **[OK]** をクリックして閉じます。

5. HTTP 転送の場合は、**[HTTP を使用してデータベースをサーバに転送します]** をクリックします。

a. **[HTTP 転送]** 画面で、**[次へ]** をクリックします。

b. 次の値を指定します。

- **[ホスト]**　デスクトップの IP アドレス。
- **[ポート]**　デスクトップアプリケーションの **[接続プロパティ]** で指定されたポート。
- **[URL サフィックス]**　**http://** サフィックス (必須) を含む、転送を受信するサーバのホスト名。

c. **[次へ]** をクリックします。

> **注意**
> 個人特定不可デバイス上の BES を介して転送する場合、**[URL サフィックス]** は空のままにします。個人特定可能デバイスでは、サフィックス **;deviceside=false** を使用します。
>
> ダイレクト TCP 経由で転送する場合、サフィックス **;deviceside=true** を使用します。この機能をサポートしていない通信事業者もあります。
>
> 通信事業者の WAP ゲートウェイについて APN 情報がわかっている場合は、その WAP ゲートウェイを使用できることがあります。その情報もサフィックスに付加する必要があります。BES 経由の場合であっても、Ultra Light Java Edition データベース転送ユーティリティを実行しているデバイスと BES との間にファイアウォールがある可能性があります。その場合は SSL トンネルを使用する必要があります。**[HTTP Transfer Parms]** 画面で、ファイアウォールの BES 側で実行している SSL サーバのポートと名前または IP アドレスを指定します。また、転送アプリケーションに SSL クライアントがマッピングされているポートを指定する必要もあります。
>
> データベースを BlackBerry シミュレータから転送する場合は、BlackBerry MDS シミュレータが実行されているか、または BlackBerry シミュレータで実行されている Ultra Light Java Edition データベース転送ユーティリティに対して URL サフィックス **;deviceside=true** を指定する必要があります。

d. デスクトップアプリケーションで、**[HTTP]** が選択されていることを確認し、**[開始]** をクリックします。

e. クライアントアプリケーションで **[次へ]** をクリックします。

BlackBerry スマートフォンで、外部デバイスへのデータベースの転送が開始されます。進行状況はデスクトップアプリケーションに表示されます。

f. クライアントアプリケーションとデスクトップアプリケーションの両方で **[OK]** をクリックして閉じます。

結果

データベースが転送されます。

Ultra Light Java Edition のデータベースの受信

Ultra Light Java Edition データベース転送ユーティリティを使用してデータベースを受信します。

前提条件

この作業を実行するための前提条件はありません。

♦ タスク

1. SQL Anywhere のインストールディレクトリの *Bin32* ディレクトリから *ULjDbTServ.cmd* を実行します。
2. **[接続]** タブで、**[接続方法]** をクリックします。
3. **[接続プロパティ]** の下で、次の値を指定します。
   - **[ポート]**　このフィールドは、HTTP 接続だけに適用されます。BlackBerry スマートフォンが接続する TCP ポート番号を入力します。通常、このポート番号は BlackBerry スマートフォン上で実行中の Ultra Light Java Edition データベース転送ユーティリティに指定されているポート番号と一致します。ただし、SSL を使用している場合は、ポート番号が異なることがあります。
   - **[Blackberry のパスワード]**　このフィールドは、USB 接続だけに適用されます。接続している BlackBerry スマートフォンがロックされている場合、アクセスするためのパスワードを入力します。パスワードがない場合は、このフィールドを空白のままにします。
   - **[タイムアウト]**　サーバアプリケーションがタイムアウトし、接続を切断するまでの分単位のアイドル時間。
   - **[出力]**　転送されたデータベースを保存するファイル名とロケーションを指定します。
4. **[開始]** をクリックして BlackBerry スマートフォンへの接続を開きます。

   サーバアプリケーションは、タイムアウトするか、接続が確立するまで待機します。既存のファイルを指定した場合は、そのファイルを上書きするかどうかという質問が表示されます。

   [ログ] タブに、エラーメッセージを含む、サーバの状態と転送の進行状況に関する詳細が表示されます。

結果

データベースが受信されます。

Ultra Light Java Edition のデータベースの削除

データベースを削除します。

前提条件

この作業を実行するための前提条件はありません。

♦ タスク

1. BlackBerry の **[アクション]** 画面で、**[データベースを削除します]** をクリックします。

2. 確認ウィンドウで、**[削除]** をクリックしてデータベースを削除します。
3. **[データベースは削除されました。]** ウィンドウで、**[OK]** をクリックしてクライアントを閉じます。

結果

データベースは削除されます。

Ultra Light Java Edition データベースの情報を表示します。

データベース情報を表示します。

前提条件

この作業を実行するための前提条件はありません。

♦ タスク

1. BlackBerry の **[アクション]** 画面で、**[データベース情報を表示します]** をクリックします。スクロールしてすべてのデータベース情報を確認します。
2. **[前へ]** をクリックして **[アクション]** 画面に戻ります。

結果

データベースの情報を読むことができます。

ログファイルの表示

ログファイルを表示します。

前提条件

この作業を実行するための前提条件はありません。

♦ タスク

1. BlackBerry の**[データベース接続]** 画面で、メニューを表示します。
2. **[ログ]** をクリックします。[ログ] 画面が表示されます。
3. ログファイルを電子メールで送信するには、ログの送信先の電子メールアドレスを入力して、**[電子メールを送信します]** をクリックします。前の画面に戻るには、Enter キーを押します。

結果

ログファイルを読み込むことができます。

Ultra Light Java Edition システムテーブル

この項では、Ultra Light Java Edition データベースで利用できるシステムテーブルを示します。

sysarticles システムテーブル

sysarticles システムテーブルの各ローは、パブリケーションに属するテーブルを示します。

カラム名	カラム型	説明
publication _id	UNSIGNED INTEGER	このアーティクルが属するパブリケーションの識別子。
table_id	UNSIGNED INTEGER	パブリケーションに属するテーブルの識別子。
predicate	VARCHAR(256)	CREATE PUBLICATION 文または ALTER PUBLICATION 文の関連する WHERE 句で指定された述部の式。

制約

PRIMARY KEY (publication_id, table_id)

syscolumn システムテーブル

syscolumn システムテーブルの各ローは、カラムを示します。

カラム名	カラム型	説明
table_id	UNSIGNED INTEGER	カラムが属するテーブルの識別子。
column_id	UNSIGNED INTEGER	カラムのユニークな識別子。
column_name	VARCHAR(128)	カラム名。 Domain インタフェース [Ultra Light J] 『Ultra Light Java プログラミング』を参照してください。
column_flags	TINY	属性を表す次のフラグのビット単位の組み合わせ。 ● **0x01** カラムはプライマリキーに含まれます。 ● **0x02** カラムは NULL 入力可です。

カラム名	カラム型	説明
column_domain	TINY	下位 6 ビットのカラムのドメインを示す列挙値。残りのビットは、内部的に使用されます。
column_length	UNSIGNED SHORT	カラムの長さ。 VARCHAR 型と BINARY 型 (Domain インタフェースで定義) のカラムの場合は、バイト単位の最大長です。NUMERIC 型のカラムの場合は、最初のバイトに精度、2 番目のバイトに位取りが格納されます。
column_default_value	VARCHAR(128)	このカラムのデフォルト値。ColumnSchema インタフェースの COLUMN_DEFAULT 値の 1 つで指定されます。たとえば、COLUMN_DEFAULT_AUTOINC はオートインクリメントするデフォルト値を表します。 varchar カラムに DEFAULT AUTOFILENAME が指定されていると、prefix|extension のエンコードを使用してパラメータのプレフィクスと拡張子が格納されます。
filename_colid	UNSIGNED INTEGER	STORE AS FILE として宣言された LONG BINARY カラム用のファイル名を含むカラムのカラム ID を格納します。それ以外の場合、このカラムは NULL です。

制約

PRIMARY KEY (table_id, column_id)

sysforeignkey システムテーブル

sysforeignkey システムテーブルの各ローは、テーブルに属する外部キーを示します。

カラム名	カラム型	説明
table_id	UNSIGNED INTEGER	外部キーが属するテーブルの識別子。
foreign_table_id	UNSIGNED SHORT	この外部キーカラムが参照するテーブルの識別子。
foreign_key_id	UNSIGNED INTEGER	外部キーの識別子。
name	VARCHAR(128)	外部キーの名前。

カラム名	カラム型	説明
index_name	VARCHAR(128)	外部キーが参照しているインデックスの名前。

制約

PRIMARY KEY (table_id, foreign_key_id)

sysfkcol システムテーブル

sysfkcol システムテーブルの各ローは、外部キーのカラムを示します。

カラム名	カラム型	説明
table_id	UNSIGNED INTEGER	外部キーが適用されるテーブルのユニークな識別子。
foreign_key_id	UNSIGNED INTEGER	このカラムが属する外部キーのユニークな識別子。
item_no	UNSIGNED SHORT	外部キー内のカラムの順序。
column_id	UNSIGNED SHORT	外部カラムを参照するテーブルカラムのユニークな識別子。
foreign_column_id	UNSIGNED SHORT	参照されているテーブルカラムのユニークな識別子。

制約

PRIMARY KEY (table_id, foreign_key_id, item_no)

sysindex システムテーブル

sysindex システムテーブルの各ローは、データベース内のインデックスを示します。

カラム名	カラム型	説明
table_id	UNSIGNED INTEGER	インデックスが適用されるテーブルのユニークな識別子。
index_id	UNSIGNED INTEGER	インデックスのユニークな識別子。
index_name	VARCHAR(128)	インデックス名。

カラム名	カラム型	説明
index_flags	TINY	インデックスのタイプとその永続性を示す次のフラグのビット単位の組み合わせ。 ● **0x01** ユニークキー。 ● **0x02** ユニークインデックス。 ● **0x04** インデックスは永続的。 ● **0x08** プライマリキー。
index_data	UNSIGNED INTEGER	内部使用のみ。
hash_size	UNSIGNED SHORT	インデックスのハッシュに使用されるハッシュサイズ。

制約

PRIMARY KEY (table_id, index_id)

sysindexcolumn システムテーブル

sysindexcolumn システムテーブルの各ローは、sysindex にリストされているインデックスのカラムを示します。

カラム名	カラム型	説明
table_id	UNSIGNED INTEGER	インデックスが適用されるテーブルのユニークな識別子。
index_id	UNSIGNED INTEGER	このインデックスカラムが属するインデックスのユニークな識別子。
order	UNSIGNED INTEGER	インデックス内のカラムの順序。
column_id	UNSIGNED SHORT	インデックス対象カラムのユニークな識別子。
index_column_flag	TINY	インデックス内のカラムが保持されている場所 (昇順 (1) または降順 (0)) を示します。

制約

PRIMARY KEY (table_id, index_id, order)

syspublications システムテーブル

syspublications システムテーブルの各ローは、パブリケーションを示します。

カラム名	カラム型	説明
publication_id	UNSIGNED INTEGER	パブリケーションのユニークな識別子。
publication_name	VARCHAR(128)	パブリケーションの名前。
download_timestamp	TIMESTAMP	最後のダウンロードの時刻。
last_sync_sent	UNSIGNED INTEGER	Mobile Link に送信されたアップロードを追跡する整数。
last_sync_confirmed	UNSIGNED INTEGER	Mobile Link での受信が確認されたアップロードを追跡する整数。

制約

PRIMARY KEY (publication_id)

systable システムテーブル

systable システムテーブルの各ローは、データベース内のテーブル 1 つを示します。

カラム名	カラム型	説明
table_id	UNSIGNED INTEGER	テーブルのユニークな識別子。
table_name	VARCHAR(128)	テーブル名。
table_flags	UNSIGNED SHORT	次のいずれかのフラグのビット単位の組み合わせ。 ● TableSchema.TABLE_IS_SYSTEM ● TableSchema.TABLE_IS_NO_SYNC ● TableSchema.TABLE_IS_DOWNLOAD_ONLY
table_data	UNSIGNED INTEGER	内部使用のみ。
table_partition_size	UNSIGNED INTEGER	指定されている場合、またはテーブルにデフォルト値が含まれている場合は、テーブルの分割サイズ。そうでない場合は NULL。 分割サイズが範囲外の場合、この値は 0 になります。

制約

PRIMARY INDEX (table_id)

sysuldata システムテーブル

sysuldata システムテーブルの各ローは、オプションとプロパティの値の組み合わせを示します。

同期プロファイルは、**name** がプロファイルの名前、**type** が **ulsync**、**long_setting** がプロファイル文字列の UTF-8 エンコードにそれぞれ設定されて、sysuldata システムテーブルに格納されます。1 つのカラム設定と **long_setting** に、使用法に応じて値が割り当てられます。

カラム名	カラム型	説明
long_setting	LONGBINARY	長い値用の BLOB。
name	VARCHAR(128)	プロパティ名。
setting	VARCHAR(128)	プロパティの値。
type	VARCHAR(128)	**opt** (オプション)、または **prop** (プロパティ) のいずれか。

制約

PRIMARY KEY (name, type)

参照

- Connection.OPTION_BLOB_FILE_BASE_DIR 変数 [BlackBerry] [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_DATABASE_ID 変数 [BlackBerry] [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_DATE_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_DATE_ORDER 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_ML_REMOTE_ID 変数 [BlackBerry] [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_NEAREST_CENTURY 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_PRECISION 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_SCALE 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_TIME_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_TIMESTAMP_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_TIMESTAMP_INCREMENT 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.OPTION_TIMESTAMP_WITH_TIME_ZONE_FORMAT 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.PROPERTY_DATABASE_NAME 変数 [Ultra Light J]『Ultra Light Java プログラミング』
- Connection.PROPERTY_PAGE_SIZE 変数 [Ultra Light J]『Ultra Light Java プログラミング』

Ultra Light SQL リファレンス

この項は、Ultra Light SQL のリファレンスです。Ultra Light SQL は、SQL Anywhere データベースでサポートされている SQL のユニークなサブセットです。

Ultra Light SQL 言語の要素

Ultra Light のキーワード

各 SQL 文には 1 つまたは複数のキーワードが含まれています。SQL 文のキーワードでは大文字と小文字を区別しませんが、このマニュアルではキーワードを大文字で表記します。キーワードの中には、二重引用符で囲まないと識別子として使用できないものがあります。このようなキーワードを「予約語」と呼びます。

注意
Ultra Light では、SQL Anywhere キーワードのサブセットだけがサポートされています。ただし、将来のリリースで問題が発生しないように、SQL Anywhere のすべての予約語が Ultra Light の予約語でもあると考えてください。

参照

- 「予約語」『SQL Anywhere サーバ SQL リファレンス』

Ultra Light の識別子

「識別子」は、ユーザ ID、テーブル、カラムなど、データベースのオブジェクト名を表します。識別子の最大長は 128 バイトです。

次のいずれかの条件と一致した場合に、識別子を二重引用符で囲みます。

- 識別子にスペースが含まれる。
- 識別子の先頭文字がアルファベット文字ではないデータベースの照合順は、どの文字がアルファベットまたは数字として扱われるかを指定する。
- 識別子に予約語が含まれる。
- 識別子にアルファベット文字や数字以外の文字が含まれる。

1 つの円記号は、エスケープ文字として使用される場合のみ識別子で使用できます。

参照

- 「予約語」『SQL Anywhere サーバ SQL リファレンス』

Ultra Light の文字列

文字列を使用して、文字データをデータベースに格納します。Ultra Light は、SQL Anywhere と同じ文字列の規則をサポートしています。文字列の比較結果や文字列のソート順は、データベース、文字セット、照合順における大文字と小文字の区別によって決まります。これらのプロパティは、データベースの作成時に設定されます。

参照

- 「文字列」『SQL Anywhere サーバ SQL リファレンス』
- 「Ultra Light 文字セット」26 ページ

Ultra Light のコメント

コメントは、SQL 文または文ブロックに説明テキストを付加するために使用します。Ultra Light ランタイムは、コメントを実行しません。

Ultra Light では次のコメントインジケータを使用できます。

- **-- (二重ハイフン)**　データベースサーバは、この行の残りの文字を無視します。このインジケータは、SQL/2003 のコメントインジケータです。
- **// (二重スラッシュ)**　二重スラッシュは、二重ハイフンと同じ意味です。
- **/* ... */ (スラッシュ ー アスタリスク)**　2 つのコメントマーカの間にある文字は、すべて無視されます。2 つのコメントマーカは、同じ行にあっても、別の行にあってもかまいません。このスタイルで示されたコメントは、ネストできます。このスタイルは、C スタイルコメントとも呼ばれます。

> **注意**
> パーセント記号 (%) は、Ultra Light ではサポートされていません。

例

●次に、二重ハイフンのコメントの使用例を示します。

```
CREATE TABLE borrowed_book (
  loaner_name CHAR(100)     PRIMARY KEY,
  date_borrowed DATE NOT NULL DEFAULT CURRENT DATE,
  date_returned       DATE,
  book              CHAR(20)
  FOREIGN KEY book REFERENCES library_books (isbn),
)
--This statement creates a table for a library database to hold information on borrowed books.
--The default value for date_borrowed indicates that the book is borrowed on the day the entry is
made.
--The date_returned column is NULL until the book is returned.
```

●次に、C スタイルコメントの使用例を示します。

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

```
  date_returned      DATE,
  book               CHAR(20)
  FOREIGN KEY book REFERENCES library_books (isbn),
)
/* This statement creates a table for a library database to hold information on borrowed books.
The default value for date_borrowed indicates that the book is borrowed on the day the entry is
made.
The date_returned column is NULL until the book is returned. */
```

Ultra Light の数値

数値を使用して、数値データをデータベースに格納します。数値は次のようなデータです。

- 任意の数字の並び
- 小数部が続く
- オプションでマイナス記号 (-) またはプラス記号 (+) を含む
- E の後に指数値が続く

たとえば、次に示す数値はすべて Ultra Light でサポートされています。

42

-4.038

.001

3.4e10

1e-10

Ultra Light の NULL 値

SQL Anywhere と同様に、NULL はすべてのデータ型のすべての有効な値とは異なる特殊な値です。ただし、NULL 値はすべてのデータ型で使用できます。NULL は、情報が不明 (値なし) であるか、該当しないことを表すために使用します。

参照

- 「NULL 特別値」『SQL Anywhere サーバ SQL リファレンス』

Ultra Light の特別値

特別値は、式や、テーブル作成時のカラムのデフォルトに使用できます。

CURRENT DATE 特別値

現在の年、月、日を返します。

データ型

DATE

備考

返される日付は、SQL 文が Ultra Light ランタイムで実行されたときのシステムクロックの読み取り値が基準になります。CURRENT DATE を次のいずれかの状態で使用すると、すべての値は、個別のクロック読み取り値が基準になります。

- 同一文で CURRENT DATE を複数回使用
- 同一文で CURRENT DATE を CURRENT TIME または CURRENT TIMESTAMP と組み合わせて使用
- 同一文で CURRENT DATE を NOW 関数または GETDATE 関数と組み合わせて使用

参照

- 「Ultra Light の式」275 ページ
- 「GETDATE 関数 [日付と時刻]」353 ページ
- 「NOW 関数 [日付と時刻]」375 ページ

CURRENT TIME 特別値

現在の時、分、秒 (小数位あり) で構成される時刻を返します。

データ型

TIME

備考

秒は小数第 6 位まで格納されます。現在時刻の精度はシステムクロックの精度によって制限されます。

返される日付は、SQL 文が Ultra Light ランタイムで実行されたときのシステムクロックの読み取り値が基準になります。CURRENT TIME を次のいずれかと組み合わせて使用すると、すべての値は、個別のクロック読み取り値が基準になります。

- 同一文で CURRENT TIME を複数回使用
- 同一文で CURRENT TIME を CURRENT DATE または CURRENT TIMESTAMP と組み合わせて使用
- 同一文で CURRENT TIME を NOW 関数または GETDATE 関数と組み合わせて使用

参照

- 「Ultra Light の式」275 ページ
- 「GETDATE 関数 [日付と時刻]」353 ページ
- 「NOW 関数 [日付と時刻]」375 ページ

CURRENT TIMESTAMP 特別値

CURRENT DATE と CURRENT TIME を結合して、TIMESTAMP 値を形成します。年、月、日、時、分、秒、秒の小数位で構成されます。

データ型

TIMESTAMP

備考

秒は小数第 3 位まで格納されます。精度はシステムクロックの精度によって制限されます。

DEFAULT CURRENT TIMESTAMP で宣言されたカラムには、ユニークな値が入るとはかぎりません。

CURRENT TIMESTAMP が返す情報は、GETDATE 関数と NOW 関数が返す情報と同じです。

CURRENT_TIMESTAMP は、CURRENT TIMESTAMP と同じです。

返される日付は、SQL 文が Ultra Light ランタイムで実行されたときのシステムクロックの読み取り値が基準になります。CURRENT TIMESTAMP を次のいずれかと組み合わせて使用すると、すべての値は、個別のクロック読み取り値が基準になります。

- 同一文で CURRENT TIMESTAMP を複数回使用
- 同一文で CURRENT TIMESTAMP を CURRENT DATE または CURRENT TIME と組み合わせて使用
- 同一文で CURRENT TIMESTAMP を NOW 関数または GETDATE 関数と組み合わせて使用

参照

- 「CURRENT TIME 特別値」272 ページ
- 「Ultra Light の式」275 ページ
- 「NOW 関数 [日付と時刻]」375 ページ
- 「GETDATE 関数 [日付と時刻]」353 ページ
- 「NOW 関数 [日付と時刻]」375 ページ

CURRENT UTC TIMESTAMP 特別値

年、月、日で構成される現在の UTC 時間を示す TIMESTAMP WITH TIME ZONE 値を返します。

データ型

DATE

備考

返される日付は、SQL 文が Ultra Light ランタイムで実行されたときのシステムクロックの読み取り値が基準になります。

- 同一文で CURRENT DATE を複数回使用
- 同一文で CURRENT DATE を CURRENT TIME または CURRENT TIMESTAMP と組み合わせて使用
- 同一文で CURRENT DATE を NOW 関数または GETDATE 関数と組み合わせて使用

参照

- 「Ultra Light の式」275 ページ
- 「GETDATE 関数 [日付と時刻]」353 ページ
- 「CURRENT TIMESTAMP 特別値」273 ページ
- 「NOW 関数 [日付と時刻]」375 ページ

SQLCODE 特別値

SQLCODE 特別値が評価された時点の値です。Ultra Light Java Edition データベースではサポートされていません。

データ型

文字列

備考

SQLCODE 値は各文の後に設定されます。SQLCODE をチェックして、文の実行が成功したかどうかを確認できます。

参照

- 「Ultra Light の式」275 ページ
- エラーメッセージ

例

SELECT 文を使用して、結果セットから新しいローをフェッチしようとするたびにエラーコードを生成します。次に例を示します。SELECT a, b, SQLCODE FROM MyTable.

Ultra Light の日付と時刻

日付と時刻の関数の多くは、日付と時刻の単位で構成される日付を使用します。Ultra Light と SQL Anywhere では同じ日付の単位がサポートされています。

参照

- 「日付の単位の指定」314 ページ

Ultra Light の式

式は、多くの場合、カラム参照の形式でデータを関数や演算子と組み合わせることによって形成されます。

構文

expression:
case-expression
| *constant*
| [*correlation-name*.]*column-name*
| - *expression*
| *expression operator expression*
| (*expression*)
| *function-name* (*expression*, ...)
| *if-expression*
| *special value*
| *input-parameter*

パラメータ

case-expression:
CASE *expression*
WHEN *expression*
THEN *expression*,...
[**ELSE** *expression*]
END

alternative form of case-expression:
CASE
WHEN *search-condition*
THEN *expression*,...
[**ELSE** *expression*]
END

constant:
integer | *number* | *string* | *host-variable*

special-value:
CURRENT { **DATE** | **TIME** | **TIMESTAMP** }
| **NULL**
| **SQLCODE**
| **SQLSTATE**

if-expression:
IF *condition*
THEN *expression*
[**ELSE** *expression*]
ENDIF

input-parameter:
{ **?** | :*name* [: *indicator-name*] }

operator:
{ **+** | **-** | ***** | **/** | **||** | **%** }

参照

- 「式内の定数」276 ページ
- 「Ultra Light の特別値」271 ページ
- 「式内のカラム名」276 ページ
- 「Ultra Light SQL 関数」313 ページ
- 「式のサブクエリ」279 ページ
- 「Ultra Light の探索条件」281 ページ
- 「Ultra Light SQL データ型」293 ページ
- 「CASE 式」277 ページ
- 「入力パラメータ」280 ページ

式内の定数

Ultra Light における定数とは、数値または文字列リテラルです。

構文

' *constant* '

使用法

文字列定数は、一重引用符 (') で囲まれています。

文字列内にアポストロフィを表すには、一重引用符を 2 つ続けて使用します (')。

参照

- 「エスケープシーケンス」『SQL Anywhere サーバ SQL リファレンス』

例

所有を表すフレーズを使用するには、次のような文字列リテラルを入力します。

'John''s database'

式内のカラム名

式内の識別子です。

構文

correlation-name.column-name

備考

カラム名は、オプションの相関名 (通常はテーブルの名前) の後に続きます。

カラム名がキーワードであるか、カラム名に英字、数字、アンダースコア以外の文字が使用されている場合は、二重引用符 ("") で囲んでください。次の例は、有効なカラム名です。

```
Employees.Name
address
```

```
"date hired"
"salary"."date paid"
```

参照

- 「FROM 句 [Ultra Light]」443 ページ

IF 式

データの特定サブセットを返す探索条件を設定します。

構文 1

IF *search-condition*
THEN *expression1*
[**ELSE** *expression2*]
ENDIF

備考

互換性を保つために、この式は ENDIF または END IF で終了できます。

この式は次のように返します。

- *search-condition* が TRUE の場合、IF 式は *expression1* を返します。
- *search-condition* が FALSE で、ELSE 句が指定されている場合、IF 式は *expression2* を返します。
- *search-condition* が FALSE で *expression2* がない場合、IF 式は NULL を返します。
- *search-condition* が UNKNOWN の場合、IF 式は NULL を返します。

参照

- 「NULL 特別値」『SQL Anywhere サーバ SQL リファレンス』
- 「探索条件」『SQL Anywhere サーバ SQL リファレンス』

CASE 式

SQL の条件式です。

構文 1

CASE *expression1*
WHEN *expression2* **THEN** *expression3*, ...
[**ELSE** *expression4*]
END

```
SELECT ID,
  ( CASE name
    WHEN 'Tee Shirt' THEN 'Shirt'
    WHEN 'Sweatshirt' THEN 'Shirt'
    WHEN 'Baseball Cap' THEN 'Hat'
    ELSE 'Unknown'
```

```
  END ) as Type
FROM Product
```

構文 2

CASE
WHEN *search-condition*
THEN *expression1*, ...
[**ELSE** *expression2*]
END

備考

互換性を保つために、この式は ENDCASE または END CASE で終了できます。

CASE 式は、正規表現が使用できればどこでも使用できます。

構文 1　CASE キーワードに続く式が最初の WHEN キーワードに続く式と等しい場合、対応する THEN キーワードの後の式が返されます。それ以外の場合、ELSE キーワードがあればそれに続く式が返されます。

たとえば、下記のコードでは CASE 式が SELECT 文の 2 番目の句として使用されています。この式によって、name カラムの値が Sweatshirt であるローが Product テーブルから選択されます。

構文 2　最初の WHEN キーワードに続く search-condition が TRUE の場合、対応する THEN キーワードに続く式が返されます。それ以外の場合、ELSE 句があればそれに続く式が返されます。

省略形 CASE 式の NULLIF 関数　NULLIF 関数は、CASE 文を省略形で記述する方法の 1 つです。NULLIF の構文は、次のとおりです。

NULLIF (*expression-1*, *expression-2*)

NULLIF は 2 つの式の値を比較します。1 番目の式と 2 番目の式が等しい場合、NULLIF は NULL を返します。1 番目の式と 2 番目の式が異なる場合、NULLIF は 1 番目の式を返します。

例

次の文では、CASE 式が SELECT 文の 3 番目の句として使用され、探索条件と文字列を関連付けています。name カラムの値が **Tee Shirt** の場合は、このクエリによって **Sale** が返されます。name カラムの値が **Tee Shirt** ではなく、quantity が 50 より大きい場合は、**Big Sale** が返されます。ただし、それ以外の場合は、このクエリによって **Regular price** が返されます。

```
SELECT ID, name,
  ( CASE
    WHEN name='Tee Shirt' THEN 'Sale'
    WHEN quantity >= 50  THEN 'Big Sale'
    ELSE 'Regular price'
  END ) as Type
FROM Product
```

集合式

Ultra Light ランタイムでは提供されない集計の計算を実行します。

構文

SUM(*expression* **)**

備考

集合式は、一連のローから単一の値を計算します。

集合式は、1 つの集合関数が使用される式か、1 つ以上のオペランドがある式です。

SELECT 文に GROUP BY 句がない場合、SELECT リストの式にすべての集合式が含まれているか、または集合式がまったく含まれていない必要があります。SELECT 文に GROUP BY 句がある場合、SELECT リストの非集合式を GROUP BY リストに記載します。

例

たとえば、次のクエリは、employee テーブル内の従業員の給与合計を計算します。このクエリでは、SUM(salary) が集合式です。

```
SELECT SUM( salary )
FROM employee
```

式のサブクエリ

別の SELECT 文の内部でネストされた SELECT 文です。

構文

サブクエリは、通常のクエリのように構成されます。

備考

Ultra Light のサブクエリの参照は、次の場合にのみ使用できます。

- FROM 句内のテーブル式として。この形式のテーブル式 (「抽出テーブル」) には、SELECT リスト内の値をフェッチする抽出テーブル名とカラム名が必要です。

- EXISTS、ANY、ALL、IN の各探索条件に値を指定するため。

サブクエリの前 (左) に指定されている名前に関するサブクエリを作成できます。これは、左側への外部参照とも呼ばれます。サブクエリ内の項目は参照できません。これは、内部参照とも呼ばれます。

参照

- 「SELECT 文 [Ultra Light]」451 ページ
- 「サブクエリの使用」『SQL Anywhere サーバ SQL の使用法』
- 「Ultra Light の探索条件」281 ページ

例

次のサブクエリは、在庫数の少ない項目のすべての製品 ID をリストするのに使用します。

```
FROM SalesOrderItems
 ( SELECT ID
```

```
FROM Products
WHERE Quantity < 20 )
```

入力パラメータ

エンドユーザが準備文に値を指定できるようにするためのプレースホルダとして機能します。文は、ユーザが指定する値を使用して実行されます。

構文

{ **?** | :*name* [: *indicator-name*] }

備考

式でプレースホルダ文字 ? または名前形式を使用します。入力パラメータは、カラム名または定数を使用できる任意の場所で使用できます。

文に値が渡される具体的なメカニズムは、Ultra Light クライアントの作成に使用する API によって異なります。

名前形式の使用　入力パラメータの名前形式には特別な意味があります。一般に、常に *name* を使用して、実際の値を渡す複数のロケーションを指定します。

Embedded SQL アプリケーションの場合にのみ、*indicator-name* が、NULL インジケータが配置される変数を指定します。名前形式と他のコンポーネントを同時に使用した場合、*indicator-name* は無視されます。

データ型の抽出　入力パラメータのデータ型は、文が次のいずれかのパターンから準備されるときに抽出されます。

○**CAST** (**?** **AS** *type*)

ここで、*type* は CHAR(32) などのデータベースの型指定です。

○バイナリ演算子の 1 つのオペランドが入力パラメータである。型が抽出され、オペランドの型になります。

Ultra Light で型を抽出できなかった場合は、エラーが発生します。次に例を示します。

○**-?**：オペランドが単項である。

○**? + ?**：両方共入力パラメータである。

参照

- 「ホスト変数」『Ultra Light C/C++ プログラミング』
- 「準備文」『SQL Anywhere サーバ プログラミング』
- Ultra Light C/C++：「INSERT、UPDATE、DELETE を使用したデータ修正」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET：「INSERT、UPDATE、DELETE を使用したデータ修正」『Ultra Light .NET プログラミング』

例

次の Embedded SQL 文には 2 つの入力パラメータがあります。

```
INSERT INTO MyTable VALUES ( :v1, :v2, :v1)
```

v1 の最初のインスタンスが、文内の v2 と v1 の両方のロケーションにその値を渡します。

Ultra Light の探索条件

探索条件は、WHERE 句、HAVING 句、ジョインの ON フレーズ、または IF 式の条件です。探索条件は、「述部」とも言います。

構文

search-condition:
expression comparison-operator expression
| *expression* **IS** [**NOT**] **NULL**
| *expression* [**NOT**] **BETWEEN** *expression* **AND** *expression*
| *expression* [**NOT**] **IN** (*expression*, ...)
| *expression* [**NOT**] **IN** (*subquery*)
| *expression* [**NOT**] { **ANY** | **ALL** } (*subquery*)
| *expression* [**NOT**] **EXISTS** (*subquery*)
| *expression* [**NOT**] **LIKE** (*pattern*)
| **NOT** *search-condition*
| *search-condition* **AND** *search-condition*
| *search-condition* **OR** *search-condition*
| (*search-condition* **IS** [**NOT**] {**TRUE** | **FALSE** | **UNKNOWN** }

comparison-operator :
=
| **>**
| **<**
| **>=**
| **<=**
| **<>**
| **!=**
| **!<**
| **!>**

パラメータ

Ultra Light では、次のタイプの探索条件がサポートされています。

- ALL 条件
- ANY 条件
- BETWEEN 条件
- EXISTS 条件
- IN 条件
- LIKE 探索条件

備考

Ultra Light では、次の箇所で探索条件を指定します。

●WHERE 句

●HAVING 句

●ON フレーズ

●SQL クエリ

探索条件は、SELECT 文の FROM 句で使用してテーブル内のローのサブセットを選択するか、IF や CASE などの式で使用して特定の値を選択できます。Ultra Light では、すべての条件が TRUE、FALSE、または UNKNOWN のいずれかに評価されます。これらを組み合わせた場合、「3 値的論理」と呼ばれます。比較される値のいずれかが NULL の場合、比較結果は UNKNOWN になります。探索条件は、条件の結果が TRUE の場合にのみ満たされます

参照

- 「Ultra Light の探索条件」281 ページ
- 「クエリ」『SQL Anywhere サーバ SQL の使用法』
- 「WHERE 句での文字列のパターン一致」『SQL Anywhere サーバ SQL の使用法』
- 「比較演算子」282 ページ
- 「3 値的論理」『SQL Anywhere サーバ SQL リファレンス』
- 「式のサブクエリ」279 ページ

比較演算子

探索条件で、複数の式を比較できる演算子です。

構文

expression operator expression

パラメータ

演算子	意味
=	等価
>	より大きい
<	より小さい
>=	大きいかまたは等価
<=	小さいかまたは等価
!=	等価ではない
<>	等価ではない
!>	以下

演算子	意味
!<	以上

備考

日付の比較　日付を比較するときには、< はより古いことを意味し、> はより新しいことを意味します。

LONG VARCHAR 値や LONG BINARY 値の比較　Ultra Light は、LONG VARCHAR 値や LONG BINARY 値を使用した比較をサポートしていません。

大文字と小文字の区別　Ultra Light では、比較処理は、該当のデータベースの文字の区別に合わせて実行されます。デフォルトでは、Ultra Light データベースは大文字と小文字を区別しないで作成されます。

NOT 演算子　NOT 演算子は式を否定します。

参照

- 「論理演算子」283 ページ
- 「Ultra Light の探索条件」281 ページ

例

次の 2 つのクエリはどちらも、単価が $10 以下の T シャツ (Tee Shirt) と野球帽 (BaseBall Cap) をすべて検索します。ただし、否定の論理演算子 (NOT) と否定の比較演算子 (!>) では、位置が異なることに注意してください。

```
SELECT ID, Name, Quantity
FROM Products
WHERE (name = 'Tee Shirt' OR name = 'BaseBall Cap')
AND NOT UnitPrice > 10

SELECT ID, Name, Quantity
FROM Products
WHERE (name = 'Tee Shirt' OR name = 'BaseBall Cap')
AND UnitPrice !> 10
```

論理演算子

次のいずれかを行います。

- 条件の比較 (AND、OR、NOT)
- 式 (IS) の真理値または NULL 値のテスト

構文 1

condition1 logical-operator condition2

構文 2

NOT *condition*

構文 3

expression **IS** [**NOT**] { *truth-value* | **NULL** }

備考

探索条件は、SELECT 文の FROM 句で使用してテーブル内のローのサブセットを選択するか、IF や CASE などの式で使用して特定の値を選択できます。Ultra Light では、すべての条件が TRUE、FALSE、または UNKNOWN のいずれかに評価されます。これらを組み合わせた場合、「3 値的論理」と呼ばれます。比較される値のいずれかが NULL の場合、比較結果は UNKNOWN になります。探索条件は、条件の結果が TRUE の場合にのみ満たされます

AND　両方の条件が TRUE の場合、結合した条件は TRUE になります。条件のいずれかが FALSE の場合は FALSE、それ以外の場合は UNKNOWN になります。

condition1 **AND** *condition2*

OR　条件のいずれかが TRUE の場合、結合した条件は TRUE になります。両方の条件が FALSE の場合は FALSE、それ以外の場合は UNKNOWN になります。

NOT　*condition* が FALSE の場合、NOT 条件は TRUE です。*condition* が TRUE の場合は FALSE、*condition* が UNKNOWN の場合は UNKNOWN になります。

IS　*expression* が指定の *truth-value* (TRUE、FALSE、UNKNOWN のいずれか) と評価されれば条件は TRUE になります。それ以外の場合、値は FALSE です。

参照

- 「3 値的論理」『SQL Anywhere サーバ SQL リファレンス』
- 「比較演算子」282 ページ
- 「Ultra Light の探索条件」281 ページ

例

IS NULL 条件は、カラムに NULL 値が含まれる場合に満たされます。IS NOT NULL 演算子を使用すると、この条件は、カラムに NULL でない値が含まれる場合に満たされます。たとえば、WHERE paid_date IS NULL は IS NULL 条件です。

ALL 探索条件

ALL 条件を比較演算子と組み合せて使用すると、1 つの値を、サブクエリが生成するデータ値と比較できます。

構文

expression compare　[**NOT**] **ALL** (*subquery*)

パラメータ

compare:
= | > | < | >= | <= | <> | != | !< | !>

備考

Ultra Light は指定された比較演算子を使用して、テスト値を結果セットのデータ値と比較します。すべての比較の結果が TRUE になる場合、ALL テストは TRUE を返します。

参照

- 「サブクエリと ALL テスト」『SQL Anywhere サーバ SQL の使用法』
- 「比較演算子」282 ページ

例

注文番号 2001 のすべての製品が出荷された後に受けた注文の注文 ID と顧客 ID を検索します。

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > ALL (
 SELECT ShipDate
 FROM SalesOrderItems
 WHERE ID=2001)
```

ANY 探索条件

ANY 条件を比較演算子と組み合せて使用すると、1 つの値を、サブクエリが生成するデータ値のカラムと比較できます。

構文 1

expression compare [**NOT**] **ANY** (*subquery*)

構文 2

expression = **ANY** (*subquery*)

パラメータ

compare:
= | > | < | >= | <= | <> | != | !< | !>

備考

Ultra Light は指定された比較演算子を使用して、テスト値をカラムのデータ値と比較します。いずれかの比較の結果が TRUE になる場合、ANY テストは TRUE を返します。

構文 1　*expression* がサブクエリ結果のいずれかの値と等しい場合は TRUE、expression が NULL ではなく、サブクエリから返されたいずれの値とも等しくない場合は FALSE です。*expression* が NULL 値の場合、サブクエリ結果にローがあれば ANY 条件は UNKNOWN です。サブクエリ結果にローがなければ、条件は必ず FALSE になります。

参照

- 「サブクエリと ANY テスト」『SQL Anywhere サーバ SQL の使用法』
- 「比較演算子」282 ページ

例

注文番号 2005 の最初の製品が出荷された後に受けた注文の注文 ID と顧客 ID を検索します。

```
SELECT ID, CustomerID
FROM SalesOrders
WHERE OrderDate > ANY (
 SELECT ShipDate
 FROM SalesOrderItems
 WHERE ID=2005)
```

BETWEEN 探索条件

包括的範囲を指定します。包括的範囲には、その範囲の間の値だけではなく、上限値と下限値も含まれます。

構文

expression [**NOT**] **BETWEEN** *start-expression* **AND** *end-expression*

備考

BETWEEN 条件は TRUE、FALSE、または UNKNOWN として評価できます。NOT キーワードがない場合、*expression* が *start-expression* と *end-expression* の間にあれば、条件は TRUE と評価されます。NOT キーワードを使用すると条件の意味が逆になりますが、UNKNOWN は変わりません。

BETWEEN 条件は、次の 2 つの不等式の組み合わせと等価です。

[**NOT**] (*expression* >= *start-expression*
AND *expression* <= *end-expression*)

例

$10 未満か、$15 を超える製品をすべてリストします。

```
SELECT Name, UnitPrice
FROM Products
WHERE UnitPrice NOT BETWEEN 10 AND 15
```

EXISTS 探索条件

サブクエリがクエリ結果のローを生成するかどうかを調べます。

構文

[**NOT**] **EXISTS (** *subquery* **)**

備考

EXISTS 条件は、サブクエリ結果にローが少なくとも 1 つあれば TRUE で、ローがなければ FALSE です。EXISTS 条件は、UNKNOWN にはなりません。

EXISTS 条件の論理は、NOT EXISTS というフォームで否定できます。この場合、テストはサブクエリがローを返さない場合に TRUE を、ローを返す場合に FALSE を返します。

例

2001 年 7 月 13 日より後に注文した顧客をリストします。

```
SELECT GivenName, Surname
FROM Customers
WHERE EXISTS (
 SELECT *
 FROM SalesOrders
 WHERE (OrderDate > '2001-07-13') AND
      (Customers.ID = SalesOrders.CustomerID))
```

IN 探索条件

メインクエリの値からサブクエリの別の値を探索することでメンバーシップをチェックします。

構文

expression [**NOT**] **IN**
{ (*subquery*) | (*value-expr*, ...) }

パラメータ

value-expr は、単一値をとる式です。これには、文字列、数字、日付、または他の任意の SQL データ型などがあります。

備考

NOT キーワードがない場合、IN 条件は次の規則に従って評価されます。

- *expression* が NULL でなく、少なくとも 1 つの値と等しい場合、TRUE です。
- *expression* が NULL で、値リストが空でない場合、または少なくとも 1 つの値が NULL で、*expression* が他の値のいずれとも等しくない場合、UNKNOWN です。
- *expression* が NULL で、*subquery* が値を返さない場合、または *expression* が NULL でなく、いずれの値も NULL でなく、*expression* がいずれの値とも等しくない場合、FALSE です。

IN 条件の論理は、NOT IN という形式で否定できます。

探索条件 *expression* **IN** (*values*) は、探索条件 *expression* = **ANY** (*values*) と同じです。探索条件 *expression* **NOT IN** (*values*) は、探索条件 *expression* <> **ALL** (*values*) と同じです。

例

カナダのオンタリオ州、マニトバ州、ケベック州に在住する顧客の会社名と状態を選択します。

```
SELECT CompanyName , Province
FROM Customers
WHERE State IN( 'ON', 'MB', 'PQ')
```

LIKE 探索条件

構文

LIKE 探索条件の構文は次のとおりです。

expression [**NOT**] **LIKE** *pattern*

パラメータ

- **expression**　検索される文字列。
- **pattern**　*expression* を検索するためのパターン。

備考

LIKE 探索条件は、*expression* を *pattern* に一致させようとし、TRUE、FALSE、または UNKNOWN と評価されます。*expression* が *pattern* に一致すれば、探索条件は TRUE と評価されます (NOT が指定されない場合)。*expression* または *pattern* が NULL 値の場合、探索条件は UNKNOWN と評価されます。

NOT キーワードを使用すると探索条件の意味が逆になりますが、UNKNOWN は変わりません。

expression と *pattern* は、CHAR 文字列として解釈されます。*pattern* には、次ののサポートされているワイルドカードをいくつでも含めることができます。

ワイルドカード	一致するもの
_ (アンダースコア)	任意の 1 文字。たとえば、a_ は ab と ac に一致しますが、a には一致しません。
% (パーセント記号)	0 個以上の文字からなる任意の文字列。たとえば、bl% は bl と bla に一致します。
[]	指定範囲内、または一連の指定文字の任意の 1 文字。たとえば、T[oi]m は Tom または Tim に一致します。
[^]	指定範囲、または一連の指定文字に**含まれない**任意の 1 文字。たとえば、M[^c] は Mb と Md に一致しますが、Mc には一致しません。

例

次の探索条件では、文字 a で始まり、末尾から 2 つ目の文字が b のカラム名のローがすべて TRUE になります。

```
SELECT * FROM table-name WHERE column-name LIKE 'a%b_'
```

LIKE 探索条件を使用するさまざまな方法

検索対象	例	その他の情報
一連の文字	LIKE 'sm[iy]th'	検索対象の一連の文字は、角カッコ内にリストして指定します。この例では、探索条件は **smith** と **smyth** に一致します。

 Copyright © 2013, SAP AG or an SAP affiliate company.

検索対象	例	その他の情報
範囲内の文字	LIKE '[a-r]ough'	検索対象の文字範囲は、角カッコ内に範囲を書いて指定します。範囲の始めと終わりの間にハイフンを書きます。この例では、探索条件は bough と rough に一致しますが、tough には一致しません。 文字列の範囲 [a-z] は「a 以上 z 以下」と解釈され、データベースの照合では、以上と以下の演算子が実行されます。 範囲は、まず小さい方の値、次に大きい方の値を指定してください。たとえば、[z-a] は何も一致しません。これは、[z-a] の範囲にいずれの文字も一致しないためです。
文字範囲と一連の文字の結合	... LIKE '[a-rt]ough'	角カッコ内で、文字範囲と一連の文字を結合できます。この例では、... LIKE '[a-rt]ough' は bough、rough、tough に一致します。 パターン [a-rt] は、a ～ r の範囲に含まれる文字または t と一致する 1 文字と解釈されます。
範囲外の 1 文字	... LIKE '[^a-r]ough'	脱字記号 (^) は、検索から除外する文字範囲を指定します。この例では、LIKE '[^a-r]ough' は文字列 tough に一致しますが、文字列 rough や bough には一致しません。 脱字記号は、この記号以外のカッコ内の内容を否定します。たとえば、角カッコ [^a-rt] は、a ～ r の範囲に含まれない文字かつ t ではない 1 文字と解釈されます。

検索対象	例	その他の情報
後続ブランクのある探索パターン	'90 '、'90[ ]'、'90_'	探索パターンに後続ブランクが含まれる場合、データベースサーバではブランクが含まれる値に対してのみパターンが一致します。文字列へのブランクの埋め込みは行われません。たとえば、検索されている値が幅 3 文字以上の CHAR 型または VARCHAR 型カラムにあっても、'90 '、'90[ ]'、'90_' のパターンは、式 '90 ' には一致しますが、式 '90' には一致しません。

文字範囲と一連の文字の特殊なケース

角カッコ内の任意の 1 文字は、その文字を指しています。たとえば、[a] は、文字 a に一致します。[^] は脱字記号、[%] はパーセント記号 (パーセント記号は、このコンテキストではワイルドカードになりません)、[_] はアンダースコア文字に一致します。また、[[] は文字 [に一致します。

- パターン [a-] は、文字 a または - に一致します。
- パターン [] は、一致する文字がないのでローを返すことはありません。
- パターン [または [abp-q は、閉じカッコが欠けているため構文エラーを返します。
- 角カッコ内ではワイルドカードは使用できません。パターン [a%b] は a、%、または b のいずれかを検出します。
- 脱字記号を使用しても、カッコ内の先頭になければ範囲を否定できません。パターン [a^b] は a、^、または b のいずれかを検出します。

大文字と小文字の区別と、比較の実行

データベース照合が大文字と小文字を区別する場合は、探索条件でも大文字と小文字を区別します。大文字と小文字を区別する照合で大文字と小文字を区別しない検索を実行するには、大文字と小文字を両方指定します。たとえば、次の探索条件では、文字列 Bough、rough、TOUGH が true と評価されます。

```
LIKE '[a-zA-Z][oO][uU][gG][hH]'
```

Ultra Light の演算子

演算子は値の計算に使用します。計算された値は、さらに上位の式でオペランドとして使用できます。

Ultra Light SQL では、次のタイプの演算子がサポートされています。

- 比較演算子は、1 つ (単項) または 2 つ (バイナリ) の比較オペランドを使用して評価し、結果を返します。比較の結果は、通常の 3 つの論理値、true、false、または unknown になります。
- 算術演算子は、浮動小数点、小数、整数の数値を評価し、結果セットを返します。
- 文字列演算子は、2 つの文字列を連結します。たとえば、"my" + "string" は文字列 "mystring" を返します。
- ビット処理演算子は、整数の内部表現内の特定のビットをオンまたはオフにします。
- 論理演算子は、探索条件を評価します。論理的評価の結果は、通常の 3 つの論理値、true、false、または unknown になります。

一般的な演算子の優先度が適用されます。

参照

- 「演算子の優先度」293 ページ
- 「比較演算子」282 ページ
- 「算術演算子」291 ページ
- 「文字列演算子」292 ページ
- 「ビット処理演算子」292 ページ
- 「論理演算子」283 ページ

算術演算子

算術演算子を使用すると、計算を実行できます。

expression + expression　加算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

expression - expression　減算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

-expression　反転。式が NULL 値の場合、結果は NULL 値になります。

expression * expression　乗算。いずれかの式が NULL 値の場合、結果は NULL 値になります。

expression / expression　除算。いずれかの式が NULL の場合、または 2 番目の式が 0 の場合、結果は NULL になります。

expression % expression　剰余による、2 つの整数での除算の余り (整数) の算出。たとえば、21 を 11 で割ると商は 1、余りは 10 なので、21 % 11 = 10 になります。いずれかの式が NULL 値の場合、結果は NULL 値になります。

参照

- 「算術演算」『SQL Anywhere サーバ SQL の使用法』

文字列演算子

文字列演算子を使用すると、文字列を結合できます。ただし、LONGVARCHAR データ型と LONGBINARY データ型ではできません。

expression || expression　文字列連結 (2 つのパイプ記号)。いずれかの文字列が NULL 値の場合、連結には空文字列として扱われます。

expression + expression　代替の文字列連結。+ 連結演算子を使用する場合は、暗黙的データ変換を行わないで、必ずオペランドを文字データ型に明示設定してください。

たとえば、次のクエリは整数値 **579** を返します。

```
SELECT 123 + 456
```

これに対し、次のクエリは文字列 **123456** を返します。

```
SELECT '123' + '456'
```

CAST または CONVERT 関数を使用すると、データ型を明示的に変換できます。

ビット処理演算子

ビット処理演算子は、2 つの式間でビット操作を実行します。Ultra Light では、次の演算子を整数データ型に対して使用できます。

演算子	説明
&	ビット処理 AND
|	ビット処理 OR
^	ビット処理排他的 OR
~	ビット処理 NOT

ビット処理演算子 &、|、~ は、論理演算子 AND、OR、NOT で代用することはできません。ビット処理演算子は、値のビット表現を使用して整数値に対して作用します。

例

次の文は指定するビットが設定されているローを選択します。

```
SELECT *
FROM tableA
WHERE (options & 0x0101) <> 0
```

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

演算子の優先度

式における演算子の優先度は次のとおりです。カッコ内の式が最初に評価され、続いて乗算と除算、最後に加算と減算が評価されます。その後、文字列の連結が行われます。リストの最上部にある演算子から順に評価されます。

> **ヒント**
> 演算子の優先度に依存しないで、演算の順序を明示的に指定してください。式で複数の演算子を使用する場合は、カッコを使用して演算の順序を明確にしてください。

1. 名前、関数、定数、IF 式、CASE 式
2. ()
3. 単項演算子 (1 つのオペランドを必要とする演算子)： +, -
4. ~
5. **&**, | , ^
6. *****, /, **%**
7. +, **-**
8. ||
9. 比較：>、<、<>、!=、<=、>=、[NOT] BETWEEN、[NOT] IN、[NOT] LIKE
10. 比較：IS [NOT] TRUE、FALSE、UNKNOWN
11. NOT
12. AND
13. OR

Ultra Light の変数

Ultra Light アプリケーションでは、SQL 変数 (グローバル変数を含む) は使用できません。

Ultra Light SQL データ型

Ultra Light は、SQL Anywhere で使用できるデータ型のサブセットをサポートします。

文字データ型

文字データ型は、文字、数字、記号などの文字列を格納します。

Ultra Light は、シングルバイトまたはマルチバイトの文字セットに格納される CHAR、VARCHAR、LONG VARCHAR データ型をサポートしており、多くの場合、データベースに格納されるプライマリ言語に最も対応するデータ型が選択されます。

格納

固定文字型 (VARCHAR など) はローに埋め込まれ、LONG 文字型 (LONG VARCHAR など) は個別に格納されます。

大きな固定型のカラム多数で構成されるテーブルを作成するときは、ページサイズを考慮します。1 行全体が 1 ページに収まる必要があります。また、固定文字カラム型は 1 行で格納します。たとえば、ページサイズを 1000 にして作成されたデータベースでは、1000 よりも大きな文字値はページに収まらないため保持できません。

参照

● 「CREATE TABLE 文 [Ultra Light]」432 ページ

CHAR データ型

CHAR データ型は、32767 バイトまでの文字データを格納します。

構文

CHAR [(*max-length*)]

備考

CHAR はドメインであり、VARCHAR として実装されます。

参照

● 「VARCHAR データ型」295 ページ

LONG VARCHAR データ型

LONG VARCHAR データ型は、任意の長さの文字データを格納します。

構文

LONG VARCHAR

備考

最大バイトサイズは 2 GB から 1 バイトを差し引いた値です (2^{31} - 1)。

LONG VARCHAR 型に対してインデックスを作成することはできません。

LONG VARCHAR 型は LENGTH 関数と CAST 関数でのみ使用できます。

参照

- 「CHAR データ型」294 ページ
- 「VARCHAR データ型」295 ページ
- 「LENGTH 関数 [文字列]」361 ページ
- 「CAST 関数 [データ型変換]」328 ページ

VARCHAR データ型

VARCHAR データ型は、32767 バイトまでの文字データを格納します。

構文

VARCHAR [**(** *max-length* **)**]

パラメータ

- **max-length**　文字列の最大長。このデフォルト値は 1 です。

  Ultra Light データベースはバイト長のセマンティックのみをサポートしています。英語以外の文字は最大 3 バイトの記憶領域が必要になることがあります。

  Ultra Light Java Edition データベースは文字長のセマンティックのみをサポートしています。*max-length* は最大 32767 文字です。

備考

マルチバイト文字は VARCHAR として格納できますが、宣言される長さは文字数ではなくバイト数です。

Ultra Light はデータを可能な限りコンパクトにします。VARCHAR 値が *max-length* で指定されたバイト数を必要としない場合は、値を格納するために必要なバイト数のみが使用されます。

> **警告**
> *max-length* がページサイズを超えている VARCHAR カラムを使用してテーブルを作成することはできますが、そのページサイズを超える長さの値を挿入するとエラーが発生します。

式を評価するときのテンポラリ文字値の最大長は 2048 バイトです。

参照

- 「CHAR データ型」294 ページ
- 「LONG VARCHAR データ型」294 ページ

数値データ型

数値データ型は数値データを格納します。

NUMERIC データ型、DECIMAL データ型、さまざまな INTEGER データ型は「真数値」データ型と呼ぶこともあります。これと対照的なのが、「概数値」データ型である FLOAT、DOUBLE、REAL です。

真数値データ型は、精度と小数点以下の桁数の値を指定できる型です。一方、概数値データ型は事前に定義された方法で格納されます。**真数値データだけが、算術演算後に指定した最少有効桁数に対して正確性が保証されます。**

1 より小さいデータ型の長さと精度は使用できません。

互換性

NUMERIC と DECIMAL のデータ型にデフォルトの精度と位取りの設定を使用する場合は注意が必要です。他のデータベースソリューションでは設定が異なる場合があるためです。デフォルトの精度は 30、デフォルトの位取りは 6 です。

FLOAT (*p*) データ型は、*p* の値によって、REAL か DOUBLE のどちらかの同義語になります。

データベースオプションの設定によるデフォルトの変更については、「Ultra Light precision 作成パラメータ」153 ページおよび「Ultra Light scale 作成パラメータ」154 ページを参照してください。

BIGINT データ型

BIGINT データ型は、8 バイトの記憶領域を必要とする整数である BIGINT を格納します。

構文

[UNSIGNED] BIGINT

備考

BIGINT データ型は真数値データ型です。精度は算術演算の後で保存されます。

BIGINT 値には 8 バイトの記憶領域が必要です。

BIGINT 値の範囲は、-2^{63} ～ $2^{63} - 1$、または -9223372036854775808 ～ 9223372036854775807 です。

UNSIGNED BIGINT 値の範囲は、0 ～ $2^{64} - 1$、または 0 ～ 18446744073709551615 です。

デフォルトでは、このデータ型は符号付きです。

文字列を BIGINT に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。変換しようとしたデータ型の有効な範囲外の値である場合、文字列に不正な文字が含まれる場合、または文字列を整数値として復号化できない場合にエラーが返されます。

参照

- 「BIT データ型」297 ページ
- 「INTEGER データ型」300 ページ
- 「SMALLINT データ型」301 ページ
- 「TINYINT データ型」302 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ

BIT データ型

BIT データ型は、1 つのビット (0 または 1) を格納します。

構文

BIT

備考

BIT は、0 または 1 の値を格納できる整数型です。

デフォルトでは、BIT データ型には NULL を入力できません。

BIT 値には 1 ビットの記憶領域が必要です。

文字列を BIT に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。値が 0 または 1 でない場合はエラーが返されます。

参照

- 「BIGINT データ型」296 ページ
- 「INTEGER データ型」300 ページ
- 「SMALLINT データ型」301 ページ
- 「TINYINT データ型」302 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ

DECIMAL データ型

DECIMAL データ型は、総桁数の *precision* と小数点以下の桁数の *scale* を持つ 10 進数です。

構文

DECIMAL [**(** *precision* [, *scale*] **)**]

パラメータ

- **precision**　式の桁数を指定する整数式。1 ～ 127 の範囲。デフォルト設定値は 30 です。

- **scale**　小数点以下の桁数を指定する整数式。0 ～ 127 の範囲。scale 値は precision 値以下にする必要があります。デフォルト設定値は 6 です。

  デフォルトは、適切な作成パラメータを設定することによって変更できます。

備考

DECIMAL データ型は真数値データ型です。精度は、算術演算の後、最小の有効桁数まで保存されます。

10 進数を格納するために必要なバイト数は、次のように計算できます。

```
2 + INT(((precision - scale) + 1) / 2) + INT((scale + 1) / 2);
```

INT 関数は引数の整数部分です。記憶領域の計算の基準は、カラムの中で使われている最大精度と小数点以下の桁数ではなく、格納されている値を使います。

使用する精度が 20 以下で小数点以下の桁数が 0 の場合、代わりに整数データ型のいずれか (BIGINT、INTEGER、SMALLINT、または TINYINT) を使用できることがあります。整数値では、有効桁数が同じくらいの NUMERIC 値や DECIMAL 値よりも少ない記憶領域が使用されます。整数値の操作 (フェッチ、挿入など) や算術演算子は、通常、NUMERIC 値や DECIMAL 値の操作よりも速く実行されます。

注意
DECIMAL データ型のカラムまたは変数を作成するとき、これらのカラムまたは変数の精度または位取りがデータベースの精度および位取りの設定を超過する場合は、値はデータベースの設定に合わせてトランケートされます。このため、DECIMAL として定義されているカラムまたは変数の値がトランケートされる場合は、精度と位取りがデータベースのオプション設定を超過していないかどうかを確認します。

参照

- 「FLOAT データ型」299 ページ
- 「REAL データ型」301 ページ
- 「DOUBLE データ型」298 ページ
- 「NUMERIC データ型」301 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ
- 「Ultra Light precision 作成パラメータ」153 ページ
- 「Ultra Light scale 作成パラメータ」154 ページ

DOUBLE データ型

DOUBLE データ型は、倍精度の浮動小数点数を格納します。

構文

DOUBLE

備考

DOUBLE データ型は概数値データ型であり、算術演算後に丸め誤差が出ます。概数値という DOUBLE 値の性質により、通常、DOUBLE 値を比較するときは等号を使用するクエリを避けてください。

DOUBLE 値には 8 バイトの記憶領域が必要です。

値の範囲は -1.79769313486231e+308 ～ 1.79769313486231e+308 であり、0 に最も近い最小の数値は 2.22507385850721e-308 です。DOUBLE として保持される値は、厳密には 15 有効桁数ですが、16 桁以上になると丸め誤差が出ます。

参照

- 「FLOAT データ型」299 ページ
- 「REAL データ型」301 ページ
- 「DECIMAL データ型」297 ページ
- 「NUMERIC データ型」301 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ
- 「数値セットの変換」『SQL Anywhere サーバ SQL リファレンス』

FLOAT データ型

FLOAT データ型は、単精度または倍精度の浮動小数点数を格納します。

構文

FLOAT [(*precision*)]

パラメータ

- **precision**　仮数部 (常用対数の小数部) の桁数を指定する整数式。たとえば、5.63428 という数値の仮数は 0.63428 です。IEEE 標準 754 による浮動小数点精度は、次のとおりです。

提供されている精度値	10 進数精度	等価の SQL データ型	記憶サイズ
1-24	7 桁	REAL	4 バイト
25-53	15 桁	DOUBLE	8 バイト

備考

FLOAT (*precision*) データ型を使ってカラムを作成すると、すべてのプラットフォームのカラムには、少なくとも指定した最小精度で値が格納されることが保証されます。REAL と DOUBLE の場合、プラットフォームに依存しない最小精度は保証されません。

precision を指定しない場合、FLOAT データ型は単精度の浮動小数点数です。これは REAL データ型と等価で、4 バイトの記憶領域を必要とします。

precision を指定する場合、FLOAT データ型は指定した精度の値によって、単精度か倍精度のいずれかになります。REAL になるか DOUBLE になるかは、プラットフォームによって異なりま

す。単精度の FLOAT 値は、4 バイトの記憶領域を必要とし、倍精度の FLOAT 値は、8 バイトを必要とします。

FLOAT データ型は概数値データ型です。算術演算後の丸め誤差がでます。概数値という FLOAT 値の性質により、FLOAT 値を比較するときは等号を使用するクエリを避けてください。

参照

- 「DOUBLE データ型」298 ページ
- 「REAL データ型」301 ページ
- 「DECIMAL データ型」297 ページ
- 「NUMERIC データ型」301 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ

INTEGER データ型

INTEGER データ型は、4 バイトの記憶領域を必要とする整数を格納します。

構文

[UNSIGNED] INTEGER

備考

INTEGER データ型は真数値データ型で、精度は算術演算の後で保存されます。

UNSIGNED を指定した場合、整数に負の数を割り当てることはできません。デフォルトでは、このデータ型は符号付きです。

INTEGER 値の範囲は、-2^{31} ～ $2^{31} - 1$、または -2147483648 ～ 2147483647 です。

UNSIGNED INTEGER 値の範囲は、0 ～ $2^{32} - 1$、または 0 ～ 4294967295 です。

文字列を INTEGER に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。変換しようとしたデータ型の有効範囲外の値である場合、文字列に不正な文字が含まれている場合、または文字列を整数値として復号化できない場合は、エラーが戻ります。

参照

- 「BIGINT データ型」296 ページ
- 「BIT データ型」297 ページ
- 「SMALLINT データ型」301 ページ
- 「TINYINT データ型」302 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ

NUMERIC データ型

NUMERIC データ型は、総桁数の *precision* と小数点以下の桁数の *scale* を持つ 10 進数を格納します。

構文

NUMERIC [**(** *precision* [, *scale*] **)**]

備考

NUMERIC はドメインであり、DECIMAL として実装されます。

参照

- 「DECIMAL データ型」297 ページ

REAL データ型

REAL データ型は、4 バイトで格納される単精度の浮動小数点数を格納します。

構文

REAL

備考

REAL データ型は概数値データ型であり、算術演算後に丸め誤差が出ます。概数値という REAL 値の性質により、通常、REAL 値を比較するときは等号を使用するクエリを避けてください。

REAL 値には 4 バイトの記憶領域が必要です。

値の範囲は -3.402823e+38 ～ 3.402823e+38 であり、0 に最も近い最小の数値は 1.175494351e-38 です。REAL として保持される値は、厳密には 7 有効桁数ですが、6 桁を越えると丸め誤差が出ます。

参照

- 「DOUBLE データ型」298 ページ
- 「FLOAT データ型」299 ページ
- 「DECIMAL データ型」297 ページ
- 「NUMERIC データ型」301 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ

SMALLINT データ型

SMALLINT データ型は、2 バイトの記憶領域を必要とする整数を格納します。

構文

[**UNSIGNED**] **SMALLINT**

備考

SMALLINT データ型は真数値データ型です。精度は算術演算の後で保存されます。2 バイトの記憶領域を必要とします。

SMALLINT 値の範囲は、-2^{15} ～ 2^{15} - 1、または -32768 ～ 32767 です。

UNSIGNED SMALLINT 値の範囲は、0 ～ 2^{16} - 1、または 0 ～ 65535 です。

文字列を SMALLINT に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。変換しようとしたデータ型の有効範囲外の値である場合、文字列に不正な文字が含まれている場合、または文字列を整数値として復号化できない場合は、エラーが戻ります。

参照

- 「BIGINT データ型」296 ページ
- 「BIT データ型」297 ページ
- 「INTEGER データ型」300 ページ
- 「TINYINT データ型」302 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ

TINYINT データ型

TINYINT データ型は、1 バイトの記憶領域を必要とする符号なし整数を格納します。

構文

TINYINT

備考

TINYINT データ型は真数値データ型です。精度は算術演算の後で保存されます。

TINYINT 値の範囲は、0 ～ 2^{8} - 1、または 0 ～ 255 です。

Embedded SQL では、TINYINT カラムを CHAR と定義された変数にフェッチしないでください。そうした場合、カラムの値を文字列に変換し、最初のバイトをプログラムの変数に割り当てようとすることになるからです。TINYINT カラムは 2 バイトまたは 4 バイト整数にフェッチしてください。TINYINT 値を C で書かれたアプリケーションからデータベースに送るには、C 変数の型に INTEGER を指定してください。

文字列を TINYINT に変換すると、前後のスペースは削除されます。先行文字が + の場合は無視されます。先行文字が - の場合は、残りの数字は負の数として解釈されます。先行の 0 文字はスキップされ、残りの文字は整数値に変換されます。変換しようとしたデータ型の有効範囲外の値である場合、文字列に不正な文字が含まれている場合、または文字列を整数値として復号化できない場合は、エラーが戻ります。

参照

- 「BIGINT データ型」296 ページ
- 「BIT データ型」297 ページ
- 「INTEGER データ型」300 ページ
- 「SMALLINT データ型」301 ページ
- 「Ultra Light 数値関数」318 ページ
- 「Ultra Light 集合関数」313 ページ

日付と時刻データ型

以下に、日付がどのように処理されるかを簡単にまとめます。

- 算出した値が別の世紀にかかるかどうかにかかわらず、日付に関する有効な算術演算や論理演算に対して常に正しい値を返します。
- 日付の内部記憶領域には、年の値の世紀にあたる部分が常に明示的に含まれています。
- 日付の値は、いつでも 4 桁の西暦で出力できます。

DATE データ型

DATE データ型は、年、月、日などの暦日を格納します。

構文

DATE

備考

アプリケーションが文字列として DATE 値を取得するときの形式は、date_format 作成パラメータによって決まります。たとえば、2010 年 7 月 19 日を表す DATE 値は、date_format 作成パラメータに従い、アプリケーションに 2010/07/19 または Jul 19, 2010 として返されます。

DATE 値には 4 バイトの記憶領域が必要です。

参照

- 「CURRENT TIME 特別値」272 ページ
- 「CURRENT TIMESTAMP 特別値」273 ページ
- 「Ultra Light 日付と時刻関数」314 ページ
- 「日付形式」『SQL Anywhere サーバ SQL リファレンス』
- 「DATE 関数 [日付と時刻]」340 ページ
- 「Ultra Light date_format 作成パラメータ」142 ページ
- 「Ultra Light Java Edition の date_format オプション」246 ページ
- 「Ultra Light date_order 作成パラメータ」145 ページ
- 「Ultra Light Java Edition の date_order オプション」247 ページ
- 「DATETIME データ型」304 ページ
- 「DATETIME 関数 [日付と時刻]」345 ページ
- 「ISDATE 関数 [データ型変換]」359 ページ
- 「Ultra Light nearest_century 作成パラメータ」149 ページ
- 「Ultra Light Java Edition の nearest_century オプション」247 ページ
- 「NOW 関数 [日付と時刻]」375 ページ
- 「TIME データ型」304 ページ
- 「TIMESTAMP データ型」305 ページ
- 「CURRENT TIMESTAMP 特別値」273 ページ
- 「TIMESTAMP WITH TIME ZONE データ型」306 ページ

DATETIME データ型

DATETIME は、日付と時刻の情報を格納します。

構文

DATETIME

備考

DATETIME はドメインであり、TIMESTAMP として実装されます。

参照

- 「TIMESTAMP データ型」305 ページ

TIME データ型

TIME データ型は、時、分、秒、秒以下で構成される時刻を格納します。

構文

TIME

備考

アプリケーションが文字列として TIME 値を取得するときの形式は、time_format 作成パラメータによって決まります。たとえば、TIME 値 23:59:59.999999 は、time_format 作成パラメータに従い、23:59:59、23:59:59.999、または 23:59:59.999999 としてアプリケーションに返されます。

TIME 値には 8 バイトの記憶領域が必要です

参照

- 「時間形式」『SQL Anywhere サーバ SQL リファレンス』
- 「CURRENT TIME 特別値」272 ページ
- 「CURRENT TIMESTAMP 特別値」273 ページ
- 「CURRENT UTC TIMESTAMP 特別値」273 ページ
- 「Ultra Light 日付と時刻関数」314 ページ
- 「DATE データ型」303 ページ
- 「DATETIME データ型」304 ページ
- 「DATE 関数 [日付と時刻]」340 ページ
- 「DATETIME 関数 [日付と時刻]」345 ページ
- 「Ultra Light の式」275 ページ
- 「GETDATE 関数 [日付と時刻]」353 ページ
- 「ISDATE 関数 [データ型変換]」359 ページ
- 「NOW 関数 [日付と時刻]」375 ページ
- 「TIMESTAMP データ型」305 ページ
- 「TIMESTAMP WITH TIME ZONE データ型」306 ページ
- 「Ultra Light timestamp_format 作成パラメータ」157 ページ
- 「Ultra Light Java Edition の timestamp_format オプション」249 ページ
- 「Ultra Light timestamp_with_time_zone_format 作成パラメータ」161 ページ
- 「Ultra Light Java Edition の timestamp_with_time_zone_format オプション」250 ページ

TIMESTAMP データ型

TIMESTAMP データ型は、年、月、日、時、分、秒、秒以下 (小数第 6 位まで) で構成される時刻を格納します。

構文

TIMESTAMP

備考

アプリケーションが文字列として TIMESTAMP 値を取得するときの形式は、timestamp_format 作成パラメータによって決まります。たとえば、TIMESTAMP 値 2010/04/01T23:59:59.999999 は、timestamp_format 作成パラメータに従い、アプリケーションに 2010/04/01 23:59:59 または April 1, 2010 23:59:59.999999 として返されます。

TIMESTAMP 値には 8 バイトの記憶領域が必要です。

TIMESTAMP データ型では、DATE 型と同じく 0001 ～ 9999 年の日付を格納できますが、TIMESTAMP データ型の有効範囲は 1600-02-28 23:59:59 ～ 7911-01-01 00:00:00 です。TIMESTAMP 値では、この範囲より前または後の時間および分の部分は保持されません。

TIMESTAMP 値を TIMESTAMP WITH TIME ZONE に変換する場合、システムのローカルタイムゾーンオフセットが最終結果に使用されます。

参照

- 「日付と時刻をデータベースに送信する方法」『SQL Anywhere サーバ SQL リファレンス』
- 「CURRENT TIME 特別値」272 ページ
- 「CURRENT TIMESTAMP 特別値」273 ページ
- 「CURRENT UTC TIMESTAMP 特別値」273 ページ
- 「Ultra Light 日付と時刻関数」314 ページ
- 「DATE データ型」303 ページ
- 「DATETIME データ型」304 ページ
- 「DATE 関数 [日付と時刻]」340 ページ
- 「DATETIME 関数 [日付と時刻]」345 ページ
- 「Ultra Light date_order 作成パラメータ」145 ページ
- 「Ultra Light Java Edition の date_order オプション」247 ページ
- 「Ultra Light の式」275 ページ
- 「GETDATE 関数 [日付と時刻]」353 ページ
- 「ISDATE 関数 [データ型変換]」359 ページ
- 「Ultra Light nearest_century 作成パラメータ」149 ページ
- 「Ultra Light Java Edition の nearest_century オプション」247 ページ
- 「NOW 関数 [日付と時刻]」375 ページ
- 「TIME データ型」304 ページ
- 「TIMESTAMP データ型」305 ページ
- 「TIMESTAMP WITH TIME ZONE データ型」306 ページ
- 「Ultra Light timestamp_format 作成パラメータ」157 ページ
- 「Ultra Light Java Edition の timestamp_format オプション」249 ページ
- 「Ultra Light timestamp_with_time_zone_format 作成パラメータ」161 ページ
- 「Ultra Light Java Edition の timestamp_with_time_zone_format オプション」250 ページ

TIMESTAMP WITH TIME ZONE データ型

TIMESTAMP WITH TIME ZONE データ型は、タイムゾーンオフセット付きの時刻を格納します。

構文

TIMESTAMP WITH TIME ZONE

備考

TIMESTAMP WITH TIME ZONE 値は、年、月、日、時、分、秒、秒の小数位、および協定世界時 (UTC: Coordinated Universal Time) との差異を示す分数を含んでいます。秒以下は 6 桁まで格納されます。

アプリケーションが文字列として TIMESTAMP WITH TIME ZONE 値を取得するときの形式は、timestamp_with_time_zone_format 作成パラメータによって決まります。たとえば、TIMESTAMP WITH TIME ZONE 値 2010/04/01T23:59:59.999999-6:00 は、timestamp_with_time_zone_format 作成パラメータに従い、アプリケーションに 2010/04/01 23:59:59 -06:00 または April 1, 2010 23:59:59.999999 -06:00 として返されます。

TIMESTAMP WITH TIME ZONE 値には 10 バイトの記憶領域が必要です。

TIMESTAMP WITH TIME ZONE データ型では、DATE 型と同じく 0001 ～ 9999 年の日付を格納できますが、TIMESTAMP WITH TIME ZONE データ型の有効範囲は 1600-02-28 23:59:59 ～ 7911-01-01 00:00:00 です。TIMESTAMP WITH TIME ZONE 値では、この範囲より前または後の時間および分の部分は保持されません。

2 つの TIMESTAMP WITH TIME ZONE 値が UTC において同じ時刻を表す場合、適用されている TIME ZONE オフセットに関係なく同じ時刻と見なされます。たとえば、次の文では、結果が同じと見なされるため、**Yes** が返されます。

```
SELECT IF CAST('2009-07-15 08:00:00 -08:00' AS TIMESTAMP WITH TIME ZONE) =
   CAST('2009-07-15 11:00:00 -05:00' AS TIMESTAMP WITH TIME ZONE)
   THEN 'Yes'
   ELSE 'No'
END IF;
```

TIMESTAMP WITH TIME ZONE 値からタイムゾーンオフセットを省略すると、タイムスタンプが日付と時刻を標準時間または夏時間のいずれで表していても、クライアントの現在の UTC オフセットがデフォルト設定されます。たとえば、クライアントが東部標準タイムゾーンに属しており、夏時間が有効なときに次の文を実行した場合、大西洋標準タイムゾーン (UTC から -4 時間) に該当するタイムゾーンでタイムスタンプが返されます。

```
SELECT CAST('2009/01/30 12:34:55' AS TIMESTAMP WITH TIME ZONE)
```

- **TIMESTAMP WITH TIME ZONE と他のデータ型の比較**　TIMESTAMP WITH TIME ZONE 値をタイムゾーン付きではないタイムスタンプと比較することはおすすめしません。これは、クライアントのデフォルトのタイムゾーンオフセットは、クライアントの地理的な場所および日付によって異なるためです。

- **TIMESTAMP WITH TIME ZONE との変換**　TIMESTAMP 値を TIMESTAMP WITH TIME ZONE に変換する場合、クライアントのタイムゾーンが結果のタイムゾーンオフセットに使用されます。つまり、値はその接続に「ローカル」なものであると見なされます。TIMESTAMP WITH TIME ZONE 値を TIMESTAMP に変換すると、オフセットは廃棄されます。文字列型、日付型、または時刻型以外の型との間の変換はサポートされていません。

参照

- 「日付と時刻の比較」『SQL Anywhere サーバ SQL リファレンス』
- 「日付と時刻をデータベースに送信する方法」『SQL Anywhere サーバ SQL リファレンス』
- 「CURRENT TIME 特別値」272 ページ
- 「CURRENT TIMESTAMP 特別値」273 ページ
- 「CURRENT UTC TIMESTAMP 特別値」273 ページ
- 「Ultra Light 日付と時刻関数」314 ページ
- 「DATE データ型」303 ページ
- 「DATETIME データ型」304 ページ
- 「DATE 関数 [日付と時刻]」340 ページ
- 「DATETIME 関数 [日付と時刻]」345 ページ
- 「Ultra Light date_order 作成パラメータ」145 ページ
- 「Ultra Light Java Edition の date_order オプション」247 ページ
- 「Ultra Light の式」275 ページ
- 「GETDATE 関数 [日付と時刻]」353 ページ
- 「ISDATE 関数 [データ型変換]」359 ページ
- 「Ultra Light nearest_century 作成パラメータ」149 ページ
- 「Ultra Light Java Edition の nearest_century オプション」247 ページ
- 「NOW 関数 [日付と時刻]」375 ページ
- 「TIME データ型」304 ページ
- 「TIMESTAMP データ型」305 ページ
- 「TIMESTAMP WITH TIME ZONE データ型」306 ページ
- 「Ultra Light timestamp_format 作成パラメータ」157 ページ
- 「Ultra Light Java Edition の timestamp_format オプション」249 ページ
- 「Ultra Light timestamp_with_time_zone_format 作成パラメータ」161 ページ
- 「Ultra Light Java Edition の timestamp_with_time_zone_format オプション」250 ページ

バイナリデータ型

バイナリデータ型は、データベースに認識されないイメージやその他の種類の情報を含めたバイナリデータを格納します。

BINARY データ型

BINARY データ型は、指定された最大長 (バイト単位) のバイナリデータを格納します。

構文

BINARY [(*max-length*)]

備考

BINARY はドメインであり、VARBINARY として実装されます。

参照

- 「VARBINARY データ型」310 ページ

LONG BINARY データ型

LONG BINARY データ型は、任意の長さのバイナリデータを格納します。

構文

LONG BINARY

備考

最大サイズは 2 GB から 1 バイトを差し引いた値です (2^{31} - 1)。

LONG BINARY 型に対してインデックスを作成することはできません。

LONG BINARY 型は LENGTH 関数と CAST 関数でのみ使用できます。

参照

- 「BINARY データ型」308 ページ
- 「VARBINARY データ型」310 ページ

UNIQUEIDENTIFIER データ型

UNIQUEIDENTIFIER データ型は、UUID (GUID とも呼ばれる) の値を格納します。

構文

UNIQUEIDENTIFIER

備考

UNIQUEIDENTIFIER データ型は、通常は、ローをユニークに識別する UUID (ユニバーサルユニーク識別子) 値を保持するために、プライマリキーまたはその他のユニークカラムに使用されます。NEWID 関数では、あるコンピュータで生成される UUID 値が他のコンピュータで生成される UUID と一致しないように UUID 値が生成されます。したがって、NEWID を使用して生成された UNIQUEIDENTIFIER 値は、同期環境でキーとして使用できます。

次に例を示します。

```
CREATE TABLE T1 (
   pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
   c1 INT );
```

UUID 値は、GUID (グローバルユニーク識別子) 値とも呼ばれます。UUID 値にはハイフンが含まれます。これは他の RDBMS と互換性を持たせるためです。 このデフォルト設定を変更するには UUIDTOSTR 関数と STRTOUUID 関数を使用します。

UNIQUEIDENTIFIER 値は、必要に応じて文字列値とバイナリ値の間で自動的に変換されます。

UNIQUEIDENTIFIER 値は、BINARY(16) として格納されますが、クライアントアプリケーションには BINARY(36) として示されます。このため、クライアントが値を文字列としてフェッチした場合に、結果に対して十分な領域が割り付けられることが保証されます。

参照

- 「NEWID 関数 [その他]」374 ページ
- 「UUIDTOSTR 関数 [文字列]」411 ページ
- 「STRTOUUID 関数 [文字列]」402 ページ
- 「Ultra Light 文字列関数」319 ページ

VARBINARY データ型

VARBINARY データ型は、指定された最大長 (バイト単位) のバイナリデータを格納します。

構文

VARBINARY [(*max-length*)]

パラメータ

- **max-length**　値の最大長 (バイト単位)。長さを指定しない場合、値は 1 になります。

  長さは 1 ～ 32767 の範囲内で指定する必要があります。

備考

比較時に、VARBINARY 値はバイトごとに正確に比較されます。この比較方法は CHAR データ型とは異なります。CHAR の場合、データベースの照合順を使用して値が比較されます。

1 つのバイナリ文字列が他の文字列のプレフィクスである場合、他の文字列と比較するときに他よりも短い文字列はゼロ (0) が埋め込まれていると見なされます。

式を評価するときのテンポラリ文字値の最大長は 2048 バイトです。

VARBINARY 値は、文字セットの変換時に変換されません。

参照

- 「BINARY データ型」308 ページ
- 「LONG BINARY データ型」『SQL Anywhere サーバ SQL リファレンス』
- 「Ultra Light 文字列関数」319 ページ
- 「ビット処理演算子」292 ページ

空間データ型

「空間データ」は、定義された空間内のオブジェクトの位置、シェイプ、方向を記述するデータです。Ultra Light には、ポイント形式の空間データ用の記憶領域およびデータ管理機能が用意されており、これらの機能を使用して、地理的ロケーションに関する情報やルート指定情報などを格納できます。ポイントは、「空間型」の ST_GEOMETRY を使用して定義します。空間データに対するアクセスと操作は、関数とコンストラクタを使用して行います。また、Ultra Light には、他の製品との互換性を確保するように設計された一連の SQL 空間関数が用意されています。

ポイントは空間内の単一の場所を定義します。ポイントジオメトリには長さも面積もありません。ポイントには必ず X 座標と Y 座標があります。

GIS データでは、ポイントは、通常、住所などの場所、または山などの地理的特性を表すために使用されます。

参照

- 「空間トピック関連の推奨ドキュメント」『SQL Anywhere サーバ 空間データサポート』

ST_GEOMETRY データ型

ST_GEOMETRY 型は、ポイント形式の空間データを格納するために使用されます。

関数

Ultra Light は次の ST_GEOMETRY 関数をサポートしています。

- 「ST_AsBinary 関数 [空間]」
- 「ST_AsExtText 関数 [空間]」
- 「ST_AsText 関数 [空間]」
- 「ST_Distance 関数 [空間]」
- 「ST_Equals 関数 [空間]」
- 「ST_IntersectsRect 関数 [空間]」
- 「ST_Point 関数 [空間]」
- 「ST_PointFromExtText 関数 [空間]」
- 「ST_PointFromText 関数 [空間]」
- 「ST_PointFromWKB 関数 [空間]」
- 「ST_SRID 関数 [空間]」
- 「ST_X 関数 [空間]」
- 「ST_Y 関数 [空間]」

備考

ST_GEOMETRY 型は、ジオメトリ型階層の最大スーパータイプです。ST_GEOMETRY 型では、任意の空間値に適用できるメソッドがサポートされています。ST_GEOMETRY 型はインスタンス化できません。代わりに、サブタイプをインスタンス化してください。元のフォーマット (WKT または WKB) を使用する場合は、ST_PointFromText/ST_PointFromWKB などのメソッドを使用して、値を元のフォーマットで表す具体的な型をインスタンス化できます。

ST_SRID メソッドを使用すると、値に関連付けられている空間参照系を取り出すことができます。

ST_GEOMETRY 型またはそのサブタイプのカラムには、プライマリキー、ユニークなインデックス、または一意性制約を入れることはできません。

カラムとオブジェクトの定義

Ultra Light では 3 種類の固定の参照系を使用でき、カラムを作成するとき、これらの参照系を属性としてカラムに割り当てることができます。個々のジオメトリオブジェクトは、未定義の参照系を除き、どの SRID 値にも関連付けることができ、一致する SRID 値または未定義の参照系に関連付けられているカラムにのみ格納できます。

事前に定義された参照系は次のとおりです。

- **未定義 (null) 参照系**　SRID 値が指定されていない場合、これがデフォルトの参照系になります。この参照系には、任意の有効な参照系を使用するジオメトリ値を含めることができます。また、この参照系では、ジオメトリオブジェクト間での参照系の一貫性を強制しない「キャッチオール」カラムが許可されます。

- **デフォルトの平面参照系**　カラムの作成時に SRID 値 0 を指定することで定義されます。このカラムには、この参照系に関連付けられているジオメトリ値のみを格納できます。値は、2D の平面空間にあるものとして処理されます。

- **WGS 84 測地参照系**　カラムの作成時に SRID 値 4326 を指定することで定義されます。このカラムには、この参照系に関連付けられているジオメトリ値のみを格納できます。値は、地球の表面上にあるものとして処理され、それに応じて操作が適用されます。

注意
SRID 4326 内のポイントは、WGS 84 参照系または未定義の参照系を使用するカラムには格納できますが、デフォルトの平面参照系を使用するカラムには格納できません。

参照系間での変換はサポートされていません。

例

次の例は、デフォルトの平面参照系に関連付けられているカラムと未定義の参照系に関連付けられているカラムを 1 つずつ持つテーブルの作成方法を示しています。

```
CREATE TABLE T1 (
   V1 INTEGER PRIMARY KEY,
   V2 ST_GEOMETRY(SRID=0),
   V3 ST_GEOMETRY
)
```

次の SQL 文は、前の例の T1 テーブルへのデータの挿入方法を示しています。

```
INSERT INTO T1(V1, V2, V3)
   VALUES (1, ST_POINTFROMTEXT('POINT(10 20)', 0), ST_POINT(5, 6, 2163))
```

ユーザ定義データ型とそれに相当する型

SQL Anywhere データベースとは異なり、Ultra Light ではユーザ定義データ型はサポートされていません。SQL Anywhere の組み込みエイリアスに相当する Ultra Light データ型を次の表に示します。

SQL Anywhere データ型	Ultra Light で相当する型
MONEY	DECIMAL(19,4)
SMALLMONEY	DECIMAL(10,4)
TEXT	LONG VARCHAR
XML	LONG VARCHAR

参照

- 「LONG VARCHAR データ型」294 ページ
- 「DECIMAL データ型」297 ページ

Ultra Light SQL 関数

関数は、データベースの情報を返すために使用します。式を使用できるところでは必ず関数を使用できます。

マニュアルに特に指定がないかぎり、関数の引数のいずれかが NULL である場合は、NULL が返されます。

関数には、SQL 文と同じ構文の表記規則を使用します。

参照

- 「SQL 構文の表記規則」『SQL Anywhere サーバ SQL リファレンス』

関数のタイプ

この項では、使用可能な関数をタイプ別に分類します。

Ultra Light では、SQL Anywhere 用に記載されている関数のサブセットをサポートしていますが、一部に違いがあります。

> **注意**
> 特に明記しないかぎり、NULL をパラメータとして受け取る関数は、NULL を返します。

参照

- 「ST_GEOMETRY データ型」311 ページ

Ultra Light 集合関数

集合関数は、データベースから選択されたローグループのデータを要約します。グループは、SELECT 文の GROUP BY 句を使用して形成されます。集合関数は、SELECT リストと、SELECT 文の HAVING 句と ORDER BY 句でのみ使用できます。

関数のリスト

次の集合関数を使用できます。

- 「AVG 関数 [集合]」
- 「COUNT 関数 [集合]」
- 「COUNT_UPLOAD_ROWS 関数 [集合]」
- 「LIST 関数 [集合]」
- 「MAX 関数 [集合]」
- 「MIN 関数 [集合]」
- 「SUM 関数 [集合]」

Ultra Light データ型変換関数

データ型変換関数を使用して、引数のデータ型を他のデータ型に変換したり、変換が可能かどうかをテストしたりします。

関数のリスト

次のデータ型変換関数を使用できます。

- 「CAST 関数 [データ型変換]」
- 「CONVERT 関数 [データ型変換]」
- 「HEXTOINT 関数 [データ型変換]」
- 「INTTOHEX 関数 [データ型変換]」
- 「ISDATE 関数 [データ型変換]」

Ultra Light 日付と時刻関数

日付と時刻の関数は、DATE、TIME、TIMESTAMP、および TIMESTAMP WITH TIME ZONE の各データ型に対する操作を実行します。

SQL Anywhere には、Transact-SQL の日付型と時間型に対する互換性サポート (DATETIME と SMALLDATETIME を含む) があります。これらの Transact-SQL データ型は、SQL Anywhere のネイティブな TIMESTAMP データ型を介したドメインとして実装されます。

日付の単位の指定

日付関数の多くは、「日付の単位」で構成される日付を使用します。次の表は、使用できる日付の単位の値を示します。

日付と時刻の関数を使用する場合、日付または時刻の減算にマイナス記号を指定できます。たとえば、31 日前のタイムスタンプを取得するには、次の文を実行します。

```
SELECT DATEADD(day, -31, NOW());
```

日付の単位	省略形	値
Year	YY	1-9999

日付の単位	省略形	値
Quarter	QQ	1-4
Month	MM	1-12
Week	WK	1-54. 日曜日を週の最初の日とします。54 週ある年は、土曜日から始まる閏年で発生します。
Day	DD	1-31
Dayofyear	DY	1-366
Weekday	DW	1 ～ 7 (日曜日 = 1、...、土曜日 = 7)
Hour	HH	0-23
Minute	MI	0-59
Second	SS	0-59
Millisecond	MS	0-999
Microsecond	MCS または US	0-999999
Calyearofweek	CYR	1-9999. その週が何年に開始したかを示します。週に年の最初の数日が含まれている場合は、その年の最初の曜日に応じて、週の最初の日が前年になる場合があります。年の最初の曜日が月曜日～木曜日の場合は、前年に属する日がその年に含まれることはありませんが、年の最初の曜日が金曜日～日曜日の場合、年の最初の週はその年の最初の月曜日から開始します。

日付の単位	省略形	値
Calweekofyear	CWK	1-53. 指定した日付がその年の第何週であるかを示します。 ISO の週のシステムと ISO 8601 の日付と時刻の標準の詳細については、http://en.wikipedia.org/wiki/ISO_week_date を参照してください。
Caldayofweek	CDW	1-7. (月曜日 = 1、...、日曜日 = 7)
TZOffset	TZ	-840 ～ 840

関数のリスト

次の日付と時刻関数を使用できます。

- 「DATE 関数 [日付と時刻]」
- 「DATEADD 関数 [日付と時刻]」
- 「DATEDIFF 関数 [日付と時刻]」
- 「DATEFORMAT 関数 [日付と時刻]」
- 「DATENAME 関数 [日付と時刻]」
- 「DATEPART 関数 [日付と時刻]」
- 「DATETIME 関数 [日付と時刻]」
- 「DAY 関数 [日付と時刻]」
- 「DAYNAME 関数 [日付と時刻]」
- 「DAYS 関数 [日付と時刻]」
- 「DOW 関数 [日付と時刻]」
- 「GETDATE 関数 [日付と時刻]」
- 「HOUR 関数 [日付と時刻]」
- 「HOURS 関数 [日付と時刻]」
- 「MINUTE 関数 [日付と時刻]」
- 「MINUTES 関数 [日付と時刻]」
- 「MONTH 関数 [日付と時刻]」
- 「MONTHNAME 関数 [日付と時刻]」
- 「MONTHS 関数 [日付と時刻]」
- 「NOW 関数 [日付と時刻]」
- 「QUARTER 関数 [日付と時刻]」
- 「SECOND 関数 [日付と時刻]」
- 「SECONDS 関数 [日付と時刻]」
- 「SWITCHOFFSET 関数 [日付と時刻]」
- 「TODAY 関数 [日付と時刻]」
- 「TODATETIMEOFFSET 関数 [日付と時刻]」
- 「WEEKS 関数 [日付と時刻]」
- 「YEAR 関数 [日付と時刻]」
- 「YEARS 関数 [日付と時刻]」
- 「YMD 関数 [日付と時刻]」

参照

- 「日付と時刻データ型」『SQL Anywhere サーバ SQL リファレンス』
- 「Ultra Light SQL データ型」293 ページ

Ultra Light のその他の関数

その他の関数は、算術式、文字列式、日付／時刻式、他の関数の戻り値に対して操作を実行します。

関数のリスト

次の各種関数を使用できます。

- 「ARGN 関数 [その他]」
- 「COALESCE 関数 [その他]」
- 「EXPLANATION 関数 [その他]」
- 「GREATER 関数 [その他]」
- 「IFNULL 関数 [その他]」
- 「ISNULL 関数 [その他]」
- 「LESSER 関数 [その他]」
- 「NEWID 関数 [その他]」
- 「NULLIF 関数 [その他]」

Ultra Light 数値関数

数値関数は、数値データ型の算術演算を実行したり、数値情報を返したりします。

関数のリスト

次の数値関数を使用できます。

- 「ABS 関数 [数値]」
- 「ACOS 関数 [数値]」
- 「ASIN 関数 [数値]」
- 「ATAN 関数 [数値]」
- 「ATAN2 関数 [数値]」
- 「CEILING 関数 [数値]」
- 「COS 関数 [数値]」
- 「COT 関数 [数値]」
- 「DEGREES 関数 [数値]」
- 「EXP 関数 [数値]」
- 「FLOOR 関数 [数値]」
- 「LOG 関数 [数値]」
- 「LOG10 関数 [数値]」
- 「MOD 関数 [数値]」
- 「PI 関数 [数値]」
- 「POWER 関数 [数値]」
- 「RADIANS 関数 [数値]」
- 「REMAINDER 関数 [数値]」
- 「ROUND 関数 [数値]」
- 「SIGN 関数 [数値]」
- 「SIN 関数 [数値]」
- 「SQRT 関数 [数値]」
- 「TAN 関数 [数値]」
- 「TRUNCNUM 関数 [数値]」

Ultra Light 空間関数

空間データは、定義された空間内のオブジェクトの位置、形、および方向を記述するデータです。Ultra Light には、ポイント形式の空間データ用の記憶領域およびデータ管理機能が用意されており、これらの機能を使用して、地理的ロケーションに関する情報やルート指定情報などを格納できます。Ultra Light には、他の製品との互換性を確保するように設計された一連の SQL 空間関数が用意されています。空間データに対するアクセスと操作は、これらの関数とコンストラクタを使用して行います。

関数のリスト

次の空間関数を使用できます。

- 「ST_AsBinary 関数 [空間]」
- 「ST_AsExtText 関数 [空間]」
- 「ST_AsText 関数 [空間]」
- 「ST_Distance 関数 [空間]」
- 「ST_Equals 関数 [空間]」
- 「ST_IntersectsRect 関数 [空間]」
- 「ST_Point 関数 [空間]」
- 「ST_PointFromExtText 関数 [空間]」
- 「ST_PointFromText 関数 [空間]」
- 「ST_PointFromWKB 関数 [空間]」
- 「ST_SRID 関数 [空間]」
- 「ST_X 関数 [空間]」
- 「ST_Y 関数 [空間]」

Ultra Light 文字列関数

文字列関数は、文字列に対して変換、抽出、操作の演算を実行したり、文字列に関する情報を返したりします。

マルチバイト文字セットを操作する場合は、使用する関数が文字とバイトのどちらの情報を返すかを十分に確認してください。

関数のリスト

次の文字列関数を使用できます。

- 「ASCII 関数 [文字列]」
- 「BYTE_LENGTH 関数 [文字列]」
- 「BYTE_SUBSTR 関数 [文字列]」
- 「CHAR 関数 [文字列]」
- 「CHARINDEX 関数 [文字列]」
- 「CHAR_LENGTH 関数 [文字列]」
- 「DIFFERENCE 関数 [文字列]」
- 「INSERTSTR 関数 [文字列]」
- 「LCASE 関数 [文字列]」
- 「LEFT 関数 [文字列]」
- 「LENGTH 関数 [文字列]」
- 「LOCATE 関数 [文字列]」
- 「LOWER 関数 [文字列]」
- 「LTRIM 関数 [文字列]」
- 「PATINDEX 関数 [文字列]」
- 「REPEAT 関数 [文字列]」
- 「REPLACE 関数 [文字列]」
- 「REPLICATE 関数 [文字列]」
- 「RIGHT 関数 [文字列]」
- 「RTRIM 関数 [文字列]」
- 「SIMILAR 関数 [文字列]」
- 「SOUNDEX 関数 [文字列]」
- 「SPACE 関数 [文字列]」
- 「STR 関数 [文字列]」
- 「STRING 関数 [文字列]」
- 「STRTOUUID 関数 [文字列]」
- 「STUFF 関数 [文字列]」
- 「SUBSTRING 関数 [文字列]」
- 「TRIM 関数 [文字列]」
- 「UCASE 関数 [文字列]」
- 「UPPER 関数 [文字列]」
- 「UUIDTOSTR 関数 [文字列]」

Ultra Light システム関数

システム関数は、システム情報を返します。

関数のリスト

Ultra Light では、次のシステム関数を使用できます。

- 「DB_PROPERTY 関数 [システム]」
- 「ML_GET_SERVER_NOTIFICATION 関数 [システム]」
- 「SYNC_PROFILE_OPTION_VALUE 関数 [システム]」

関数

関数を 1 つずつリストし、その右側に関数のタイプ (数値、文字など) を示します。

特定のタイプのすべての関数へのリンクについては、「関数のタイプ」313 ページを参照してください。

ABS 関数 [数値]

数値式の絶対値を返します。

構文

ABS(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 絶対値が返される数値。

戻り値

数値式の絶対値。

数値式データ型	戻り値
INT	INT
FLOAT	FLOAT
DOUBLE	DOUBLE
NUMERIC	NUMERIC

例

次の文は、値 66 を返します。

```
SELECT ABS( -66 );
```

ACOS 関数 [数値]

数値式のアークコサインをラジアン単位で返します。Ultra Light Java Edition データベースではサポートされていません。

構文

ACOS(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 角度のコサイン。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「ASIN 関数 [数値]」323 ページ
- 「ATAN 関数 [数値]」324 ページ
- 「ATAN2 関数 [数値]」325 ページ
- 「COS 関数 [数値]」336 ページ

例

次の文は、0.52 のアークコサイン値を返します。

```
SELECT ACOS( 0.52 );
```

ARGN 関数 [その他]

引数リストから選択された引数を返します。

構文

ARGN(*integer-expression*, *expression* [, ...] **)**

パラメータ

- ***integer-expression*** 式リスト内での引数の位置。
- ***expression*** 関数に渡される任意のデータ型の式。すべて同じデータ型の式を指定してください。

戻り値

integer-expression の値に n を使用すると、残りの引数リストから n 番目の引数 (1 から開始) が返されます。

備考

式のデータ型は任意ですが、すべて同じデータ型にしてください。integer-expression は、1 からリスト内の式の数までの範囲内で指定してください。範囲外の値を指定すると、NULL が返されます。複数の式は、カンマで区切って指定します。

例

次の文は、値 6 を返します。

```
SELECT ARGN( 6, 1,2,3,4,5,6 );
```

ASCII 関数 [文字列]

文字列式の最初のバイトの ASCII 値を整数で返します。

構文

ASCII(*string-expression* **)**

パラメータ

- ***string-expression*** 文字列。

戻り値

SMALLINT

備考

文字列が空の場合は、0 を返します。リテラル文字列は、引用符で囲んで指定します。データベース文字セットがマルチバイトであり、パラメータ文字列の最初の文字が複数バイトから構成される場合、結果は NULL です。

参照

- 「CHAR 関数 [文字列]」331 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 90 を返します。

```
SELECT ASCII( 'Z' );
```

ASIN 関数 [数値]

数値のアークサインをラジアン単位で返します。

構文

ASIN(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 角度のサイン。

戻り値

DOUBLE

備考

SIN 関数と ASIN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「ACOS 関数 [数値]」321 ページ
- 「ATAN 関数 [数値]」324 ページ
- 「ATAN2 関数 [数値]」325 ページ
- 「SIN 関数 [数値]」391 ページ

例

次の文は、0.52 のアークサイン値を返します。

```
SELECT ASIN( 0.52 );
```

ATAN 関数 [数値]

数値のアークタンジェントをラジアン単位で返します。Ultra Light Java Edition データベースではサポートされていません。

構文

ATAN(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 角度のタンジェント。

備考

ATAN 関数と TAN 関数は逆変換の演算です。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「ACOS 関数 [数値]」321 ページ
- 「ASIN 関数 [数値]」323 ページ
- 「ATAN2 関数 [数値]」325 ページ
- 「TAN 関数 [数値]」407 ページ

例

次の文は、0.52 のアークタンジェント値を返します。

```
SELECT ATAN( 0.52 );
```

ATAN2 関数 [数値]

2 つの数の比率のアークタンジェントをラジアン単位で返します。Ultra Light Java Edition データベースではサポートされていません。

構文

ATAN2 (*numeric-expression-1*, *numeric-expression-2* **)**

パラメータ

- ***numeric-expression-1*** アークタンジェントが計算される比率の分子。
- ***numeric-expression-2*** アークタンジェントが計算される比率の分母。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「ACOS 関数 [数値]」 321 ページ
- 「ASIN 関数 [数値]」 323 ページ
- 「ATAN 関数 [数値]」 324 ページ
- 「TAN 関数 [数値]」 407 ページ

例

次の文は、比率 0.52 ～ 0.60 のアークタンジェント値を返します。

```
SELECT ATAN2( 0.52, 0.60 );
```

AVG 関数 [集合]

対象となるローセットの、数値式の平均値またはユニークな値からなるセットの平均値を計算します。

構文 1

AVG([DISTINCT] *numeric-expression* **)**

パラメータ

- **[ALL]** ***numeric-expression*** 各グループのローで平均値が計算される式。
- **DISTINCT 句** グループごとにユニークな数値の平均を計算します。

戻り値

グループにローが含まれていない場合は、NULL 値を返します。

引数が DOUBLE の場合は DOUBLE を、それ以外の場合は NUMERIC を返します。

備考

この平均値には、*numeric-expression* が NULL 値のローは含まれません。

この関数ではオーバーフローエラーが発生することがあり、これにより、エラーが返される場合があります。オーバーフローエラーを回避するために、*numeric-expression* で CAST 関数を使用できます。

参照

- 「SUM 関数 [集合]」405 ページ
- 「COUNT 関数 [集合]」338 ページ

参照

- 「トラブルシューティング：集合関数と外部参照」『SQL Anywhere 16 変更点とアップグレード』

例

次の文は、SQL Anywhere 16 Demo に接続されると値 49988.623200 を返します。

```
SELECT AVG( Salary ) FROM Employees;
```

次の文は、SQL Anywhere 16 Demo に接続されると Products テーブルの製品価格の平均を返します。

```
SELECT AVG( DISTINCT UnitPrice ) FROM Products;
```

BYTE_LENGTH 関数 [文字列]

文字列のバイト数を返します。

構文

BYTE_LENGTH(*string-expression* **)**

パラメータ

- ***string-expression*** 長さが計算される文字列。

戻り値

INT

備考

string-expression の後続空白スペースは、返される長さに含まれます。

NULL 文字列の戻り値は NULL です。

マルチバイト文字セットの文字列の場合、BYTE_LENGTH の値は、CHAR_LENGTH で返される文字数と異なることがあります。

参照

- 「CHAR_LENGTH 関数 [文字列]」332 ページ
- 「DATALENGTH 関数 [システム]」339 ページ
- 「LENGTH 関数 [文字列]」361 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 12 を返します。

```
SELECT BYTE_LENGTH( 'Test Message' );
```

BYTE_SUBSTR 関数 [文字列]

文字列の部分文字列を返します。部分文字列は、文字ではなくバイトを使用して計算されます。

構文

BYTE_SUBSTR(*string-expression*, *start* [, *length*] **)**

パラメータ

- ***string-expression*** 部分文字列が取得される文字列。
- ***start*** 部分文字列の開始位置を示す整数式。正の整数の場合、文字列の先頭から開始します。先頭文字の位置は 1 です。負の整数の場合、文字列の末尾から開始します。最後の文字の位置は -1 です。
- ***length*** 部分文字列の長さを示す整数式。正の *length* は、取得するバイト数を開始位置から開始することを指定します。負の *length* は、開始位置から最大 *length* バイト左側のバイトを返します。

戻り値

BINARY、または VARCHAR。返される値は *string-expression* の型によって異なります。また、戻り値が LONG になるかどうかは、指定した引数によって決まります。たとえば、長さが 32K 未満の定数を指定した場合、LONG は返されません。

備考

length を指定すると、部分文字列は指定したバイト数に制限されます。*start* と *length* には、正または負の値を指定できます。負の数と正の数を適切に組み合わせて使用すると、文字列の先頭または末尾のどちらからでも部分文字列を取得できます。

start が 0 で length が負でない場合は、1 の *start* 値が使用されます。*start* が 0 で *length* が負の場合は、-1 の start 値が使用されます。

参照

- 「SUBSTRING 関数 [文字列]」403 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 Test を返します。

```
SELECT BYTE_SUBSTR( 'Test Message', 1, 4 );
```

CAST 関数 [データ型変換]

指定されたデータ型に変換した式の値を返します。

構文

CAST(*expression* **AS** *datatype* **)**

パラメータ

- ***expression*** 変換される式。
- ***datatype*** ターゲットのデータ型。

戻り値

要求されたデータ型によって異なります。

備考

文字列のタイプを示していない場合は、適切な長さが選択されます。DECIMAL 変換で精度も位取りも指定しない場合は、デフォルト値が選択されます。CAST 関数で精度と位取りを明示的に示すことをおすすめします。

変換で使用する値に基づいて変換する機能変換エラーが発生しないように、元のデータ型の値は新しいデータ型と互換性がある必要があります。

次のチャートを使用して、変換がサポートされているかどうかを判断します。

FROM: \ TO:	BIT	TINYINT	UNSIGNED SMALLINT	SMALLINT	UNSIGNED INTEGER	INTEGER	UNSIGNED BIGINT	BIGINT	FLOAT	REAL	DOUBLE	NUMERIC OR DECIMAL	DATE	TIME	DATETIME OR TIMESTAMP	TIMESTAMP WITH TIME ZONE	UNIQUEIDENTIFIER	BINARY OR VARBINARY	LONG BINARY	CHAR OR VARCHAR	LONG VARCHAR	ST_GEOMETRY
BIT		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗
TINYINT	⚠		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
UNSIGNED SMALLINT	⚠	⚠		✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
SMALLINT	⚠	⚠	⚠		✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
UNSIGNED INTEGER	⚠	⚠	⚠	⚠		✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
INTEGER	⚠	⚠	⚠	⚠	⚠		✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
UNSIGNED BIGINT	⚠	⚠	⚠	⚠	⚠	⚠		✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
BIGINT	⚠	⚠	⚠	⚠	⚠	⚠	⚠		✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
FLOAT	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠		⚠	✓	✓	✗	✗	✗	✗	✗	⚠	✗	✓	✗	✗
REAL	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓		✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
DOUBLE	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓	⚠		✓	✗	✗	✗	✗	✗	⚠	✗	✓	✗	✗
NUMERIC OR DECIMAL	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓	✓	✓		✗	✗	✗	✗	✗	⚠	✗	✓	✗	✗
DATE	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗		✓	✓	✓	✗	✗	✗	✓	✗	✗
TIME	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓		✓	✓	✗	✗	✓	✓	✗	✗
DATETIME OR TIMESTAMP	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓		✓	✗	✗	✗	✓	✗	✗
TIMESTAMP WITH TIME ZONE	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓		✗	✗	✗	✓	✗	✗
UNIQUEIDENTIFIER	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗		⚠	✗	✓	✗	✗
BINARY OR VARBINARY	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	⚠	✗	✗	✗	✗	⚠		✓	✓	✗	⚠
LONG BINARY	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓		✗	✗	✗
CHAR OR VARCHAR	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	⚠	✓	✗		✓	⚠
LONG VARCHAR	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓		✗
ST_GEOMETRY	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	⚠	✗	✓	✗	

記号	互換性
✓	常に変換
✗	変換しない
⚠	値依存

注意
VARBINARY と UNIQUEIDENTIFIER 間で変換を行うには、VARBINARY 値が 16 バイト長である必要があります。

NUMERIC と VARBINARY 間で変換を行うには、NUMERIC ソースが BIGINT としてもキャスト可能な値を持っている必要があります。

VARCHAR から ST_GEOMETRY に変換するには、VARCHAR ソースが WKT フォーマットまたは EWKT フォーマットのいずれかで有効なジオメトリを表している必要があります。

VARBINARY から ST_GEOMETRY に変換するには、VARBINARY ソースが WKB フォーマットで有効なジオメトリを表している必要があります。

WKB または WKT でフォーマットされているソースから ST_GEOMETRY にキャストするには、0 の SRID が ST_GEOMETRY 値に割り当てられている必要があります。ST_GEOMETRY からキャストする場合、VARCHAR 値は EWKT でフォーマットされ、VARBINARY 値は WKB でフォーマットされます。

HEXTOINT および INTTOHEX 関数を使用すると、16 進値変換を行うことができます。これらの関数の使用の詳細については、「16 進値との変換」『SQL Anywhere サーバ SQL リファレンス』を参照してください。

参照

● 「CONVERT 関数 [データ型変換]」334 ページ

例

次の関数は、文字列を日付として使用することを保証します。

```
SELECT CAST( '2000-10-31' AS DATE );
```

式 1 + 2 の値を計算し、その結果を 1 文字の文字列にキャストします。

```
SELECT CAST( 1 + 2 AS CHAR );
```

次のように CAST 関数を使用して、文字列を短縮できます。

```
SELECT CAST ( 'Surname' AS CHAR(5) );
```

VARCHAR と ST_GEOMETRY 間のキャストは通常暗黙的に行われます。たとえば、次の文は、ST_POINT 関数と VARCHAR を使用して ST_GEOMETRY カラムに値を追加します。それぞれの値はテーブルカラム全体で ST_GEOMETRY データ型に暗黙的にキャストされますが、その結果は依然として VARCHAR として示されます。

```
INSERT INTO T1 VALUES (2, ST_POINT(1,2,0), 'SRID=2163;Point(1 2)');
```

CEILING 関数 [数値]

指定した値以上になる最初の整数を返します。正の数値の場合、この処理は丸めとも呼ばれています。

構文

{ **CEILING** | **CEIL** } (*numeric-expression*)

パラメータ

- ***numeric-expression*** 切り上げ値を計算する対象の数値。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「FLOOR 関数 [数値]」352 ページ

例

次の文は、値 60 を返します。

```
SELECT CEILING( 59.84567 );
```

CHAR 関数 [文字列]

数値の ASCII 値を持つ文字を返します。

構文

CHAR(*integer-expression* **)**

パラメータ

- ***integer-expression*** ASCII 文字に変換される数値。0 ～ 255 の範囲内の数を指定します。

戻り値

VARCHAR

備考

返される文字は、バイナリソート順に従って、現在のデータベース側文字セットの指定した数値式に対応しています。

整数式の値が 255 より大きいか、0 より小さい場合、CHAR は NULL を返します。

参照

- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 Y を返します。

```
SELECT CHAR( 89 );
```

CHAR_LENGTH 関数 [文字列]

文字列の文字数を返します。

構文

CHAR_LENGTH (*string-expression* **)**

パラメータ

- ***string-expression*** 長さが計算される文字列。

戻り値

INT

備考

後続空白スペースは、返される長さに含まれます。

NULL 文字列の戻り値は NULL です。

マルチバイト文字セットの文字列の場合、CHAR_LENGTH 関数で返される値は、BYTE_LENGTH 関数で返されるバイト数と異なることがあります。

> **注意**
> データ型が CHAR、VARCHAR、LONG VARCHAR の場合、CHAR_LENGTH 関数と LENGTH 関数を使用すると同じ結果が得られます。ただし、BINARY データ型とビット配列データ型には LENGTH 関数を使用します。

参照

- 「BYTE_LENGTH 関数 [文字列]」326 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 8 を返します。

```
SELECT CHAR_LENGTH( 'Chemical' );
```

CHARINDEX 関数 [文字列]

ある文字列内で 1 つの文字列の位置を返します。

構文

CHARINDEX(*string-expression-1*, *string-expression-2* **)**

パラメータ

- ***string-expression-1*** 検索する文字列。
- ***string-expression-2*** 検索される文字列。

戻り値

INT

備考

string-expression-1 の先頭文字の位置を 1 とします。検索される文字列内に、検索する文字列のインスタンスが複数存在する場合、CHARINDEX 関数は最初のインスタンスの位置を返します。

検索される文字列内に、検索する文字列が存在しない場合、CHARINDEX 関数は 0 を返します。

いずれかの引数が NULL の場合、結果も NULL になります。

参照

- 「SUBSTRING 関数 [文字列]」403 ページ
- 「REPLACE 関数 [文字列]」384 ページ
- 「LOCATE 関数 [文字列]」363 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、姓が K という文字で始まる場合にのみ、Employees テーブルの Surname カラムと GivenName カラムから姓名を返します。

```
SELECT Surname, GivenName
FROM GROUPO.Employees
WHERE CHARINDEX( 'K', Surname ) = 1;
```

返された結果：

Surname	GivenName
Klobucher	James
Kuo	Felicia
Kelly	Moira

COALESCE 関数 [その他]

リストの中から NULL でない最初の式を返します。この関数は ISNULL 関数と同じです。

構文

COALESCE(*expression*, *expression* [, ...] **)**

パラメータ

- ***expression*** 任意の式。

  2 つ以上の式を関数に渡します。すべての式は比較可能であることが必要です。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベースサーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベースサーバは式を比較し、比較に使用したデータ型で結果を返します。データベースサーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

備考

結果が NULL になるのはすべての引数が NULL の場合のみです。

このパラメータにはスカラ型を指定できますが、同じ型を指定する必要はありません。

この関数がデータベースサーバで処理される方法の詳細については、「ISNULL 関数 [その他]」359 ページを参照してください。

参照

- 「ISNULL 関数 [その他]」359 ページ

例

次の文は、値 34 を返します。

```
SELECT COALESCE( NULL, 34, 13, 0 );
```

CONVERT 関数 [データ型変換]

指定されたデータ型に変換した式を返します。

この関数は CAST 関数と同様ですが、形式スタイルを指定して、日付と時刻のデータ型を変換するのに使用できます。 その他の変換の詳細については、「CAST 関数 [データ型変換]」328 ページを参照してください。

構文

CONVERT(*datatype*, *expression* [, *format-style*] **)**

パラメータ

- ***datatype*** 変換後の式のデータ型。
- ***expression*** 変換される式。
- ***format-style*** 出力値に適用されるスタイルコード。このパラメータを使用するのは、文字列を日付または時刻のデータ型に変換するとき、またはその反対方向に変換するときです。次の表は、サポートされるスタイルコードと、そのスタイルコードで作成される出力形式の表現です。スタイルコードは世紀を出力形式に含めるかどうかによって 2 つのカラムに分けられます (たとえば、06 と 2006 など)。

  スタイルコード 0 は、引数が指定されない場合に使用されます。

100 以上の位なし (yy) のスタイルコード	100 以上の位あり (yyyy) のスタイルコード	出力形式
-	0 または 100	Mmm dd yyyy hh:nnAA
1	101	mm/dd/yy[yy]
2	102	[yy]yy.mm.dd
3	103	dd/mm/yy[yy]
4	104	dd.mm.yy[yy]
5	105	dd-mm-yy[yy]
6	106	dd Mmm yy[yy]
7	107	Mmm dd, yy[yy]
8	108	hh:nn:ss
-	9 または 109	Mmm dd yyyy hh:nn:ss:sssAA
10	110	mm-dd-yy[yy]
11	111	[yy]yy/mm/dd
12	112	[yy]yymmdd
-	13 または 113	dd Mmm yyyy hh:nn:ss:sss (24 時間表記、ヨーロッパのデフォルト、ミリ秒、4 桁の年)
-	14 または 114	hh:nn:ss:sss (24 時間表記)
-	20 または 120	yyyy-mm-dd hh:nn:ss (24 時間表記、ODBC 標準、4 桁の年)
-	21 または 121	yyyy-mm-dd hh:nn:ss.sss (24 時間表記、ODBC 標準、ミリ秒、4 桁の年)

戻り値

指定されたデータ型によって異なります。

備考

CONVERT 関数を使用して、文字列の解析時にあいまいさがない場合、文字列を DATE、TIME、または TIMESTAMP データ型に変換できます。*format-style* が指定されている場合、データベースサーバはこれを文字列の解析方法のヒントとして使用します。データベースサーバは、文字列を明確に解析できない場合にエラーを返します。

各出力記号 (Mmm など) が出力するスタイルの詳細については、「Ultra Light date_format 作成パラメータ」142 ページを参照してください。

例

次の文は、形式スタイルの使用方法を示します。

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 104 ) FROM GROUPO.SalesOrders;
```

OrderDate
16.03.2000
20.03.2000
23.03.2000
25.03.2000
...

```
SELECT CONVERT( CHAR( 20 ), OrderDate, 7 ) FROM GROUPO.SalesOrders;
```

OrderDate
Mar 16, 00
Mar 20, 00
Mar 23, 00
Mar 25, 00
...

次の文は、整数への変換を示し、値 5 を返します。

```
SELECT CONVERT( integer, 5.2 );
```

COS 関数 [数値]

引数で与えられた角度のコサインをラジアン単位で返します。

構文

COS(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 角度 (ラジアン)。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

参照

- 「ACOS 関数 [数値]」 321 ページ
- 「COT 関数 [数値]」 337 ページ
- 「SIN 関数 [数値]」 391 ページ
- 「TAN 関数 [数値]」 407 ページ

例

次の文は、角度 0.52 ラジアンのコサイン値を返します。

```
SELECT COS( 0.52 );
```

COT 関数 [数値]

引数で与えられた角度のコタンジェントをラジアン単位で返します。

構文

COT(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 角度 (ラジアン)。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

参照

- 「COS 関数 [数値]」 336 ページ
- 「SIN 関数 [数値]」 391 ページ
- 「TAN 関数 [数値]」 407 ページ

例

次の文は、0.52 のコタンジェント値を返します。

```
SELECT COT( 0.52 );
```

COUNT 関数 [集合]

指定されたパラメータに従って、グループのロー数をカウントします。

構文 1

COUNT([* | [DISTINCT] *expression*])

パラメータ

- ***** 各グループの中のロー数を返します。セマンティック上、COUNT(*) と COUNT() は同義です。
- ***expression*** *expression* の値が NULL でないロー数をグループごとに返します。
- **DISTINCT *expression*** *expression* の値が NULL でないすべてのローについて、重複しない *expression* 値の数をグループごとに返します。

戻り値

COUNT 関数は、INT データ型の値を返します。

COUNT は、NULL 値を返しません。グループにローがないか、またはグループにおいて *expression* の値がすべて NULL である場合、COUNT は 0 を返します。

備考

COUNT 関数から返される最大値は、2147483647 です。

参照

- 「AVG 関数 [集合]」325 ページ
- 「SUM 関数 [集合]」405 ページ
- 「トラブルシューティング：集合関数と外部参照」『SQL Anywhere 16 変更点とアップグレード』

例

次の文は、ユニークな各都市と、その都市で働く従業員数を返します。

```
SELECT City, COUNT( * ) FROM GROUPO.Employees GROUP BY City;
```

COUNT_UPLOAD_ROWS 関数 [集合]

次回の同期でアップロードされるローの数を返します。

構文

COUNT_UPLOAD_ROWS(*pubs*,*threshold***)**

パラメータ

- ***pubs*** ローをチェックするパブリケーションのカンマ区切りのリスト。
- ***threshold*** カウントするローの最大数 (値 0 は最大限度を示します)。

戻り値

INT

例

次の文は、*mypub1* と *mypub2* にアップロードするローの総数を返します。

```
SELECT COUNT_UPLOAD_ROWS('mypub1,mypub2', 0);
```

DATALENGTH 関数 [システム]

式の結果に必要な基本となる記憶領域の長さ (バイト) を返します。

構文

DATALENGTH(*expression* **)**

パラメータ

- ***expression*** 通常は、カラム名です。*expression* が文字列定数の場合は、引用符で囲みます。

戻り値

UNSIGNED INT

備考

DATALENGTH 関数の戻り値は次のとおりです。

データ型	DATALENGTH
BIT	1
TINYINT	1
SMALLINT	2
INTEGER	4
BIGINT	8
REAL	4
DOUBLE	8
TIME	8
DATE	4
TIMESTAMP	8
DATETIME	8

データ型	DATALENGTH
TIMESTAMP WITH TIME ZONE	29
UNIQUEIDENTIFIER	16
CHAR	データの長さ
VARCHAR	データの長さ
BINARY	データの長さ
VARBINARY	データの長さ
NCHAR	データの長さ
NVARCHAR	データの長さ
TEXT	データの長さ
NTEXT	データの長さ
IMAGE	データの長さ
XML	データの長さ

参照

- 「SQL データ型」『SQL Anywhere サーバ SQL リファレンス』

例

次の文は、CompanyName カラムで最も長い文字列の長さ 27 を返します。

```
SELECT MAX( DATALENGTH( CompanyName ) )
FROM GROUPO.Customers;
```

次の文は、文字列 '8sdofinsv8s7a7s7gehe4h' の長さを返します。

```
SELECT DATALENGTH( '8sdofinsv8s7a7s7gehe4h' );
```

DATE 関数 [日付と時刻]

式を日付に変換し、時間、分、秒を削除します。

日付形式の解釈の制御については、「Ultra Light date_order 作成パラメータ」145 ページを参照してください。

構文

DATE(*expression* **)**

戻り値

DATE

パラメータ

- ***expression*** 日付形式に変換される値。通常、文字列。

例

次の文は、値 1999-01-02 を日付として返します。

```
SELECT DATE( '1999-01-02 21:20:53' );
```

次の文は、SYSOBJECT システムビューにリストされるすべてのオブジェクトについて、作成日を返します。

```
SELECT DATE( creation_time ) FROM SYSOBJECT;
```

DATEADD 関数 [日付と時刻]

日付の単位をその引数に追加することによって求められる TIMESTAMP または TIMESTAMP WITH TIME ZONE の値を返します。

構文

DATEADD(*date-part*, *integer-expression*, *timestamp-expression* **)**

date-part :
year
| **quarter**
| **month**
| **week**
| **day**
| **dayofyear**
| **hour**
| **minute**
| **second**
| **millisecond**
| **microsecond**

パラメータ

- ***date-part*** *integer-expression* で表される日付の単位。

  使用可能な日付の単位の完全なリストについては、「日付の単位の指定」314 ページを参照してください。

- ***integer-expression*** *timestamp-expression* に追加される *date-part* 値の数。*integer-expression* はあらゆる数値型が可能ですが、その値は INTEGER にトランケートする必要があります。

- ***timestamp-expression*** 変更される TIMESTAMP または TIMESTAMP WITH TIME ZONE の値。

戻り値

timestamp-expression が TIMESTAMP WITH TIME ZONE の場合は TIMESTAMP WITH TIME ZONE を返し、それ以外の場合は TIMESTAMP を返します。

例

次の文は TIMESTAMP 値 1995-11-02 00:00:00.000 を返します。

```
SELECT DATEADD( month, 102, '1987/05/02' );
```

次の文は TIMESTAMP 値 1987-05-02 04:00:00.000 を返します。

```
SELECT DATEADD( hour, 4, '1987/05/02' );
```

次の文は、TIMESTAMP WITH TIME ZONE 値 1987-05-06 11:33:00.000+04:00 を返します。

```
SELECT DATEADD( day, 4, CAST( '1987/05/02 11:33:00.000000+04:00' as TIMESTAMP WITH TIME
ZONE ));
```

DATEDIFF 関数 [日付と時刻]

2 つの日付間の期間を返します。

構文

DATEDIFF(*date-part*, *date-expression-1*, *date-expression-2* **)**

date-part :
year
| **quarter**
| **month**
| **week**
| **day**
| **dayofyear**
| **hour**
| **minute**
| **second**
| **millisecond**
| **microsecond**

パラメータ

- ***date-part*** 期間を測定する日付の単位を指定します。

  上記の日付オブジェクトから 1 つを選択します。日付の単位の完全なリストについては、「日付の単位の指定」314 ページを参照してください。

- ***date-expression-1*** 期間の開始日。この値を *date-expression-2* から引いて、2 つの引数の間に存在する *date-part* の数を返します。

- ***date-expression-2*** 期間の終了日。この値から *date-expression-1* を引いて、2 つの引数の間に存在する *date-part* の数を返します。

戻り値

年、四半期、月、週、日、通し日数を示す INT。時、分、秒、ミリ秒、マイクロ秒を表す BIGINT。

備考

この関数は、指定した 2 つの日付間に存在する日付の単位の数を計算します。結果は、日付の単位の数を表す (date-expression-2 - date-expression-1) と同値の符号付き整数です。

DATEDIFF 関数の結果が日付の単位の整数倍でない場合、結果は丸められずに切り捨てになります。

day を日付の単位として使用する場合、DATEDIFF 関数は指定された 2 つの時刻の間の午前 0 時の回数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

month を日付の単位として使用する場合、DATEDIFF 関数は 2 つの日付の間に存在する月の初日の数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

week を日付の単位として使用する場合、DATEDIFF 関数は 2 つの日付の間に存在する日曜日の数を返します。このとき、2 番目の日付は計算に含まれますが、最初の日付は含まれません。

例

次の文は、1 を返します。

```
SELECT DATEDIFF( hour, '4:00AM', '5:50AM' );
```

次の文は、102 を返します。

```
SELECT DATEDIFF( month, '1987/05/02', '1995/11/15' );
```

次の文は、0 を返します。

```
SELECT DATEDIFF( day, '00:00', '23:59' );
```

次の文は、4 を返します。

```
SELECT DATEDIFF( day,
  '1999/07/19 00:00',
  '1999/07/23 23:59' );
```

次の文は、0 を返します。

```
SELECT DATEDIFF( month, '1999/07/19', '1999/07/23' );
```

次の文は、1 を返します。

```
SELECT DATEDIFF( month, '1999/07/19', '1999/08/23' );
```

DATEFORMAT 関数 [日付と時刻]

日付式を表す文字列を、指定した形式で返します。

構文

DATEFORMAT(*datetime-expression*, *string-expression* **)**

パラメータ

- ***datetime-expression*** 変換される日時。
- ***string-expression*** 変換後の日付の形式。

  日付形式の説明については、「Ultra Light date_format 作成パラメータ」142 ページを参照してください。

戻り値

VARCHAR

備考

string-expression には、許容される任意の日付形式を使用できます。

例

次の文は、値 Jan 01, 1989 を返します。

```
SELECT DATEFORMAT( '1989-01-01', 'Mmm dd, yyyy' );
```

DATENAME 関数 [日付と時刻]

TIMESTAMP または TIMESTAMP WITH TIME ZONE の値の特定部分の名前 (月の名前 "June" など) を文字列で返します。

構文

DATENAME(*date-part*, *timestamp-expression* **)**

パラメータ

- ***date-part*** 名前が返される日付の単位。

  使用可能な日付の単位の完全なリストについては、「日付の単位の指定」314 ページを参照してください。

- ***timestamp-expression*** 日付の単位名が返される TIMESTAMP または TIMESTAMP WITH TIME ZONE の値。意味のある結果を取得するには、必要な *date-part* を *timestamp-expression* に含めてください。

戻り値

VARCHAR

備考

結果が数値 (23 日など) の場合でも、DATENAME 関数は文字列を返します。

参照

- 「DATEPART 関数 [日付と時刻]」345 ページ

例

次の文は、値 May を返します。

```
SELECT DATENAME( month, '1987/05/02' );
```

DATEPART 関数 [日付と時刻]

構文

DATEPART(*date-part*, *timestamp-expression* **)**

パラメータ

- ***date-part*** 名前が返される日付の単位。

  使用可能な日付の単位の完全なリストについては、「日付の単位の指定」314 ページを参照してください。

- ***timestamp-expression*** 一部分が返される TIMESTAMP または TIMESTAMP WITH TIME ZONE の値。

戻り値

INT

備考

意味のある結果を取得するには、必要な *date-part* 部分を *timestamp-expression* に含めてください。

例

次の文は、値 5 を返します。

```
SELECT DATEPART( month , '1987/05/02' );
```

次の例では、TableStatistics テーブルを作成し、SalesOrders テーブルに格納されている年ごとの注文の総数を挿入します。

```
CREATE TABLE TableStatistics (
  ID INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  Year INT,
  NumberOrders INT );
INSERT INTO TableStatistics ( Year, NumberOrders )
  SELECT DATEPART( Year, OrderDate ), COUNT(*)
  FROM GROUPO.SalesOrders
  GROUP BY DATEPART( Year, OrderDate );
```

DATETIME 関数 [日付と時刻]

式を TIMESTAMP 値に変換します。

構文

DATETIME(*expression* **)**

パラメータ

- ***expression*** 変換される式。通常は文字列です。

戻り値

TIMESTAMP

備考

数値を変換しようとするとエラーが返ります。

参照

- 「CAST 関数 [データ型変換]」328 ページ

例

次の文は、値 1998-09-09 12:12:12.000 のタイムスタンプを返します。

```
SELECT DATETIME( '1998-09-09 12:12:12.000' );
```

DAY 関数 [日付と時刻]

引数の日付を 1 ～ 31 の間の整数で返します。

構文

DAY(*date-expression* **)**

パラメータ

- ***date-expression*** DATE データ型の日付。

戻り値

SMALLINT

備考

DAY 関数は、引数の日付に対応する 1 ～ 31 の間の整数を返します。

例

次の文は、値 12 を返します。

```
SELECT DAY( '2001-09-12' );
```

DAYNAME 関数 [日付と時刻]

日付から曜日の名前を返します。

構文

DAYNAME(*date-expression* **)**

パラメータ

- ***date-expression*** 日付。

戻り値

VARCHAR

備考

返される曜日名は日本語で、日曜日、月曜日、火曜日、水曜日、木曜日、金曜日、土曜日です。

例

次の文は、値土曜日を返します。

```
SELECT DAYNAME ( '1987/05/02' );
```

DAYS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の日数を返します。詳細については、以下の「備考」の項を参照してください。

構文 1

DAYS(*timestamp-expression* **)**

構文 2

DAYS(*timestamp-expression*, *timestamp-expression* **)**

構文 3

DAYS(*timestamp-expression*, *integer-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP 値。

- ***integer-expression*** *timestamp-expression* に加算される日数。*integer-expression* が負の場合、*timestamp-expression.* から適切な日数が引かれます。整数式を指定する場合は、*timestamp-expression* を TIME、DATE または TIMESTAMP として明示的にキャストしてください。*timestamp-expression* が TIME 値の場合、現在の日付が使用されます。

  データ型のキャストの詳細については、「CAST 関数 [データ型変換]」328 ページを参照してください。

戻り値

構文 1 または構文 2 では INTEGER。

構文 3 では TIMESTAMP。

備考

DAYS 関数の結果は、その引数によって異なります。DAYS 関数では、引数内の時間、分、および秒は無視されます。

- **構文 1**　1 つの *timestamp-expression* を DAYS 関数に渡すと、0000-02-29 から *timestamp-expression* までの日数を INTEGER として返します。

> **注意**
> 0000-02-29 は実際の日付を指すための値ではありません。DAYS 関数で使用されるデフォルトの日付です。

- **構文 2**　2 つの TIMESTAMP 値を DAYS 関数に渡すと、その間の日数を整数で返します。

- **構文 3**　TIMESTAMP 値と整数を DAYS 関数に渡すと、整数値を *timestamp-expression* 引数に加算した TIMESTAMP 結果を返します。

構文 2 の代わりに、DATEDIFF 関数を使用します。構文 3 の代わりに、DATEADD 関数を使用します。

参照

- 「DATEDIFF 関数 [日付と時刻]」342 ページ
- 「DATEADD 関数 [日付と時刻]」341 ページ

例

次の文は、整数 729889 を返します。

```
SELECT DAYS( '1998-07-13 06:07:12' );
```

次の文は、整数値 -366 を返します。このことは、2 番目の DATE 値が最初の日付より 366 日前の日付であることを示します。2 つ目の例 (DATEDIFF) の使用をおすすめします。

```
SELECT DAYS( '1998-07-13 06:07:12',
             '1997-07-12 10:07:12' );

SELECT DATEDIFF( day,
  '1998-07-13 06:07:12',
  '1997-07-12 10:07:12' );
```

次の文は、TIMESTAMP 値 1999-07-14 00:00:00.000 を返します。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT DAYS( CAST('1998-07-13' AS DATE ), 366 );

SELECT DATEADD( day, 366, '1998-07-13' );
```

DB_PROPERTY 関数 [システム]

指定したプロパティの値を返します。Ultra Light Java Edition データベースではサポートされていません。

構文

DB_PROPERTY(*property-name* **)**

パラメータ

- ***property-name*** データベースプロパティ名。

戻り値

VARCHAR、LONG VARCHAR

備考

文字列を返します。

Ultra Light でオプションを設定するには、SET OPTION 文、またはコンポーネントの API に固有のデータベースオプション設定メソッドを使用します。

権限

現在のデータベースに対してこの関数を実行する場合、権限は必要ありません。この関数を他のデータベースに対して実行するには、SERVER OPERATOR または MONITOR のシステム権限を持っていることが必要です。

無効なパラメータ値を指定した場合、または必要とされるシステム権限のどれかがない場合は、NULL が戻ります。

参照

- 「Ultra Light データベースプロパティ」187 ページ
- 「SET OPTION 文 [Ultra Light]」453 ページ
- ULConnection.SetDatabaseOption メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULDatabaseSchema.SetDatabaseOption メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』

例

次の文は、現在のデータベースのページサイズをバイト単位で返します。

```
SELECT DB_PROPERTY( 'page_size');
```

DEGREES 関数 [数値]

数値をラジアンから度数に変換します。

構文

DEGREES(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 角度 (ラジアン)。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、*numeric-expression* で指定される角度を返します。パラメータが NULL 値の場合、結果は NULL 値になります。

例

次の文は、値 29.79380534680281 を返します。

```
SELECT DEGREES( 0.52 );
```

DIFFERENCE 関数 [文字列]

2 つの文字列式の SOUNDEX 値の差を返します。

構文

DIFFERENCE (*string-expression-1*, *string-expression-2* **)**

パラメータ

- ***string-expression-1*** 最初の SOUNDEX 引数。
- ***string-expression-2*** 2 番目の SOUNDEX 引数。

戻り値

SMALLINT

備考

DIFFERENCE 関数は 2 つの文字列の SOUNDEX 値を比較し、値の類似性を評価し、0 ～ 4 の値を返します。最も一致する場合は 4 です。

この関数は常に何らかの値を返します。結果が NULL になるのは引数の 1 つが NULL の場合のみです。

参照

- 「SOUNDEX 関数 [文字列]」392 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、test と chest という単語の類似点を返します。

```
SELECT DIFFERENCE( 'test', 'chest' );
```

DOW 関数 [日付と時刻]

指定した日付の曜日を表す 1 ～ 7 の数を返します (日曜日 = 1、月曜日 = 2、以下同様)。

構文

DOW(*date-expression* **)**

パラメータ

- ***date-expression*** 評価される値 (DATE データ型)。

戻り値

SMALLINT

備考

DOW 関数は、first_day_of_week データベースオプションで指定された値の影響を受けません。たとえば、first_day_of_week が月曜日に設定されている場合でも、DOW 関数は月曜日に 2 を返します。

例

次の文は、値 5 を返します。

```
SELECT DOW( '1998-07-09' );
```

次の文は、値 1 を返します。

```
SELECT DOW( CAST( '2010/05/30 11:33:00.000000+04:00' as TIMESTAMP WITH TIME ZONE ));
```

次の文は、Employees テーブルに問い合わせ、週の曜日を表す数として StartDate を返します。

```
SELECT DOW( StartDate ) FROM GROUPO.Employees;
```

EXP 関数 [数値]

自然対数の底 e を、与えられた引数でべき乗した結果を返します。Ultra Light Java Edition データベースではサポートされていません。

構文

EXP(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 指数。

戻り値

DOUBLE

備考

EXP 関数は、自然対数の底 e を *numeric-expression* で指定された値でべき乗した結果を返します。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

例

次の文は、値 3269017.3724721107 を返します。

```
SELECT EXP( 15 );
```

EXPLANATION 関数 [その他]

SQL 文の最適化方法をプレーンテキスト文字列で返します。

構文

EXPLANATION(
string-expression
[, *cursor-type*]
[, *update-status*]
)

EXPLANATION(string-expression **)**

パラメータ

- ***string-expression*** SQL 文。通常は SELECT 文ですが、UPDATE 文、MERGE 文、または DELETE 文も指定できます。

戻り値

LONG VARCHAR

備考

文のアクセスプランが文字列で返されます。オプションの詳細については、「高度：クエリ実行プラン」『SQL Anywhere サーバ SQL の使用法』 を参照してください。

GRAPHICAL_PLAN 関数では、文の最適化方法の影響を受けるシステムプロパティなど、アクセスプランに関するさまざまな情報が提供されます。

この情報は、追加するインデックスの決定や、パフォーマンスを向上するためのデータベース構造の決定に役立ちます。

参照

- 「Ultra Light の実行プラン」469 ページ

例

次の文は、SELECT 文を文字列パラメータとして渡し、クエリを実行するためのプランを返します。

```
SELECT EXPLANATION( 'SELECT * FROM Departments WHERE DepartmentID > 100' );
```

FLOOR 関数 [数値]

指定された値以下の、最大の整数値を返します。

構文

FLOOR(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** トランケートされる値。通常は、位取りがゼロ以外の固定数値型または適切な数値型 (DOUBLE、REAL、または FLOAT) です。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

参照

- 「CEILING 関数 [数値]」330 ページ

例

次の文は、123 の Floor 値を返します。

```
SELECT FLOOR (123);
```

次の文は、123 の Floor 値を返します。

```
SELECT FLOOR (123.45);
```

次の文は、-124 の Floor 値を返します。

```
SELECT FLOOR (-123.45);
```

GETDATE 関数 [日付と時刻]

現在の年、月、日、時、分、秒、秒以下を返します。

構文

GETDATE( )

戻り値

TIMESTAMP

備考

精度はシステムクロックの精度によって制限されます。

GETDATE 関数が返す情報は、NOW 関数と CURRENT TIMESTAMP 特別値が返す情報と同じです。

参照

- 「NOW 関数 [日付と時刻]」375 ページ

例

次の文は、システムの日付と時刻を返します。

```
SELECT GETDATE( );
```

GREATER 関数 [その他]

2 つのパラメータ値のうち、より大きい値を返します。

構文

GREATER(*expression-1*, *expression-2* **)**

パラメータ

- ***expression-1*** 比較される最初のパラメータ値。
- ***expression-2*** 比較される 2 番目のパラメータ値。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベースサーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベースサーバは式を比較し、比較に使用したデータ型で結果を返します。データベースサーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

備考

パラメータが等しい場合は、最初のパラメータを返します。

参照

- 「LESSER 関数 [その他]」362 ページ

例

次の文は、値 10 を返します。

```
SELECT GREATER( 10, 5 ) FROM dummy;
```

HEXTOINT 関数 [データ型変換]

16 進文字列と同等の 10 進整数を返します。

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。これらの関数の使用の詳細については、「16 進値との変換」『SQL Anywhere サーバ SQL リファレンス』を参照してください。

構文

HEXTOINT(*hexadecimal-string* **)**

パラメータ

- ***hexadecimal-string*** 整数に変換される文字列。

戻り値

HEXTOINT 関数は、プラットフォームに依存しない SQL INTEGER に相当する 16 進文字列を INT として返します。右から 8 桁目が数値 8 ～ 9 か、大文字または小文字の A ～ F のいずれかであり、その前の桁がすべて大文字または小文字の F の場合は、16 進値が負の整数値になります。次の HEXTOINT は無効な使用例です。これは、引数が符号付き 32 ビット整数で表現できない正の整数値を示しているためです。

```
SELECT HEXTOINT( '0x0080000001' );
```

備考

HEXTOINT 関数は、数値、大文字または小文字の A ～ F のみで構成される文字列リテラルまたは変数を受け入れます (0x プレフィクスが付いている場合、付いていない場合の両方)。次に HEXTOINT の有効な使用例をすべて示します。

```
SELECT HEXTOINT( '0xFFFFFFFF' );
SELECT HEXTOINT( '0x00000100' );
SELECT HEXTOINT( '100' );
SELECT HEXTOINT( '0xffffffff80000001' );
```

HEXTOINT 関数は 0x プレフィクスがあれば削除します。データが 8 桁を超える場合、符号付き 32 ビット整数値として表現できる値を示す必要があります。

参照

- 「INTTOHEX 関数 [データ型変換]」358 ページ

例

次の文は、値 420 を返します。

```
SELECT HEXTOINT( '1A4' );
```

HOUR 関数 [日付と時刻]

TIMESTAMP 値の時間部分を返します。

構文

HOUR(*timestamp-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP 値。

戻り値

SMALLINT

備考

TIMESTAMP 式の時間部分である、0 ～ 23 の間の SMALLINT 値が返されます。

例

次の文は、値 21 を返します。

```
SELECT HOUR( '1998-07-09 21:12:13' );
```

HOURS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の時間数を返します。詳細については、この関数の使用法を参照してください。

構文 1

HOURS (*timestamp-expression* **)**

構文 2

HOURS (*timestamp-expression*, *timestamp-expression* **)**

構文 3

HOURS (*time-or-timestamp-expression*, *integer-expression* **)**

パラメータ

- ***time-or-timestamp-expression*** TIME または TIMESTAMP データ型の値。
- ***timestamp-expression*** TIMESTAMP データ型の値。
- ***integer-expression*** *time-or-timestamp-expression* に加算される時間数。*integer-expression* が負の場合、*time-or-timestamp-expression* から適切な時間数が引かれます。

  データ型のキャストの詳細については、「CAST 関数 [データ型変換]」328 ページを参照してください。

戻り値

構文 1 または構文 2 では INTEGER。

構文 3 では TIME または TIMESTAMP。

備考

HOURS 関数の結果は、その引数によって異なります。

- **構文 1** 1 つの *timestamp-expression* を HOURS 関数に渡すと、0000-02-29 の真夜中から *timestamp-expression* までの時間数を INTEGER として返します。

  > **注意**
  > 0000-02-29 は実際の日付を指すための値ではありません。HOURS 関数で使用されるデフォルトの TIMESTAMP 値です。

- **構文 2** 2 つの TIMESTAMP 値を HOURS 関数に渡すと、その間の時間数を整数で返します。

- **構文 3**　TIMESTAMP 値と INTEGER 値を HOURS 関数に渡すと、整数の時間数を *time-or-timestamp-expression* 引数に加算した TIMESTAMP 結果を返します。同様に、TIME 値を最初の引数として渡すと、TIME 値を結果として返します。構文 3 では、最初の引数の暗黙的変換をサポートしていません。最初の引数を DATE、TIME または TIMESTAMP の値に明示的にキャストしてください。最初の引数が DATE の場合、時刻部分には真夜中が使用されます。

構文 2 の代わりに、DATEDIFF 関数を使用します。構文 3 の代わりに、DATEADD 関数を使用します。

参照

- 「DATEDIFF 関数 [日付と時刻]」342 ページ
- 「DATEADD 関数 [日付と時刻]」341 ページ

例

次の文は、値 4 を返します。これは、2 番目の TIMESTAMP 値が、最初のタイムスタンプの 4 時間後であることを示します。2 つ目の例 (DATEDIFF) の使用をおすすめします。

```
SELECT HOURS( '1999-07-13 06:07:12', '1999-07-13 10:07:12' );

SELECT DATEDIFF( hour, '1999-07-13 06:07:12', '1999-07-13 10:07:12' );
```

次の文は、値 17517342 を返します。

```
SELECT HOURS( '1998-07-13 06:07:12' );
```

次の文は、日時 1999-05-13 02:05:07.000 を返します。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT HOURS( CAST( '1999-05-12 21:05:07' AS DATETIME ), 5 );

SELECT DATEADD( hour, 5, '1999-05-12 21:05:07' );
```

IFNULL 関数 [その他]

最初の式が NULL 値の場合は、2 番目の式の値を返します。最初の式が NULL でない場合は、3 番目の式の値を返します。最初の式が NULL ではなく、3 番目の式が存在しない場合は、NULL を返します。

構文

IFNULL(*expression-1*, *expression-2* [, *expression-3*] **)**

パラメータ

- ***expression-1***　評価される式。この値によって、*expression-2* または *expression-3* のどちらが返るかが決まります。
- ***expression-2***　*expression-1* が NULL の場合の戻り値。
- ***expression-3***　*expression-1* が NULL でない場合の戻り値。

戻り値

返されるデータ型は、*expression-2* と *expression-3* のデータ型によって異なります。

例

次の文は、値 -66 を返します。

```
SELECT IFNULL( NULL, -66 );
```

次の文は、最初の式が NULL ではなく、3 番目の式が存在しないため、NULL を返します。

```
SELECT IFNULL( -66, -66 );
```

INSERTSTR 関数 [文字列]

別の文字列の指定された位置に文字列を挿入します。

構文

INSERTSTR(*integer-expression*, *string-expression-1*, *string-expression-2* **)**

パラメータ

- ***integer-expression*** 文字列の挿入位置。文字列の先頭に挿入する場合は、0 を使用します。
- ***string-expression-1*** 別の文字列が挿入される文字列。
- ***string-expression-2*** 挿入する文字列。

戻り値

LONG VARCHAR

参照

- 「STUFF 関数 [文字列]」403 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 backoffice を返します。

```
SELECT INSERTSTR( 0, 'office ', 'back' );
```

INTTOHEX 関数 [データ型変換]

整数に対応する 16 進値の文字列を返します。

構文

INTTOHEX(*integer-expression* **)**

パラメータ

- ***integer-expression*** 16 進に変換される整数。

戻り値

VARCHAR

備考

CAST、CONVERT、HEXTOINT、INTTOHEX 関数を使用すると、16 進値変換を行うことができます。

参照

- 「HEXTOINT 関数 [データ型変換]」354 ページ
- 「16 進値との変換」『SQL Anywhere サーバ SQL リファレンス』

例

次の文は、値 0000009c を返します。

```
SELECT INTTOHEX( 156 );
```

ISDATE 関数 [データ型変換]

文字列引数を日付に変換できるかどうかをテストします。

構文

ISDATE(*string* **)**

パラメータ

- ***string*** 文字列が有効な日付を表すかどうかを判断するために分析される文字列。

戻り値

INT

備考

変換できる場合は 1 を返し、できない場合は 0 を返します。引数が NULL の場合は 0 を返します。

ISNULL 関数 [その他]

リストの中から NULL でない最初の式を返します。この関数は COALESCE 関数と同じです。

構文

ISNULL(*expression*, *expression* [, ...] **)**

パラメータ

- ***expression*** NULL かどうかがテストされる式。

  2 つ以上の式を関数に渡します。すべての式は比較可能であることが必要です。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベースサーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベースサーバは式を比較し、比較に使用したデータ型で結果を返します。データベースサーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

参照

- 「COALESCE 関数 [その他]」333 ページ

例

次の文は、値 -66 を返します。

```
SELECT ISNULL( NULL ,-66, 55, 45, NULL, 16 );
```

LCASE 関数 [文字列]

文字列中のすべての文字を小文字に変換します。

構文

LCASE(*string-expression* **)**

パラメータ

- ***string-expression*** 小文字に変換される文字列。

戻り値

- CHAR
- LONG VARCHAR
- VARCHAR

備考

LCASE 関数は LOWER 関数と同じです。

参照

- 「LOWER 関数 [文字列]」366 ページ
- 「UCASE 関数 [文字列]」410 ページ
- 「UPPER 関数 [文字列]」411 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 chocolate を返します。

```
SELECT LCASE( 'ChoCOlatE' );
```

LEFT 関数 [文字列]

文字列の先頭からいくつかの文字を返します。

構文

LEFT(*string-expression*, *integer-expression* **)**

パラメータ

- ***string-expression*** 文字列。
- ***integer-expression*** 返す文字数。

戻り値

LONG VARCHAR

備考

文字列にマルチバイト文字があり、適切な照合が使用されている場合、返されるバイト数が指定した文字数よりも大きい場合があります。

引数 string-expression の値より大きな *integer-expression* を指定できます。この場合、全体の値が返されます。

入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値が文字長のセマンティックで記述されます。

参照

- 「RIGHT 関数 [文字列]」385 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、Customers テーブルに含まれる各 Surname 値の最初の 5 文字を返します。

```
SELECT LEFT( Surname, 5) FROM GROUPO.Customers;
```

LENGTH 関数 [文字列]

指定した文字列の文字数を返します。

構文

LENGTH(*string-expression* **)**

パラメータ

- ***string-expression*** 文字列。

戻り値

INT

備考

この関数を使用すると、文字列の長さがわかります。たとえば、*string-expression* にカラム名を指定すると、カラムの値の長さがわかります。

文字列にマルチバイト文字があり、適切な照合が使用されている場合、LENGTH はバイト数ではなく、文字数を返します。文字列が BINARY データ型の場合、LENGTH 関数は BYTE_LENGTH 関数のように動作します。

> **注意**
> データ型が CHAR、VARCHAR、LONG VARCHAR の場合、LENGTH 関数と CHAR_LENGTH 関数を使用すると同じ結果が得られます。ただし、BINARY データ型とビット配列データ型には LENGTH 関数を使用します。

参照

- 「BYTE_LENGTH 関数 [文字列]」326 ページ
- 「国際言語と文字セット」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 9 を返します。

```
SELECT LENGTH( 'chocolate' );
```

LESSER 関数 [その他]

2 つのパラメータ値のうち、より小さい値を返します。

構文

LESSER(*expression-1*, *expression-2* **)**

パラメータ

- ***expression-1*** 比較される最初のパラメータ値。
- ***expression-2*** 比較される 2 番目のパラメータ値。

戻り値

この関数の戻り値は、指定した式によって異なります。具体的には、データベースサーバが関数を評価するとき、まず、式の比較が可能なデータ型を検索します。該当するデータ型が見つかると、データベースサーバは式を比較し、比較に使用したデータ型で結果を返します。データベースサーバは、一般に比較が可能なデータ型を見つけることができないと、エラーを返します。

備考

パラメータが等しい場合は、最初の値を返します。

参照

- 「GREATER 関数 [その他]」354 ページ

例

次の文は、値 5 を返します。

```
SELECT LESSER( 10, 5 ) FROM dummy;
```

LIST 関数 [集合]

グループ内のローごとに値のデリミタ付きリストを返します。

構文

LIST(
[**DISTINCT**] *string-expression*
[, *delimiter-string*] **)**

パラメータ

- ***string-expression*** 文字列式。通常はカラム名です。グループ内のローごとに *string-expression* の値が結果文字列に追加されます。各値は、*delimiter-string* で区切られます。DISTINCT が指定されていると、ユニークな *string-expression* 値のみが追加されます。

- ***delimiter-string*** リスト項目のデリミタ文字列。デフォルト設定はカンマです。NULL 値または空の文字列を指定した場合は、デリミタはありません。*delimiter-string* は定数です。

戻り値

LONG VARCHAR

備考

LIST 関数は、グループの各ローの、NULL 以外のすべての X の値を (デリミタ付きで) 連結して返します。グループ内に明確な X 値を持ったローが 1 つ以上存在しない場合、LIST(X) は空の文字列を返します。

LIST 関数では、NULL 値と空の文字列は無視されます。

LIST 関数は Window 関数として使用できませんが、Window 関数の入力には使用できます。

例

次の文は、Employees テーブルのすべての住所を返します。

```
SELECT LIST( Street ) FROM GROUPO.Employees;
```

LOCATE 関数 [文字列]

異なる文字列内のある文字列の位置を返します。

構文

LOCATE(*string-expression-1*, *string-expression-2* [, *integer-expression*] **)**

パラメータ

- ***string-expression-1*** 検索される文字列。
- ***string-expression-2*** 検索する文字列。
- ***integer-expression*** 文字列内の、検索を開始する位置。最初の文字の位置は 1 です。開始オフセットが負の場合は、最初に一致した文字列ではなく、最後に一致した文字列のオフセットを返します。負のオフセットは、文字列の末尾のどれだけの部分が検索から除外されるかを示します。除外されるバイト数は、(-1 * offset) -1 で計算されます。

戻り値

INT

備考

integer-expression を指定すると、文字列のオフセットから検索が開始します。

最初の文字列には長い文字列 (255 バイト以上) を指定できますが、2 番目の文字列は 255 バイトに制限されます。長い文字列を 2 番目の引数として指定すると、LOCATE 関数は NULL 値を返します。文字列が見つからない場合は、0 を返します。長さ 0 の文字列を検索すると、1 を返します。いずれかの引数が NULL の場合は、結果は NULL です。

マルチバイト文字が使用され、適切な照合が使用されている場合は、開始位置と返される値は**バイト**で計算した位置と異なる場合があります。

参照

- 「Ultra Light 文字列関数」319 ページ
- 「CHARINDEX 関数 [文字列]」332 ページ

例

次の文は、値 8 を返します。

```
SELECT LOCATE(
  'office party this week - rsvp as soon as possible',
  'party',
  2 );
```

LOG 関数 [数値]

数の自然対数を返します。Ultra Light Java Edition データベースではサポートされていません。

構文

LOG(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 数値。

戻り値

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。パラメータが NULL 値の場合、結果は NULL 値になります。

備考

引数は、組み込みの数値データ型の値を返す式です。

参照

- 「LOG10 関数 [数値]」365 ページ

例

次の文は、50 の自然対数を返します。

```
SELECT LOG( 50 );
```

LOG10 関数 [数値]

数値の対数 (基数 10) を返します。Ultra Light Java Edition データベースではサポートされていません。

構文

LOG10(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 数値。

戻り値

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。パラメータが NULL 値の場合、結果は NULL 値になります。

備考

引数は、組み込みの数値データ型の値を返す式です。

参照

- 「LOG 関数 [数値]」364 ページ

例

次の文は、50 の底が 10 の対数を返します。

```
SELECT LOG10( 50 );
```

LOWER 関数 [文字列]

文字列中のすべての文字を小文字に変換します。この関数は LCASE 関数と同じです。

構文

LOWER(*string-expression* **)**

パラメータ

- ***string-expression*** 小文字に変換される文字列。

戻り値

引数のデータ型に対応する CHAR、VARCHAR、または LONG VARCHAR。

備考

LCASE 関数は LOWER 関数と同じです。

参照

- 「LCASE 関数 [文字列]」360 ページ
- 「UCASE 関数 [文字列]」410 ページ
- 「UPPER 関数 [文字列]」411 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 chocolate を返します。

```
SELECT LOWER( 'chOCOLate' );
```

LTRIM 関数 [文字列]

文字列の先行ブランクを削除します。

構文

LTRIM(*string-expression* **)**

パラメータ

- ***string-expression*** 削除する文字列。

戻り値

- VARCHAR
- LONG VARCHAR

備考

結果の実際の長さは、式の長さから、削除する文字数を引いた値です。すべての文字を削除すると、結果は空の文字列になります。

パラメータに NULL を指定できる場合、結果は NULL になります。

パラメータが NULL の場合、結果は NULL 値になります。

参照

- 「RTRIM 関数 [文字列]」387 ページ
- 「TRIM 関数 [文字列]」409 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、すべての先行ブランクを削除して、値 Test Message を返します。

```
SELECT LTRIM( '    Test Message' );
```

MAX 関数 [集合]

各ローグループで見つかった式の最大値を返します。

構文 1

MAX([DISTINCT] *expression* **)**

パラメータ

- ***expression*** 最大値が計算される式。通常はカラム名です。
- **DISTINCT *expression*** MAX(*expression*) の場合と同じ値を返します。万全を期すために含まれています。

戻り値

引数と同じデータ型。

備考

expression が NULL のローは、無視されます。グループにローが含まれていない場合は、NULL を返します。

また、2 つの式を単純に比較する場合は、GREATER 関数を使用することもできます。

参照

- 「MIN 関数 [集合]」368 ページ
- 「トラブルシューティング：集合関数と外部参照」『SQL Anywhere 16 変更点とアップグレード』

例

次の文は、Employees テーブル内の給与の最大値 138948.000 を返します。

```
SELECT MAX( Salary )
FROM GROUPO.Employees;
```

MIN 関数 [集合]

各ローグループで見つかった expression の最小値を返します。

構文 1

MIN([DISTINCT] *expression* **)**

パラメータ

- ***expression*** 最小値が計算される式。通常はカラム名です。
- **DISTINCT *expression*** MIN(*expression*) の場合と同じ値を返します。万全を期すために含まれています。

戻り値

引数と同じデータ型。

備考

expression が NULL のローは、無視されます。グループにローが含まれていない場合は、NULL を返します。

また、2 つの式を単純に比較する場合は、LESSER 関数を使用することもできます。「LESSER 関数 [その他]」『SQL Anywhere サーバ SQL リファレンス』を参照してください。

参照

- 「MAX 関数 [集合]」367 ページ

例

次の文は、Employees テーブル内の給与の最小値 24903.000 を返します。

```
SELECT MIN( Salary )
FROM GROUPO.Employees;
```

MINUTE 関数 [日付と時刻]

TIMESTAMP 値の分部分を返します。

構文

MINUTE(*timestamp-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP 値。

戻り値

SMALLINT

備考

TIMESTAMP 式の分部分である、0 ～ 59 の間の SMALLINT 値が返されます。

例

次の文は、値 22 を返します。

```
SELECT MINUTE( '1998-07-13 12:22:34' );
```

MINUTES 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の分数を返します。次の「備考」の項を参照してください。

構文 1

MINUTES(*timestamp-expression* **)**

構文 2

MINUTES(*timestamp-expression*, *timestamp-expression* **)**

構文 3

MINUTES(*timestamp-or-time-expression*, *integer-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP データ型の式。
- ***timestamp-or-time-expression*** TIME または TIMESTAMP データ型の式。
- ***integer-expression*** *timestamp-or-time-expression* に加算される分数。*integer-expression* が負の場合、*timestamp-or-time-expression* から適切な分数が引かれます。

戻り値

構文 1 または構文 2 では INTEGER。

構文 3 では TIME または TIMESTAMP。

備考

MINUTES 関数の結果は、その引数によって異なります。

- **構文 1** 1 つの *timestamp-expression* を MINUTES 関数に渡すと、0000-02-29 の真夜中から *timestamp-expression* までの分数を INTEGER として返します。

  > **注意**
  > 0000-02-29 は実際の日付を指すための値ではありません。MINUTES 関数で使用されるデフォルトの日付です。

- **構文 2** 2 つの TIMESTAMP 値を MINUTES 関数に渡すと、その間の分数を整数で返します。
- **構文 3** TIMESTAMP 値と INTEGER 値を MINUTES 関数に渡すと、整数の分数を *timestamp-expression* 引数に加算した TIMESTAMP 結果を返します。同様に、MINUTES の最初の引数

が TIME 値の場合は、結果も TIME 値になります。構文 3 では、最初の引数の暗黙的変換をサポートしていません。最初の引数を DATE、TIME または TIMESTAMP の値に明示的にキャストしてください。最初の引数が DATE データ型の場合、時刻部分には真夜中が使用されます。

MINUTES は整数を返すので、構文 1 で 4083-03-23 02:08:00 以上の TIMESTAMP 値を使用すると、オーバーフローが起こる場合があります。

構文 2 の代わりに、DATEDIFF 関数を使用します。構文 3 の代わりに、DATEADD 関数を使用します。

参照

- 「DATEDIFF 関数 [日付と時刻]」『SQL Anywhere サーバ SQL リファレンス』
- 「DATEADD 関数 [日付と時刻]」『SQL Anywhere サーバ SQL リファレンス』
- 「CAST 関数 [データ型変換]」328 ページ

例

次の文は、値 240 を返します。これは、2 番目の TIMESTAMP 値が、最初のタイムスタンプの 240 分後であることを示します。2 つ目の例 (DATEDIFF) の使用をおすすめします。

```
SELECT MINUTES( '1999-07-13 06:07:12',
   '1999-07-13 10:07:12' );

SELECT DATEDIFF( minute,
    '1999-07-13 06:07:12',
    '1999-07-13 10:07:12' );
```

次の文は、値 1051040527 を返します。

```
SELECT MINUTES( '1998-07-13 06:07:12' );
```

次の文は、TIMESTAMP 値 1999-05-12 21:10:07.000 を返します。最初の文では、リテラル文字列パラメータの明示的なキャストが必要です。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT MINUTES( CAST( '1999-05-12 21:05:07' AS TIMESTAMP ), 5);

SELECT DATEADD( minute, 5, '1999-05-12 21:05:07' );
```

ML_GET_SERVER_NOTIFICATION 関数 [システム]

この関数を使用すると、Ultra Light ユーザはライトウェイトポーリングを使用して、サーバ起動同期要求について Mobile Link サーバ上の Notifier にクエリできます。Ultra Light Java Edition データベースではサポートされていません。

構文

ML_GET_SERVER_NOTIFICATION(*notifier*, *address*, *key* **)**

パラメータ

- ***Notifier*** ポーリング対象の Mobile Link サーバ上の Notifier 名。

- ***address*** ストリームパラメータ。たとえば、

  tcpip{host=pc1;port=1234}

- ***key*** 省略可。通知要求キーです。

戻り値

指定された要求キーに対する通知要求の件名と内容を返します。

備考

指定された要求キーに対する要求がない場合、または指定された Notifier 名が Mobile Link サーバ上に存在しない場合、結果は NULL になります。要求キーに NULL が指定された場合、そのユーザのリモート ID が要求キーとして使用されます。要求が存在する場合、結果メッセージが **[***件名***]***内容* (たとえば、[sync]profile1) という形式で返されます。

この関数は、Mobile Link サーバからの応答を取得する際にネットワーク経由の通信を行います。そのため、ネットワーク遅延により実行に時間がかかることがあります。実行中、この関数をバックグラウンドで実行できる期間がある場合があります。その場合、他の接続上のランタイム内で作業を行うことができます。ただし、このような期間の存在が保証されているわけではなく、SQL の複雑さに依存します。この関数で使用する Mobile Link アドレスを取得する場合は、sync_profile_option_value 関数で既存の同期プロファイルを使用して、**Stream** プロファイルオプションの値を取得することをおすすめします。この関数が返す値は、そのまま Mobile Link アドレスパラメータとして使用できます。

参照

- 「SYNC_PROFILE_OPTION_VALUE 関数 [システム]」406 ページ

例

```
SELECT ML_GET_SERVER_NOTIFICATION('Notifier1', 'tcpip{host=sybase;port=1234}', 'MyKey');
```

MOD 関数 [数値]

整数を整数で割ったときの余りを返します。

構文

MOD(*dividend*, *divisor* **)**

パラメータ

- ***dividend*** 被除数 (分数の分子)。
- ***divisor*** 除数 (分数の分母)。

戻り値

- SMALLINT
- INT
- NUMERIC

備考

被除数が負の場合、除算の結果は負または 0 になります。除数の符号は計算結果に影響を与えません。

参照

- 「REMAINDER 関数 [数値]」382 ページ

例

次の文は、値 2 を返します。

```
SELECT MOD( 5, 3 );
```

MONTH 関数 [日付と時刻]

指定した日付の月を返します。

構文

MONTH(*date-expression* **)**

パラメータ

- ***date-expression*** DATE データ型の値。

戻り値

SMALLINT

備考

戻り値は、指定の日付の月に対応する 1 ～ 12 の間の数です。

例

次の文は、値 7 を返します。

```
SELECT MONTH( '1998-07-13' );
```

MONTHNAME 関数 [日付と時刻]

日付から月の名前を返します。

構文

MONTHNAME(*date-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP 値。

戻り値

VARCHAR

備考

結果が数値 (2 月を示す 2 など) の場合でも、MONTHNAME 関数は文字列を返します。

参照

- 「DATEPART 関数 [日付と時刻]」345 ページ

例

次の文は、値 September を返します。

```
SELECT MONTHNAME( '1998-09-05' );
```

MONTHS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の月数を返します。次の「備考」の項を参照してください。

構文 1

MONTHS(*timestamp-expression* **)**

構文 2

MONTHS(*timestamp-expression*, *timestamp-expression* **)**

構文 3

MONTHS(*timestamp-expression*, *integer-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP データ型の日付と時刻。
- ***integer-expression*** *timestamp-expression* に加算される整数の月数 (SMALLINT データ型)。*integer-expression* が負の場合、*timestamp-expression* から適切な月数が引かれます。*integer-expression* を指定する場合は、*timestamp-expression* を TIME、DATE または TIMESTAMP データ型として明示的にキャストしてください。*timestamp-expression* が TIME 値の場合、現在の月が使用されます。

  データ型のキャストの詳細については、「CAST 関数 [データ型変換]」328 ページを参照してください。

戻り値

構文 1 または構文 2 では INTEGER。

構文 3 では TIMESTAMP。

備考

MONTHS 関数の結果は、その引数によって異なります。MONTHS 関数では、引数内の時間、分、および秒は無視されます。

- **構文 1**　1 つの *timestamp-expression* を MONTHS 関数に渡すと、0000-02 から *timestamp-expression* までの月数を INTEGER として返します。

> **注意**
> 0000-02-29 は実際の日付を指すための値ではありません。MONTHS 関数で使用されるデフォルトの日付です。

- **構文 2**　2 つの TIMESTAMP 値を MONTHS 関数に渡すと、その間の月数を整数で返します。
- **構文 3**　TIMESTAMP 値と SMALLINT 値を MONTHS 関数に渡すと、整数の月数を *timestamp-expression* に加算した TIMESTAMP 結果を返します。

構文 2 の代わりに、DATEDIFF 関数を使用します。構文 3 の代わりに、DATEADD 関数を使用します。

MONTHS の値を求めるには、2 つの日付の間に月の最初の日がいくつあるかを計算します。

参照

- 「DATEDIFF 関数 [日付と時刻]」342 ページ
- 「DATEADD 関数 [日付と時刻]」341 ページ

例

次の文は、値 2 を返します。これは、2 番目の日付が、最初の日付の 2 か月後であることを示します。2 つ目の例 (DATEDIFF) の使用をおすすめします。

```
SELECT MONTHS( '1999-07-13 06:07:12', '1999-09-13 10:07:12' );

SELECT DATEDIFF( month,
  '1999-07-13 06:07:12',
  '1999-09-13 10:07:12' );
```

次の文は、値 23981 を返します。

```
SELECT MONTHS( '1998-07-13 06:07:12' );
```

次の文は、TIMESTAMP 値 1999-10-12 21:05:07.000 を返します。2 つ目の例 (DATEADD) の使用をおすすめします。

```
SELECT MONTHS( CAST( '1999-05-12 21:05:07' AS DATETIME ), 5);

SELECT DATEADD( month, 5, '1999-05-12 21:05:07' );
```

NEWID 関数 [その他]

UUID (ユニバーサルユニーク識別子) 値を生成します。UUID は、GUID (グローバルユニーク識別子) と同じです。

構文

NEWID( )

パラメータ

NEWID 関数に関連付けられているパラメータはありません。

戻り値

UNIQUEIDENTIFIER

備考

NEWID 関数は、カラムの DEFAULT 句で使用できます。

UUID を使用して、テーブルのローをユニークに識別できます。コンピュータが異なると生成される値も異なるので、値は同期環境やレプリケーション環境でキーとして使用できます。

参照

- 「NEWID デフォルト」『SQL Anywhere サーバ SQL の使用法』
- 「STRTOUUID 関数 [文字列]」402 ページ
- 「UUIDTOSTR 関数 [文字列]」411 ページ

例

次の文は、2 つのカラムを持つテーブル mytab を作成します。カラム pk は uniqueidentifier データ型とし、NEWID 関数をデフォルト値として割り当てます。カラム c1 は integer データ型です。

```
CREATE TABLE mytab(
  pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
  c1 INT );
```

次の文は、ユニーク識別子を文字列として返します。

```
SELECT UUIDTOSTR( NEWID() );
```

たとえば、戻り値が 96603324-6FF6-49DE-BF7D-F44C1C7E6856 の場合もあります。

NOW 関数 [日付と時刻]

現在の日付と時刻を TIMESTAMP 値で返します。精度はシステムクロックの精度によって制限されます。

構文

NOW([*])

戻り値

TIMESTAMP

備考

NOW は、GETDATE 関数および CURRENT TIMESTAMP 特別値と同義です。NOW(*) と NOW() の構成は同義です。

要求に含まれる NOW 関数の各インスタンスは、多くても 1 回のみ評価されます。同じ要求に含まれる NOW の複数のインスタンスは、同じ TIMESTAMP 値を共有する場合もあれば、そうでない場合もあります。

参照

- 「GETDATE 関数 [日付と時刻]」353 ページ
- 「CURRENT TIMESTAMP 特別値」273 ページ

例

次の文は、現在の日付と時刻を返します。

```
SELECT NOW( * );
```

NULLIF 関数 [その他]

式を比較して、CASE 式の省略形を提供します。

構文

NULLIF(*expression-1*, *expression-2* **)**

パラメータ

- ***expression-1*** 比較される式。
- ***expression-2*** 比較される式。

戻り値

最初の引数のデータ型。

備考

NULLIF は 2 つの式の値を比較します。

1 番目の式と 2 番目の式が等しい場合、NULLIF は NULL を返します。

1 番目の式と 2 番目の式が異なる場合、または 2 番目の式が NULL の場合、NULLIF は 1 番目の式を返します。

NULLIF 関数は、いくつかの CASE 式の簡単な作成方法を提供します。

参照

- 「CASE 式」277 ページ

例

次の文は、値 a を返します。

```
SELECT NULLIF( 'a', 'b' );
```

次の文は、NULL を返します。

```
SELECT NULLIF( 'a', 'a' );
```

PATINDEX 関数 [文字列]

文字列のパターンが最初に出現した開始位置を表す整数を返します。

構文

PATINDEX('%*pattern*%', *string-expression* **)**

パラメータ

- ***pattern*** 検索するパターン。先頭の % ワイルドカードを省略すると、PATINDEX 関数は、パターンが文字列の先頭に出現する場合は 1 を返し、それ以外の場合は 0 を返します。

  Ultra Light のパターンは、次のワイルドカードを使用します。

ワイルドカード	一致するもの
_ (アンダースコア)	任意の 1 文字
% (パーセント記号)	0 個以上の文字からなる任意の文字列
[]	指定範囲内、または一連の指定文字の任意の 1 文字
[^]	指定範囲外、または一連の指定文字以外の任意の 1 文字

- ***string-expression*** パターンを検索する文字列。

戻り値

INT

備考

PATINDEX 関数は、パターンが最初に出現した開始位置を返します。パターンが見つからない場合は、0 を返します。

参照

- 「LOCATE 関数 [文字列]」363 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 2 を返します。

```
SELECT PATINDEX( '%hoco%', 'chocolate' );
```

次の文は、値 11 を返します。

```
SELECT PATINDEX( '%4_5_', '0a1A 2a3A 4a5A' );
```

次の文は、14 を返します。これは、文字列式の最初の英数字以外の文字です。データベースで大文字と小文字が区別されない場合は、'%[^a-zA-Z0-9]%' の代わりにパターン '%[^a-z0-9]%' を使用できます。

```
SELECT PATINDEX( '%[^a-zA-Z0-9]%', 'SQLAnywhere16 has many new features' );
```

次の文を使用して、文字列の最初の英数字以外の文字までのすべての文字を取得することができます。

```
SELECT LEFT( @string, PATINDEX( '%[^a-zA-Z0-9]%', @string ) );
```

次の文は、myTable というテーブルを作成し、英数字文字、スペース (ブランク)、英数字以外の文字が含まれる各種の文字列を入力します。続いて、SELECT 文とその後の結果により、PATINDEX を使用して文字列内のスペースと英数字以外の文字の開始位置を検索する方法がわかります。

```
CREATE TABLE myTable( col1 LONG VARCHAR );

INSERT INTO myTable (col1) VALUES( 'the quick brown fox jumped over the lazy dog' ),
( 'the quick brown fox $$$$ jumped over the lazy dog' ),
( 'the quick brown fox 0999 jumped over the lazy dog' ),
( 'the quick brown fox ** jumped over the lazy dog' ),
( 'thequickbrownfoxjumpedoverthelazydog' ),
( 'thequickbrownfoxjum999pedoverthelazydog' ),
( 'thequick$$$$brownfox' ),
( 'the quick brown fox$$ jumped over the lazy dog' );

SELECT col1,
  //position of first non-alphanumeric character or space:
  PATINDEX( '%[^a-z0-9]%', col1) AS blank_posn,
  //position of first non-alphanumeric char that isn't a space:
  PATINDEX( '%[^ a-z0-9]%', col1) AS non_alpha_char,
  //everything up to and including first non-alphanumeric char that isn't a space:
  LEFT ( col1, PATINDEX( '%[^ a-zA-Z0-9]%', col1) ) AS left_str,
  //first non-alphanumeric char that isn't a space, and everything to the right:
  SUBSTRING ( col1, PATINDEX( '%[^ a-zA-Z0-9]%', col1) ) AS sub_str
FROM myTable;
```

col1	blank_posn	non_alpha_char	left_str	sub_str
the quick brown fox jumped over the lazy dog	4	0		the quick brown fox jumped over the lazy dog
the quick brown fox $$$$ jumped over the lazy dog	4	21	the quick brown fox $	$$$$ jumped over the lazy dog
the quick brown fox 0999 jumped over the lazy dog	4	0		the quick brown fox 0999 jumped over the lazy dog

col1	blank_posn	non_alpha_char	left_str	sub_str
the quick brown fox ** jumped over the lazy dog	4	21	the quick brown fox *	** jumped over the lazy dog
thequickbrownfox jumpedoverthelazydog	0	0		thequickbrownfox jumpedoverthelazydog
thequickbrownfox jum999pedoverthelazydog	0	0		thequickbrownfox jum999pedoverthelazydog
thequick$$$$brownfox	9	9	thequick$	$$$$brownfox
the quick brown fox$$ jumped over the lazy dog	4	20	the quick brown fox$	$$ jumped over the lazy dog

PI 関数 [数値]

PI の数値を返します。

構文

PI([*])

戻り値

DOUBLE

備考

この関数は DOUBLE 値を返します。

セマンティック上、PI(*) と PI() は同義です。

例

次の文は、値 3.141592653(...) を返します。

```
SELECT PI( * );
```

POWER 関数 [数値]

数のべき乗を表す数を計算します。

構文

POWER(*numeric-expression-1*, *numeric-expression-2* **)**

パラメータ

- ***numeric-expression-1*** べき乗の底。
- ***numeric-expression-2*** 指数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。引数のいずれかが NULL の場合、結果は NULL 値になります。

例

次の文は、値 64 を返します。

```
SELECT POWER( 2, 6 );
```

QUARTER 関数 [日付と時刻]

指定した TIMESTAMP 式から、四半期を示す数を返します。

構文

QUARTER(*timestamp-expression* **)**

パラメータ

- ***timestamp-expression*** 四半期にする日付。

戻り値

INTEGER

備考

各四半期は次のように定めます。

四半期	期間 (開始日と終了日を含む)
1	1 月 1 日～ 3 月 31 日
2	4 月 1 日～ 6 月 30 日
3	7 月 1 日～ 9 月 30 日
4	10 月 1 日～ 12 月 31 日

例

次の文は、値 2 を返します。

```
SELECT QUARTER( '1987/05/02' );
```

RADIANS 関数 [数値]

度数をラジアンに変換します。

構文

RADIANS(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 度数。この角度をラジアンに変換します。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、倍精度浮動小数点で計算を行います。

例

次の文は、約 0.5236 の値を返します。

```
SELECT RADIANS( 30 );
```

RAND 関数 [数値]

オプションのシードから、間隔 0 ～ 1 の乱数を返します。Ultra Light Java Edition データベースではサポートされていません。

構文

RAND([*integer-expression*] **)**

パラメータ

- ***integer-expression*** 乱数の作成に使用するオプションのシード。この引数を使用して、繰り返し可能な乱数列を作成できます。

戻り値

DOUBLE

備考

RAND 関数は、乗算線形合同法で乱数を生成します。詳細については、CACM 31 (10) の Park と Miller (1988) の寄稿 (1192 ～ 1201 ページ) と、Press 他 (1992) の『Numerical Recipes in C』(第 2 版の第 7 章 279 ページ、邦訳は『ニューメリカルレシピインシー日本語版—C 言語による数値

計算のレシピ』) を参照してください。279). RAND 関数の呼び出し結果は、0 < *n* < 1 の疑似乱数 *n* です (0.0 と 1.0 はどちらも結果に含まれません)。

サーバへの接続が確立するときに、この乱数生成関数は初期値のシードを指定します。各接続で異なる乱数シーケンスが生成されるように、各接続には一意のシードが指定されます。また、シード値 (*integer-expression*) を引数に指定することもできます。通常、シード値の指定は、以降の RAND 関数の呼び出しで乱数シーケンスを要求する前に 1 回だけ実行します。複数回シード値を初期化すると、シーケンスは再開されます。同じシード値を指定すると、同じシーケンスが生成されます。値が近いシード値の場合、似た初期シーケンスが生成されますが、シーケンスが進むにつれて相違が多くなります。

適切な乱数結果を取得するためでも、あるシード値から生成されたシーケンスと、別のシード値から生成されたシーケンスとは結合しないでください。つまり、乱数値のシーケンスを生成しているときに、シード値をリセットしないでください。

RAND 関数は非決定的関数として扱われます。クエリオプティマイザは、RAND 関数の結果をキャッシュしません。

非決定的関数の詳細については、「関数のキャッシュ」『SQL Anywhere サーバ SQL の使用法』を参照してください。

例

次の文は偶数の乱数結果を生成します。以降、シードを指定しないで RAND 関数を呼び出すと、毎回異なる結果が生成されます。

```
SELECT RAND( 1 );
SELECT RAND( ), RAND( ), RAND( ), RAND( ), RAND( );
SELECT RAND( ), RAND( ), RAND( ), RAND( ), RAND( );
```

次の文では、シーケンスが同じ 2 つの結果セットが生成されます。これはシード値が 2 度指定されるためです。

```
SELECT RAND( 1 ), RAND( ), RAND( ), RAND( ), RAND( );
SELECT RAND( 1 ), RAND( ), RAND( ), RAND( ), RAND( );
```

次の例では、値が互いに近く、分布の点からは乱数とは言えない 5 つの結果が生成されます。このため、シード値が似た RAND 関数を複数回呼び出すことはおすすめしません。

```
SELECT RAND( 1 ), RAND( 2 ), RAND( 3 ), RAND( 4 ), RAND( 5 );
```

次の例は、5 つのまったく同じ結果が生成されるため、回避する必要があります。

```
SELECT RAND( 1 ), RAND( 1 ), RAND( 1 ), RAND( 1 ), RAND( 1 );
```

REMAINDER 関数 [数値]

整数を整数で割ったときの余りを返します。

構文

REMAINDER(*dividend*, *divisor* **)**

パラメータ

- ***dividend*** 被除数 (分数の分子)。
- ***divisor*** 除数 (分数の分母)。

戻り値

- INTEGER
- NUMERIC

備考

MOD 関数を使用して、余りを返すこともできます。

参照

- 「MOD 関数 [数値]」371 ページ

例

次の文は、値 2 を返します。

```
SELECT REMAINDER( 5, 3 );
```

REPEAT 関数 [文字列]

文字列を指定された回数だけ連結します。

構文

REPEAT(*string-expression*, *integer-expression* **)**

パラメータ

- ***string-expression*** 繰り返される文字列。
- ***integer-expression*** 文字列を繰り返す回数。*integer-expression* が負の場合は、空の文字列を返します。

戻り値

LONG VARCHAR

備考

実際の結果文字列の長さが戻り値の最大長を超えると、エラーが発生します。この場合、結果は文字列に使用できる最大サイズまでトランケートされます。

この関数の動作は、REPLICATE 関数と同じです。

参照

- 「REPLICATE 関数 [文字列]」385 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 repeatrepeatrepeat を返します。

```
SELECT REPEAT( 'repeat', 3 );
```

REPLACE 関数 [文字列]

文字列を別の文字列で置換し、新しい結果を返します。

構文

REPLACE(*original-string*, *search-string*, *replace-string* **)**

パラメータ

いずれかの引数が NULL の場合、NULL を返します。

- ***original-string*** 検索される文字列。任意の長さの文字列を指定できます。
- ***search-string*** 検索して *replace-string* に置換する文字列。この文字列は 255 バイトに制限されています。*search-string* が空の文字列の場合は、元の文字列を未変更のまま返します。
- ***replace-string*** *search-string* と置き換える置換文字列。任意の長さの文字列を指定できます。*replacement-string* が空の文字列の場合は、すべての *search-string* が削除されます。

戻り値

LONG VARCHAR

備考

この関数はすべてのオカレンスを置換します。

大文字と小文字を区別するデータベースでの比較では、大文字と小文字が区別されます。

参照

- 「SUBSTRING 関数 [文字列]」403 ページ
- 「CHARINDEX 関数 [文字列]」332 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 xx.def.xx.ghi を返します。

```
SELECT REPLACE( 'abc.def.abc.ghi', 'abc', 'xx' );
```

次の文は、ALTER PROCEDURE 文を含む結果セットを生成します。ALTER PROCEDURE を実行すると、名前が変更されたテーブルを参照する格納済みプロシージャが修復されます (テーブル名を一意にすることをおすすめします)。

```
SELECT REPLACE(
  REPLACE( proc_defn, 'OldTableName', 'NewTableName' ),
  'CREATE PROCEDURE',
  'ALTER PROCEDURE')
```

```
FROM SYS.SYSPROCEDURE
WHERE proc_defn LIKE '%OldTableName%';
```

REPLICATE 関数 [文字列]

文字列を指定された回数だけ連結します。

構文

REPLICATE(*string-expression*, *integer-expression* **)**

パラメータ

- ***string-expression*** 繰り返される文字列。
- ***integer-expression*** 文字列を繰り返す回数。

戻り値

LONG VARCHAR

備考

実際の結果文字列の長さが戻り値の最大長を超えると、エラーが発生します。この場合、結果は文字列に使用できる最大サイズまでトランケートされます。

この関数の動作は、REPEAT 関数と同じです。

参照

- 「REPEAT 関数 [文字列]」383 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 repeatrepeatrepeat を返します。

```
SELECT REPLICATE( 'repeat', 3 );
```

RIGHT 関数 [文字列]

文字列の一番右側にある文字を返します。

構文

RIGHT(*string-expression*, *integer-expression* **)**

パラメータ

- ***string-expression*** 一番右側にある文字を返す文字列。
- ***integer-expression*** 返される文字列の末尾の文字数。

戻り値

LONG VARCHAR

備考

文字列にマルチバイト文字が含まれている場合は、返されるバイト数が指定した文字数よりも大きい場合があります。

カラムの値より大きな *integer-expression* を指定できます。この場合、全体の値が返されます。

入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値が文字長のセマンティックで記述されます。

参照

- 「LEFT 関数 [文字列]」361 ページ
- 「国際言語と文字セット」『SQL Anywhere サーバ データベース管理』
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、Customers テーブルに含まれる各 Surname 値の最後の 5 文字を返します。

```
SELECT RIGHT( Surname, 5 ) FROM GROUPO.Customers;
```

ROUND 関数 [数値]

integer-expression で指定した小数点以下の桁数に *numeric-expression* を丸めます。

構文

ROUND(*numeric-expression*, *integer-expression* **)**

パラメータ

- ***numeric-expression*** 関数に渡される、丸めの対象となる数。
- ***integer-expression*** 正の整数は、丸めを行う小数点の右側の有効桁数を指定します。負の式は、丸めを行う小数点の左側の有効桁数を指定します。

戻り値

NUMERIC

備考

この関数の結果は numeric または double です。数値の結果があり、整数 *integer-expression* が負の値の場合、精度は 1 ずつ増えます。

参照

- 「TRUNCNUM 関数 [数値]」409 ページ

例

次の文は、値 123.200 を返します。

```
SELECT ROUND( 123.234, 1 );
```

RTRIM 関数 [文字列]

文字列の後続ブランクを削除します。

構文

RTRIM(*string-expression* **)**

パラメータ

- ***string-expression*** 削除する文字列。

戻り値

- VARCHAR
- LONG VARCHAR

備考

結果の実際の長さは、式の長さから、削除する文字数を引いた値です。すべての文字を削除すると、結果は空の文字列になります。

引数が NULL の場合、結果は NULL 値になります。

参照

- 「TRIM 関数 [文字列]」409 ページ
- 「LTRIM 関数 [文字列]」366 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、すべての後続ブランクが削除された文字列 Test Message を返します。

```
SELECT RTRIM( 'Test Message     ' );
```

SECOND 関数 [日付と時刻]

TIMESTAMP 引数の秒の値を返します。

構文

SECOND(*timestamp-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP 値。

戻り値

SMALLINT

備考

指定した TIMESTAMP 引数値の秒に相当する 0 ～ 59 の数を返します。

例

次の文は、値 25 を返します。

```
SELECT SECOND( '1998-07-13 21:21:25' );
```

SECONDS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の秒数を返します。次の「備考」の項を参照してください。

構文 1

SECONDS(*timestamp-expression* **)**

構文 2

SECONDS(*timestamp-expression*, *timestamp-expression* **)**

構文 3

SECONDS(*time-or-timestamp-expression*, *integer-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP 値。
- ***time-or-timestamp-expression*** TIME または TIMESTAMP データ型の値。
- ***integer-expression*** *time-or-timestamp-expression* に加算される秒数。*integer-expression* が負の場合、*time-or-timestamp-expression* から適切な秒数が引かれます。整数式を指定する場合は、*time-or-timestamp-expression* を TIME、DATE または TIMESTAMP データ型として明示的にキャストしてください。*time-or-timestamp-expression* が DATE データ型の場合、時刻部分には真夜中が使用されます。

戻り値

構文 1 では UNSIGNED BIGINT。

構文 2 では SIGNED BIGINT。

構文 3 では TIME または TIMESTAMP。

備考

SECONDS 関数の結果は、その引数によって異なります。

- **構文 1**　1 つの *timestamp-expression* を SECONDS 関数に渡すと、0000-02-29 の真夜中から *timestamp-expression* までの秒数を UNSIGNED BIGINT として返します。

> **注意**
> 0000-02 は実際の日付を指すための値ではありません。SECONDS 関数で使用されるデフォルトの日付です。

- **構文 2**　2 つの TIMESTAMP 値を SECONDS 関数に渡すと、その間の秒数を整数の SIGNED BIGINT 値で返します。

- **構文 3**　TIMESTAMP 値と INTEGER 値を SECONDS 関数に渡すと、整数の秒数を *time-or-timestamp-expression* に加算した TIMESTAMP 結果を返します。同様に、TIME 値を SECONDS 関数に渡すと、TIME データ型の値を返します。

構文 2 の代わりに、DATEDIFF 関数を使用します。構文 3 の代わりに、DATEADD 関数を使用します。

参照

- 「CAST 関数 [データ型変換]」328 ページ
- 「DATEADD 関数 [日付と時刻]」341 ページ
- 「DATEDIFF 関数 [日付と時刻]」342 ページ

例

次の文は、値 14400 を返します。これは、2 番目の TIMESTAMP 値が、最初のタイムスタンプの 14400 秒後であることを示します。

```
SELECT SECONDS( '1999-07-13 06:07:12',
  '1999-07-13 10:07:12' );
SELECT DATEDIFF( second,
  '1999-07-13 06:07:12',
  '1999-07-13 10:07:12' );
```

次の文は、値 63062431632 を返します。

```
SELECT SECONDS( '1998-07-13 06:07:12' );
```

次の文は、TIMESTAMP 値 1999-05-12 21:05:12.000 を返します。

```
SELECT SECONDS( CAST( '1999-05-12 21:05:07' AS TIMESTAMP ), 5);
SELECT DATEADD( second, 5, '1999-05-12 21:05:07' );
```

SHORT_PLAN 関数 [その他]

SQL 文の Ultra Light プランの最適化方法を短い記述の文字列で返します。この記述は、EXPLANATION 関数が返す記述と同じです。

構文

SHORT_PLAN(*string-expression* **)**

備考

クエリによっては、SQL Anywhere に選択したプランと Ultra Light の実行プランが異なる場合があります。

パラメータ

- ***string-expression*** SQL 文。通常は SELECT 文ですが、UPDATE 文または DELETE 文も指定できます。

戻り値

LONG VARCHAR

参照

- 「EXPLANATION 関数 [その他]」352 ページ

例

次の文は、SELECT 文を文字列パラメータとして渡し、クエリを実行するためのプランを返します。

```
SELECT SHORT_PLAN(
  'SELECT * FROM GROUPO.Departments WHERE DepartmentID > 100' );
```

SIGN 関数 [数値]

指定された数値の符号 (正または負) を返します。

構文

SIGN(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 符号が返される数値。*numeric-expression* には、INTEGER、DOUBLE、NUMERIC のデータ型を使用できます。

戻り値

SMALLINT

備考

負の数を指定すると、SIGN 関数は -1 を返します。

0 を指定すると、SIGN 関数は 0 を返します。

正の数を指定すると、SIGN 関数は 1 を返します。

例

次の文は、値 -1 を返します。

```
SELECT SIGN( -550 );
```

SIMILAR 関数 [文字列]

2 つの文字列の類似性を示す数を返します。

構文

SIMILAR(*string-expression-1*, *string-expression-2* **)**

パラメータ

- ***string-expression-1*** 比較する最初の文字列。
- ***string-expression-2*** 比較する 2 番目の文字列。

戻り値

SMALLINT

備考

SIMILAR 関数は、2 つの文字列の類似性を表す 0 ～ 100 の整数を返します。結果は、2 つの文字列の文字が一致している割合として解釈できます。値が 100 の場合は、2 つの文字列は同じです。

この関数を使用して、名前 (顧客名など) のリストを修正できます。顧客がわずかに異なる名前で重複して登録されている場合があります。SIMILAR 関数を使用して、類似する顧客名を見つけることができます。このことを行うには、customer テーブルをそのテーブル自体にジョインし、類似性が 90 パーセントより大きく、かつ 100 パーセント未満のすべての顧客のレポートを作成します。

SIMILAR 関数で実行される計算は、単に文字数が一致する場合よりも複雑になります。

参照

- 「Ultra Light 文字列関数」319 ページ

例

次の文は値 75 を返します。2 つの値が 75 % 前後であることを示します。

```
SELECT SIMILAR( 'toast', 'coast' );
```

SIN 関数 [数値]

数のサインを返します。

構文

SIN(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 角度 (ラジアン)。

戻り値

DOUBLE

備考

SIN 関数は、引数のサインを返します。この引数はラジアン単位で表現される角度です。SIN 関数と ASIN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

参照

- 「ASIN 関数 [数値]」323 ページ
- 「COS 関数 [数値]」336 ページ
- 「COT 関数 [数値]」337 ページ
- 「TAN 関数 [数値]」407 ページ

例

次の文は、0.52 の SIN 値を返します。

```
SELECT SIN( 0.52 );
```

SOUNDEX 関数 [文字列]

文字列の発音を表す数を返します。

構文

SOUNDEX(*string-expression* **)**

パラメータ

- ***string-expression*** 評価される文字列。

戻り値

SMALLINT

備考

文字列の SOUNDEX 関数の値は、先頭の文字とそれに続く H、Y、W 以外の 3 つの子音がベースになっています。文字列の最初の文字である場合を除き、*string-expression* の母音は無視されます。同じ文字が 2 つ続く場合は 1 文字としてカウントされます。たとえば、"apples" という単語は、文字 A、P、L、S がベースになります。

マルチバイト文字は、SOUNDEX 関数では無視されます。

完全とは言えませんが、SOUNDEX 関数は通常、類似発音で同じ文字で始まる語句に対して同じ数を返します。

SOUNDEX 関数は、英単語の処理に最も優れています。他の言語の処理は英語の場合よりも劣ります。

参照

- 「Ultra Light 文字列関数」319 ページ

例

次の文は、それぞれの名前の発音を表す 2 つの同じ数 3827 を返します。

```
SELECT SOUNDEX( 'Smith' ), SOUNDEX( 'Smythe' );
```

SPACE 関数 [文字列]

指定した数のスペースを返します。

構文

SPACE(*integer-expression* **)**

パラメータ

- ***integer-expression*** 返すスペースの数。

戻り値

LONG VARCHAR

備考

integer-expression が負の場合は、NULL 文字列を返します。

参照

- 「Ultra Light 文字列関数」319 ページ

例

次の文は、10 のスペースから成る文字列を返します。

```
SELECT SPACE( 10 );
```

SQRT 関数 [数値]

数の平方根を返します。

構文

SQRT(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 平方根が計算される数。

戻り値

DOUBLE

備考

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

例

次の文は、値 3 を返します。

```
SELECT SQRT( 9 );
```

ST_AsBinary 関数 [空間]

指定されたジオメトリを表すバイナリ文字列を返します。出力フォーマットは、OGC SFS 1.1 で定義されている WKB です。このフォーマットには、Z 値と M 値は含まれません。

構文

ST_AsBinary(*geometry-expression*)

戻り値

- **BINARY** *geometry-expression* の WKB 表現を返します。

例

次の文は、結果 0x01010000000000000000000f03f0000000000000040 を返します。

```
SELECT ST_AsBinary(ST_Point(1.0, 2.0, 4326))
```

ST_AsBinary 関数は、サーバでジオメトリを BINARY に変換するときに暗黙的に呼び出されます。たとえば、次の文は結果 0x01010000000000000000000f03f0000000000000040 を返します。

```
SELECT CAST(ST_Point(1.0, 2.0, 4326) AS BINARY(50))
```

ST_AsExtText 関数 [空間]

指定されたジオメトリを表すバイナリ文字列を返します。出力フォーマットは EWKT です。

構文

ST_AsExtText(*geometry-expression*)

戻り値

- **VARCHAR** *geometry-expression* の EWKT 表現を返します。

例

次の文は、SRID をプレフィクスとして含む結果 SRID=4326;Point (1 2) を返します。

```
SELECT ST_AsExtText(ST_Point(1.0, 2.0, 4326))
```

ST_AsExtText() 関数は、ジオメトリを VARCHAR 型に変換するときに暗黙的に呼び出されます。たとえば、次の文は結果 SRID=4326;Point(1 2) を返します。

```
SELECT CAST(ST_Point(1.0, 2.0, 4326) AS VARCHAR(25))
```

ST_AsText 関数 [空間]

指定されたジオメトリを表すバイナリ文字列を返します。出力フォーマットは、OGC SFS 1.1 で定義されている WKT です。

構文

ST_AsText(*geometry-expression*)

戻り値

- **VARCHAR** *geometry-expression* の WKT 表現を返します。

例

次の文は、結果 Point (1 2) を返します。

```
SELECT ST_AsText(ST_Point(1.0, 2.0, 4326))
```

ST_Distance 関数 [空間]

指定した 2 つのジオメトリ値間の最短距離を返します。ポイントが SRID 4326 内にある場合、単位はメートルです。

構文

ST_Distance(*geo1*,*geo2* **)**

パラメータ

名前	型	説明
geo1	ST_Geometry	2 つのジオメトリ値間の距離を計算するために使用する最初のジオメトリ値
geo2	ST_Geometry	2 つのジオメトリ値間の距離を計算するために使用する 2 番目のジオメトリ値

戻り値

- **DOUBLE** 指定したジオメトリ値間の最短距離を返します。

例

次の文は、結果 3367142.4632130372 を返します。

```
SELECT ST_Distance(ST_Point(-79.38,43.65,4326),ST_Point(-123.1,49.28,4326))
```

ST_Equals 関数 [空間]

ST_Geometry 値が別の ST_Geometry 値と空間的に等しいかどうかをテストします。2 つのジオメトリ値が同じ x 座標と y 座標を使用し、同じ参照系にある場合、値は等しいと見なされます。

空間参照系の精度またはデータの精度によってテストが制限される場合があります。

ST_Equals 関数は、比較述部 (= と <>)、IN リスト述部、DISTINCT、GROUP BY に使用されるセマンティックを定義します。

構文

ST_Equals(*geo1*,*geo2* **)**

パラメータ

名前	型	説明
geo1	ST_Geometry	比較する最初のジオメトリ値
geo2	ST_Geometry	比較する 2 番目のジオメトリ値

戻り値

- **BIT**　2 つのジオメトリ値が空間的に等しい場合は 1 を返し、それ以外の場合は 0 を返します。

例

次の文は、結果 1 を返します。

```
SELECT ST_Equals(ST_Point(1,1,4326),ST_Point(1,1,4326))
```

ST_IntersectsRect 関数 [空間]

2 つのポイント (最小と最大) によって指定されたボックス内にポイントがあるかどうかをテストします。

構文

ST_IntersectsRect(*location*,*min*,*max***)**

パラメータ

名前	型	説明
location	ST_Geometry	テストするポイント
min	ST_Geometry	ボックスの定義に使用する最小ポイント値

名前	型	説明
max	ST_Geometry	ボックスの定義に使用する最大ポイント値

戻り値

- **BIT**　*location* が指定したボックスと交差する場合は 1 を返し、それ以外の場合は 0 を返します。

例

次の文は、結果 1 を返します。

```
SELECT ST_IntersectsRect(ST_Point(1,1,4326),ST_Point(0,0,4326), ST_Point(3,3,4326))
```

ST_Point 関数 [空間]

x 座標と y 座標に基づくポイントを作成します。

構文

ST_Point(*x*,*y*,*SRID*)

パラメータ

名前	型	説明
x	DOUBLE	ポイントの作成に使用する x 座標
y	DOUBLE	ポイントの作成に使用する y 座標
SRID	INTEGER	ポイントに関連付けられる SRID 値

戻り値

- **ST_Point**　入力文字列から作成された ST_Geometry 値を返します。

例

次の文は、2163 参照系にポイント (10.0,20.0) を作成します。

```
SELECT ST_Point(10.0,20.0,2163)
```

ST_PointFromExtText 関数 [空間]

ST_Geometry の EWKT 表現を含む VARCHAR 値から変換される ST_Geometry 値を返します。

構文

ST_PointFromText(*ewkt*)

パラメータ

名前	型	説明
ewkt	VARCHAR	EWKT 表現

戻り値

- **ST_Geometry**　入力文字列から作成された ST_Geometry 値を返します。

例

次の文は、結果 SRID=4326;Point(10 20) を返して、4326 参照系にポイント (10,20) が作成されたことを示します。

```
SELECT ST_PointFromExtText('SRID=4326;Point(10 20)')
```

ST_PointFromText 関数 [空間]

ST_Geometry の WKT 表現を含む VARCHAR 値から変換される ST_Geometry 値を返します。

構文

ST_PointFromText(*wkt*, *srid***)**

パラメータ

名前	型	説明
wkt	VARCHAR	WKT 表現
srid	INT	結果の空間参照系識別子は、SRID パラメータで示したものです。

戻り値

- **ST_Geometry**　入力文字列から作成された ST_Geometry 値を返します。

  結果の空間参照系識別子は、パラメータ *srid* で指定します。

例

次の文は、結果 SRID=4326;Point(10 20) を返して、4326 参照系にポイント (10,20) が作成されたことを示します。

```
SELECT ST_PointFromText('Point(10 20)',4326)
```

ST_PointFromWKB 関数 [空間]

ST_Geometry の WKB 表現を含む BINARY 値から変換される ST_Geometry 値を返します。

構文

ST_PointFromWKB(*wkb*, *srid***)**

パラメータ

名前	型	説明
wkb	BINARY	WKB 表現
srid	INTEGER	ポイントに関連付けられる SRID 値

戻り値

- **ST_Geometry**　入力文字列から作成された ST_Geometry 値を返します。

例

次の文は、結果 (1.0, 2.0, 4326) を返します。

```
SELECT ST_PointFromWKB(0x01010000000000000000000f03f0000000000000040,4326)
```

ST_SRID 関数 [空間]

ジオメトリ値に関連付けられている空間参照系 (SRID) を取得します。

構文

ST_SRID(*geo1*, *srid* **)**

パラメータ

名前	型	説明
geo1	ST_Geometry	ポイント値
srid	INTEGER	ポイントに関連付けられる SRID 値

戻り値

- **INT**　ジオメトリの SRID を返します。

例

次の文は、結果 4326 を返します。

```
SELECT ST_SRID( ST_Point ( 10, 20, 4326 ) );
```

ST_X 関数 [空間]

ST_Geometry 値の x 座標を返します。

構文

ST_X(geo1)

パラメータ

名前	型	説明
geo1	ST_Geometry	x 座標を決定する ST_Geometry 値。

戻り値

- **DOUBLE**　ST_Geometry 値の x 座標を返します。

例

次の文は、結果 10.0 を返します。

```
SELECT ST_X(ST_Point(10.0,20.0,4326))
```

ST_Y 関数 [空間]

ST_Geometry 値の y 座標を返します。

構文

.**ST_Y**(geo1)

パラメータ

名前	型	説明
geo1	ST_Geometry	y 座標を決定する ST_Geometry 値。

戻り値

- **DOUBLE**　ST_Geometry 値の y 座標を返します。

例

次の例では、結果として 20.0 を返します。

```
SELECT ST_Y(ST_Point(10.0,20.0,4326))
```

STR 関数 [文字列]

指定した数に相当する文字列を返します。

構文

STR(*numeric-expression* [, *length* [, *decimal*]] **)**

パラメータ

- ***numeric-expression*** -1E126 と 1E127 との間の任意の概数 (浮動小数点、実数、または倍精度)。
- ***length*** 返される文字数 (小数点、小数点の左右のすべての桁、ブランクを含む)。デフォルトは 10 です。
- ***decimal*** 返される小数点以下の桁数。デフォルトは 0 です。

戻り値

VARCHAR

備考

数の整数部分が指定した長さに合わない場合は、指定した長さの文字列がすべてアスタリスクで埋められて返されます。たとえば、次の文は *** を返します。

```
SELECT STR( 1234.56, 3 );
```

注意
サポートされている最大長は 128 です。長さが 1 ～ 128 の範囲外である場合は、結果が NULL になります。

参照

- 「Ultra Light 文字列関数」319 ページ

例

次の文は、6 個のスペースと 1235 (合計 10 文字) の文字列を返します。

```
SELECT STR( 1234.56 );
```

次の文は、結果 1234.6 を返します。

```
SELECT STR( 1234.56, 6, 1 );
```

STRING 関数 [文字列]

1 つ以上の文字列を連結して 1 つの長い文字列にします。

構文

STRING(*string-expression* [, ...] **)**

パラメータ

- ***string-expression*** 評価される文字列。

引数を 1 つだけ指定する場合は、1 つの式に変換されます。複数の引数を指定する場合は、連結されて 1 つの文字列になります。

戻り値

- LONG VARCHAR
- LONG BINARY

備考

数値または日付をパラメータとして指定した場合は、文字列に変換されてから連結されます。また、1 つの式を唯一のパラメータとして指定すると、STRING 関数を使用して、その式を文字列に変換できます。

すべてのパラメータが NULL の場合、STRING は NULL を返します。NULL でないパラメータが存在すると、NULL パラメータはすべて空の文字列として処理されます。

参照

- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 *testing123* を返します。

```
SELECT STRING( 'testing', NULL, 123 );
```

STRTOUUID 関数 [文字列]

文字列の値をユニークな識別子 (UUID または GUID) の値に変換します。

> **注意**
> バージョン 9.0.2 より前に作成されたデータベースでは、UUID 値のバイナリ表現と文字列表現の間を変換するための STRTOUUID 関数と UUIDTOSTR 関数が必要です。
>
> バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型がネイティブデータ型に変更されています。これらのバージョンでは STRTOUUID 関数と UUIDTOSTR 関数を使用する必要はありません。
>
> 詳細については、「Ultra Light SQL データ型」293 ページを参照してください。

構文

STRTOUUID(*string-expression* **)**

パラメータ

- ***string-expression*** フォーマットが *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の文字列。

戻り値

UNIQUEIDENTIFIER

備考

フォーマットが *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の文字列をユニークな識別子の値に変換します。ここで、*x* は 16 進数です。

この関数は、UUID 値をデータベースに挿入するときに役立ちます。

参照

- 「UUIDTOSTR 関数 [文字列]」411 ページ
- 「NEWID 関数 [その他]」374 ページ
- 「Ultra Light 文字列関数」319 ページ

STUFF 関数 [文字列]

1 つの文字列から複数の文字を削除して、別の文字列に置き換えます。

構文

STUFF(*string-expression-1*, *start*, *length*, *string-expression-2* **)**

パラメータ

- ***string-expression-1*** STUFF 関数によって変更される文字列。
- ***start*** 削除を開始する文字の位置。文字列の先頭文字の位置を 1 とします。
- ***length*** 削除する文字数。
- ***string-expression-2*** 挿入する文字列。STUFF 関数を使用して文字列の一部を削除するには、NULL の置換文字列を使用します。

戻り値

VARCHAR

参照

- 「INSERTSTR 関数 [文字列]」358 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 chocolate pie を返します。

```
SELECT STUFF( 'chocolate cake', 11, 4, 'pie' );
```

SUBSTRING 関数 [文字列]

文字列の部分文字列を返します。

構文

{ **SUBSTRING** | **SUBSTR** } **(** *string-expression*, *start*
[, *length*] **)**

パラメータ

- ***string-expression*** 部分文字列が返される文字列。
- ***start*** 文字単位で指定した、返される部分文字列の開始位置。
- ***length*** 文字単位で指定した、返される部分文字列の長さ。*length* を指定すると、部分文字列は指定した長さに限定されます。

戻り値

- LONG BINARY
- LONG VARCHAR

備考

Ultra Light では、データベースに ansi_substring オプションがありませんが、デフォルトでは、SUBSTR 関数は ansi_substring が ON に設定されているように動作します。関数の動作は ANSI/ISO SQL/2008 の動作に対応しています。

- **Start 値** 文字列の最初の文字の位置を 1 とします。負または 0 の開始オフセットは、文字列の左側が文字以外で埋められているように扱われます。
- **Length 値** 正の *length* は、部分文字列が開始位置の右側から *length* 文字で終わることを示します。

  負の *length* はエラーを返します。

  0 の *length* は、空の文字列を返します。

string-expression が binary データ型の場合、SUBSTRING 関数は BYTE_SUBSTR のように動作します。

文字列の末尾の文字を取得するには、RIGHT 関数を使用します。

入力文字が文字長のセマンティックを使用している場合、可能であれば、戻り値が文字長のセマンティックで記述されます。

参照

- 「BYTE_SUBSTR 関数 [文字列]」327 ページ
- 「LEFT 関数 [文字列]」361 ページ
- 「RIGHT 関数 [文字列]」385 ページ
- 「CHARINDEX 関数 [文字列]」332 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の表は、SUBSTRING 関数から返される値を示します。

例	結果
SUBSTRING('front yard', 1, 4)	fron
SUBSTRING('back yard', 6, 4)	yard
SUBSTR('abcdefgh', 0, -2)	エラーを返します
SUBSTR('abcdefgh', -2, 2)	空の文字列を返します

SUM 関数 [集合]

ローグループごとに、指定された式の合計を返します。

構文 1

SUM([DISTINCT] *expression* **)**

パラメータ

- ***expression*** 合計される式の名前。通常はカラム名です。
- **DISTINCT *expression*** 各グループの *expression* で一意の値の合計を計算します。

戻り値

- INTEGER
- DOUBLE
- NUMERIC

備考

指定された式が NULL のローは含まれません。

グループにローが含まれていない場合は、NULL を返します。

この関数ではオーバーフローエラーが発生することがあり、これにより、エラーが返される場合があります。オーバーフローエラーを回避するために、*numeric-expression* で CAST 関数を使用できます。「CAST 関数 [データ型変換]」328 ページを参照してください。

参照

- 「COUNT 関数 [集合]」338 ページ
- 「AVG 関数 [集合]」325 ページ

例

次の文は、値 3749146.740 を返します。

```
SELECT SUM( Salary )
FROM GROUPO.Employees;
```

SWITCHOFFSET 関数 [日付と時刻]

元のタイムゾーンオフセットから指定のタイムゾーンオフセットに変換される TIMESTAMP WITH TIME ZONE 値を返します。

構文

SWITCHOFFSET(*tmz-expression*, *time-zone-offset* **)**

パラメータ

- ***tmz-expression*** 変換される TIMESTAMP WITH TIME ZONE 値。
- ***time-zone-offset*** 結果のタイムゾーンオフセット。協定世界時 (UTC: Coordinated Universal Time) の前または後の分数を表す整数、{ + | - } hh:nn 形式の文字列、またはズールータイムゾーンを表す Z のいずれかをこの値として使用できます。ズールータイムゾーンは、UTC と同じタイムゾーンです。

戻り値

TIMESTAMP WITH TIME ZONE

参照

- 「TIMESTAMP WITH TIME ZONE データ型」『SQL Anywhere サーバ SQL リファレンス』
- 「SYSDATETIMEOFFSET 関数 [日付と時刻]」『SQL Anywhere サーバ SQL リファレンス』

例

次の例は、タイムゾーンオフセット値を -04:00 時から -07:00 時に変更します。戻り値は 2009-04-03 11:45:12.123-07:00 です。

```
SELECT CAST ( '2009-04-03 14:45:12.123-04:00' AS datetimeoffset ) AS EDT,
SWITCHOFFSET( EDT,'-07:00' ) AS PDT;
```

SYNC_PROFILE_OPTION_VALUE 関数 [システム]

指定されたオプション名に対応するオプションの値を返します。Ultra Light Java Edition データベースではサポートされていません。

構文

SYNC_PROFILE_OPTION_VALUE(*profile_name*, *option_name***)**

パラメータ

- ***profile_name*** 調べる同期プロファイルの名前。
- ***option_name*** 値の取得対象であるオプションの名前。

戻り値

指定されたオプション名に対応するオプションの値を返します。

備考

ピリオドを含むオプション名を指定すると、ピリオドの前の部分を基本オプション名、ピリオドの後ろの部分をサブリストオプション名として、サブリストから値を取得します。

参照

- 「ML_GET_SERVER_NOTIFICATION 関数 [システム]」370 ページ

例

次のプロファイルの場合、以下のようになります。

```
MobiLinkUid=joe;Stream=tcpip{host=sybase;port=1234};Ping=1
```

- **MobiLinkUid** joe
- **Stream** tcpip{host=sybase;port=1234}
- **Stream.host** sybase
- **Stream.port** 1234
- **Ping** 1

TAN 関数 [数値]

数のタンジェントを返します。

構文

TAN(*numeric-expression* **)**

パラメータ

- ***numeric-expression*** 角度 (ラジアン)。

戻り値

DOUBLE

備考

ATAN 関数と TAN 関数は逆変換の演算です。

この関数は、引数を DOUBLE に変換し、計算を倍精度浮動小数点で行い、結果を DOUBLE で返します。

参照

- 「COS 関数 [数値]」336 ページ
- 「SIN 関数 [数値]」391 ページ

例

次の文は、0.52 の tan 値を返します。

```
SELECT TAN( 0.52 );
```

TODATETIMEOFFSET 関数 [日付と時刻]

指定のタイムゾーンオフセットを使用して、TIMESTAMP 値を TIME STAMP WITH TIME ZONE 値に変換します。

構文

TODATETIMEOFFSET(*timestamp-expression*, *time-zone-offset* **)**

パラメータ

- ***timestamp-expression*** 変換される TIMESTAMP 式。
- ***time-zone-offset*** タイムゾーンオフセット。UTC の前または後の分数を表す INTEGER、{ + | - } hh:nn 形式の VARCHAR、またはズールータイムゾーンを表す文字列 "Z" のいずれかをこの値として使用できます。ズールータイムゾーンは、UTC と同じタイムゾーンです。

戻り値

TIMESTAMP WITH TIME ZONE

参照

- 「TIMESTAMP WITH TIME ZONE データ型」『SQL Anywhere サーバ SQL リファレンス』

例

次の例は、TIMESTAMP 値を TIMESTAMP WITH TIME ZONE 値に変換します。

```
SELECT CAST('2009-04-03 14:45:12.123' AS TIMESTAMP) AS orig,
    TODATETIMEOFFSET (orig,'+11:00');
```

TODAY 関数 [日付と時刻]

現在の日付を DATE 値で返します。

構文

TODAY([*])

戻り値

DATE

備考

セマンティック上、TODAY(*) と TODAY() は同義です。TODAY は CURRENT DATE 特別値と同義です。

要求に含まれる TODAY 関数の各インスタンスは、多くても 1 回のみ評価されます。同じ要求に含まれる TODAY の複数のインスタンスは、同じ DATE 値を共有する場合もあれば、そうでない場合もあります。

例

次の文は、システムクロックによる現在の日付を返します。

```
SELECT TODAY( * );
SELECT CURRENT DATE;
```

TRIM 関数 [文字列]

先行ブランクと後続ブランクを文字列から削除します。

構文

TRIM(*string-expression* **)**

パラメータ

- ***string-expression*** 削除する文字列。

戻り値

- VARCHAR
- LONG VARCHAR

参照

- 「LTRIM 関数 [文字列]」366 ページ
- 「RTRIM 関数 [文字列]」387 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、先行ブランクと後続ブランクなしの値 chocolate を返します。

```
SELECT TRIM( '  chocolate  ' );
```

TRUNCNUM 関数 [数値]

指定した桁数で小数点以下を切り捨てます。

構文

{ **TRUNCNUM** | **TRUNCATE** }**(** *numeric-expression*, *integer-expression* **)**

パラメータ

- ***numeric-expression*** トランケートされる数。この引数には、NUMERIC または DOUBLE データ型を使用できます。

- ***integer-expression*** 正の整数は、丸めを行う小数点の右側の有効桁数を指定します。負の値は、丸めを行う小数点の左側の有効桁数を指定します。

戻り値

NUMERIC または DOUBLE

備考

いずれかのパラメータが NULL 値の場合、結果は NULL 値になります。

数字をトランケートする場合、TRUNCATE 関数ではなく TRUNCNUM を使用します。

TRUNCATE 関数の使用はおすすめしません。truncate という単語はキーワードなので、quoted_identifier オプションを OFF に設定するか、単語 TRUNCATE を引用符で囲む必要があるためです。

参照

● 「ROUND 関数 [数値]」386 ページ

例

次の文は、値 600 を返します。

```
SELECT TRUNCNUM( 655, -2 );
```

次の文は、値 655.340 を返します。

```
SELECT TRUNCNUM( 655.348, 2 );
```

UCASE 関数 [文字列]

文字列中のすべての文字を大文字に変換します。

構文

UCASE(*string-expression* **)**

パラメータ

● ***string-expression*** 大文字に変換される文字列。

戻り値

引数のデータ型に対応する CHAR、VARCHAR、または LONG VARCHAR。

備考

この関数は UPPER 関数と同じです。

参照

● 「UPPER 関数 [文字列]」411 ページ
● 「LCASE 関数 [文字列]」360 ページ
● 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 CHOCOLATE を返します。

```
SELECT UCASE( 'ChocoLate' );
```

UPPER 関数 [文字列]

文字列中のすべての文字を大文字に変換します。

構文

UPPER(*string-expression* **)**

パラメータ

- ***string-expression*** 大文字に変換される文字列。

戻り値

引数のデータ型に対応する CHAR、VARCHAR、または LONG VARCHAR。

備考

この関数は UCASE 関数と同じです。

参照

- 「UCASE 関数 [文字列]」410 ページ
- 「LCASE 関数 [文字列]」360 ページ
- 「LOWER 関数 [文字列]」366 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、値 CHOCOLATE を返します。

```
SELECT UPPER( 'ChocoLate' );
```

UUIDTOSTR 関数 [文字列]

ユニークな識別子の値 (UUID または GUID) を文字列の値に変換します。

> **注意**
> バージョン 9.0.2 より前に作成されたデータベースでは、UUID 値のバイナリ表現と文字列表現の間を変換するための STRTOUUID 関数と UUIDTOSTR 関数が必要です。
>
> バージョン 9.0.2 以降を使用して作成されたデータベースでは、UNIQUEIDENTIFIER データ型がネイティブデータ型に変更されています。これらのバージョンでは STRTOUUID 関数と UUIDTOSTR 関数を使用する必要はありません。
>
> 詳細については、「Ultra Light SQL データ型」293 ページを参照してください。

構文

UUIDTOSTR(*uuid-expression* **)**

パラメータ

- ***uuid-expression*** ユニークな識別子の値。

戻り値

VARCHAR

備考

ユニークな識別子を、フォーマットが *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx* の文字列値に変換します。ここで、x は 16 進数です。バイナリ値が有効な uniqueidentifier でない場合は、NULL を返します。

この関数は、UUID 値を表示するのに便利です。

参照

- 「NEWID 関数 [その他]」374 ページ
- 「STRTOUUID 関数 [文字列]」402 ページ
- 「Ultra Light 文字列関数」319 ページ

例

次の文は、2 つのカラムを持つテーブル mytab を作成します。カラム pk は uniqueidentifier データ型で、カラム c1 は integer データ型です。それぞれに値 1 と値 2 を持つ 2 つのローをカラム c1 に挿入します。

```
CREATE TABLE mytab(
   pk UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
   c1 INT );
INSERT INTO mytab( c1 ) values ( 1 );
INSERT INTO mytab( c1 ) values ( 2 );
```

次の SELECT 文を実行すると、新規に作成したテーブルのすべてのデータを返します。

```
SELECT * FROM mytab;
```

2 つのカラムと 2 つのローを持ったテーブルが表示されます。カラム pk に表示される値は、バイナリ値です。

ユニークな識別子の値を読みやすい形式に変換するには、次の文を実行します。

```
SELECT UUIDTOSTR(pk), c1 FROM mytab;
```

UUIDTOSTR 関数は、バージョン 9.0.2 以降で作成されたデータベースでは不要です。

WEEKS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の週の数を返します。次の「備考」の項を参照してください。

構文 1

WEEKS(*timestamp-expression* **)**

構文 2

WEEKS(*timestamp-expression*, *timestamp-expression* **)**

構文 3

WEEKS(*timestamp-expression*, *integer-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP データ型の値の日付と時刻。
- ***integer-expression*** *timestamp-expression* に加算される週数。*integer-expression* が負の場合、*timestamp-expression.* から適切な週数が引かれます。*integer-expression* を指定する場合は、*timestamp-expression* を DATE または TIMESTAMP として明示的にキャストしてください。

戻り値

構文 1 または構文 2 では INTEGER。

構文 3 では TIMESTAMP。

備考

1 つの日付を指定すると (構文 1)、WEEKS 関数は、0000-02-29 からの週数を返します。

2 つの日付を指定すると (構文 2)、WEEKS 関数は、2 つの日付の間の週数を返します。WEEKS 関数は DATEDIFF 関数に似ていますが､2 つの日付間の週数を計算する場合に使用される方法はこれと同じではなく、返される結果が異なる場合があります。WEEKS の戻り値は、2 つの日付の間の日数を 7 で割り、余りを切り捨てて求められます。ただし、DATEDIFF では週の境目の数が使用されます。このため、2 つの関数から返される値が異なる場合があります。たとえば、最初の日付が金曜日で次の日付が翌月曜日の場合、WEEKS 関数は週数として 0 を返しますが、DATEDIFF 関数は 1 を返します。どちらかが優れているという問題ではないので、WEEKS と DATEDIFF との動作の違いをふまえて使用してください。

DATEDIFF 関数の詳細については、「DATEDIFF 関数 [日付と時刻]」342 ページを参照してください。

1 つの日付と整数を指定すると (構文 3)、WEEKS 関数は、*timestamp-expression* に、指定された整数の週数を加算します。構文 3 では、*timestamp-expression* を TIME、DATE、または TIMESTAMP データ型として明示的にキャストしてください。*timestamp-expression* が TIME 値の場合、現在の日付が使用されます。構文 3 の代わりに、DATEADD 関数を使用します。

DATEADD 関数の詳細については、「DATEADD 関数 [日付と時刻]」341 ページを参照してください。

参照

データ型のキャストの詳細については、「CAST 関数 [データ型変換]」328 ページを参照してください。

例

次の文は、値 8 を返します。これは、2008-09-13 10:07:12 が、2008-07-13 06:07:12 の 8 週間後であることを示します。

```
SELECT WEEKS( '2008-07-13 06:07:12', '2008-09-13 10:07:12' );
```

次の文は、値 104792 を返します。これは、指定された日付が、0000-02-29 の 104792 週間後であることを示します。

```
SELECT WEEKS( '2008-07-13 06:07:12' );
```

次の文は、TIMESTAMP 値 2008-06-16 21:05:07.0 を返します。これは、返された日時が 2008-05-12 21:05:07 の 5 週間後であることを示します。

```
SELECT WEEKS( CAST( '2008-05-12 21:05:07' AS TIMESTAMP ), 5);
```

YEAR 関数 [日付と時刻]

TIMESTAMP 引数の年部分を返します。

構文

YEAR(*timestamp-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP 値。

戻り値

SMALLINT

備考

指定された TIMESTAMP 値の年部分が SMALLINT として返されます。

例

次の例は、値 2001 を返します。

```
SELECT YEAR( '2001-09-12' );
```

YEARS 関数 [日付と時刻]

TIMESTAMP を操作するか、2 つの TIMESTAMP 値の間の年数を返します。次の「備考」の項を参照してください。

構文 1

YEARS(*timestamp-expression* **)**

構文 2

YEARS(*timestamp-expression*, *timestamp-expression* **)**

構文 3

YEARS(*timestamp-expression*, *integer-expression* **)**

パラメータ

- ***timestamp-expression*** TIMESTAMP データ型の値の日付と時刻。
- ***integer-expression*** *timestamp-expression* に (SMALLINT 値として) 追加される年数。*integer-expression* が負の場合、*timestamp-expression.* から適切な年数が引かれます。*integer-expression* を指定する場合は、*timestamp-expression* を DATE、TIME、または TIMESTAMP 値として明示的にキャストしてください。*timestamp-expression* が TIME 値の場合、現在の年が使用されます。

  データ型のキャストの詳細については、「CAST 関数 [データ型変換]」328 ページを参照してください。

戻り値

構文 1 または構文 2 では SMALLINT。

構文 3 では TIMESTAMP。

備考

YEARS の値は、2 つの日付の間に年の最初の日がいくつあるかをカウントして求められます。

参照

- 「DATEDIFF 関数 [日付と時刻]」342 ページ
- 「DATEADD 関数 [日付と時刻]」341 ページ

例

次の文はどちらも -4 を返します。

```
SELECT YEARS( '1998-07-13 06:07:12',
              '1994-03-13 08:07:13' );

SELECT DATEDIFF( year,
  '1998-07-13 06:07:12',
  '1994-03-13 08:07:13' );
```

次の文は、1998 を返します。

```
SELECT YEARS( '1998-07-13 06:07:12' )
SELECT DATEPART( year, '1998-07-13 06:07:12' );
```

次の文は、指定した日付の 300 年後を返します。

```
SELECT YEARS( CAST( '1998-07-13 06:07:12' AS TIMESTAMP ), 300 )

SELECT DATEADD( year, 300, '1998-07-13 06:07:12' );
```

YMD 関数 [日付と時刻]

指定した年、月、日に相当する日付値を返します。引数は -32768 ～ 32767 の SMALLINT です。

構文

YMD(*smallint-expression1*, *smallint-expression2*, *smallint-expression3* **)**

パラメータ

- ***smallint-expression1*** 年。
- ***smallint-expression2*** 月の数。月が 1 ～ 12 の範囲外である場合は、年が相応に調整されます。
- ***smallint-expression3*** 日の数。任意の整数を指定でき、それに応じて日付が調整されます。

戻り値

DATE

例

次の文は、値 1998-06-12 を返します。

```
SELECT YMD( 1998, 06, 12 );
```

値が通常の範囲から外れている場合、それに応じて日付が調整されます。たとえば、次の文は DATE 値 2000-03-01 を返します。

```
SELECT YMD( 1999, 15, 1 );
```

Ultra Light SQL 文

Ultra Light SQL でサポートされている SQL 文は、SQL Anywhere データベースでサポートされている文のサブセットです。

始める前に

- Ultra Light のテーブルでは、所有者の概念がサポートされていません。既存の SQL と、プログラムで生成される SQL のために、Ultra Light では *owner.table-name* の構文が許可されます。ただし、テーブル所有者は Ultra Light ではサポートされていないので、所有者はチェックされません。
- Ultra Light SQL 文のマニュアルは、SQL Anywhere の文で使用されている構文と同じ表記規則に従っています。この表記規則について、またこの表記規則を使用して SQL 構文が表されている方法を理解している必要があります。
- Ultra Light SQL を使用するとトランザクションが作成されます。トランザクションは、最後の ROLLBACK または COMMIT 文以降のすべての変更内容 (INSERT、UPDATE、DELETE 文) で構成されます。

  変更内容は、COMMIT を実行することによって確定できます。ROLLBACK 文を実行すると、変更内容が削除されます。

参照

- 「SQL 文」『SQL Anywhere サーバ SQL リファレンス』
- 「SQL 構文の表記規則」『SQL Anywhere サーバ SQL リファレンス』
- 「Ultra Light でのトランザクション処理」477 ページ
- 「COMMIT 文 [Ultra Light]」427 ページ
- 「ROLLBACK 文 [Ultra Light]」450 ページ

Ultra Light の文のカテゴリ

SQL 文は、文の最初の語で分類されています。この語はほとんど常に動詞です。このアクション指向型の構文では、言語の性質が、データベースに対する一連の命令文 (コマンド) になります。Ultra Light でサポートされている SQL 文は次のように分類できます。

- **データ検索文**　クエリとも呼ばれます。これらの文は、テーブルからデータ式のローを選択できます。データ検索は SELECT 文を使用して行います。

- **データ操作文**　データベースの内容を変更できます。データ操作は、次の文を使用して実行します。

  - INSERT
  - UPDATE
  - DELETE

- **データ定義文**　データベースの構造またはスキーマを定義できます。スキーマの変更には次の文を使用します。

  - ALTER DATABASE SCHEMA FROM FILE
  - CREATE INDEX
  - CREATE TABLE
  - DROP INDEX
  - DROP TABLE
  - ALTER TABLE
  - TRUNCATE TABLE

- **トランザクション制御文**　Ultra Light アプリケーション内でトランザクションを制御できます。トランザクション制御は、次の文を使用して実行します。

  - CHECKPOINT
  - COMMIT
  - ROLLBACK

- **同期管理**　Mobile Link サーバとの同期を一時的に制御できます。同期管理には次の文を使用します。
  - START SYNCHRONIZATION DELETE
  - STOP SYNCHRONIZATION DELETE
  - CREATE PUBLICATION
  - ALTER PUBLICATION
  - DROP PUBLICATION

参照

- 「Ultra Light の式」275 ページ
- 「Ultra Light の演算子」290 ページ
- 「SELECT 文 [Ultra Light]」451 ページ
- 「INSERT 文 [Ultra Light]」
- 「UPDATE 文 [Ultra Light]」
- 「DELETE 文 [Ultra Light]」
- 「ALTER DATABASE SCHEMA FROM FILE 文 [Ultra Light]」
- 「CREATE INDEX 文 [Ultra Light]」
- 「CREATE TABLE 文 [Ultra Light]」
- 「DROP INDEX 文 [Ultra Light]」
- 「DROP TABLE 文 [Ultra Light]」
- 「ALTER TABLE 文 [Ultra Light]」
- 「TRUNCATE TABLE 文 [Ultra Light]」
- 「CHECKPOINT 文 [Ultra Light]」
- 「COMMIT 文 [Ultra Light]」
- 「ROLLBACK 文 [Ultra Light]」
- 「START SYNCHRONIZATION DELETE 文 [Ultra Light]」
- 「STOP SYNCHRONIZATION DELETE 文 [Ultra Light]」
- 「CREATE PUBLICATION 文 [Ultra Light]」
- 「ALTER PUBLICATION 文 [Ultra Light]」
- 「DROP PUBLICATION 文 [Ultra Light]」

ALTER DATABASE SCHEMA FROM FILE 文 [Ultra Light]

SQL スクリプトを使用して既存の Ultra Light データベースのスキーマ定義を変更します。

> **注意**
> この文は、Ultra Light Java Edition データベースではサポートされていません。

構文

ALTER DATABASE SCHEMA FROM FILE *filename*

パラメータ

filename　既存の Ultra Light データベースのスキーマをアップグレードするときに使用する SQL スクリプトの名前とパスを定義します。

備考

スクリプトに必要な DDL 文を抽出するには、ulinit または ulunload のいずれかを使用します。これらのユーティリティを使用して、DDL 文が構文的に正しいことを確認してください。ulinit (-l *ログファイル* オプション) または ulunload (-n -s *出力ファイル* オプション) を使用します。

この文は、データベースのバックアップを作成した後に実行します。

SQL スクリプトファイルの文字セットは、アップグレード対象のデータベースの文字セットに一致している必要があります。

この文の実行中は、デバイスをリセットしないでください。スキーマのアップグレード中にデバイスをリセットすると、Ultra Light データベースは使用できなくなります。

スキーマに適合しないローはすべて削除されます (たとえば、一意性制約が追加され、複数のローに同じ値が入っている場合、1 つのローだけを残して他のローはすべて削除されます)。この場合、SQLE_ROW_DROPPED_DURING_SCHEMA_UPGRADE 警告が生成されます。この警告によってエラーを検出し、データベースをバックアップからリストアできます。

参照

- 「Ultra Light データベースのスキーマ」49 ページ
- 「Ultra Light データベーススキーマのアップグレードの配備」122 ページ
- 「Ultra Light データベース初期化ユーティリティ (ulinit)」211 ページ
- 「Ultra Light データベースのアンロードユーティリティ (ulunload)」232 ページ

例

次の文は、SQL スクリプト *MySchema.sql* を使用して、データベースのスキーマを修正します。

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

ALTER PUBLICATION 文 [Ultra Light]

パブリケーションを変更します。

構文

ALTER PUBLICATION *publication-name alterpub-clause*

alterpub-clause :
ADD TABLE *table-name* [**WHERE** *search-condition*]
| **ALTER TABLE** *table-name* [**WHERE** *search-condition*]
| { **DROP** | **DELETE** } **TABLE** *table-name*
| **RENAME** *publication-name*

備考

パブリケーションは、同期対象のリモートデータベース内のデータを識別します。

関連する動作

オートコミット。

参照

- 「Ultra Light の探索条件」281 ページ
- 「Ultra Light クライアントの同期設計」74 ページ
- 「CREATE PUBLICATION 文 [Ultra Light]」430 ページ
- 「DROP PUBLICATION 文 [Ultra Light]」441 ページ
- 「START SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ
- 「STOP SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ

例

次の ALTER PUBLICATION 文は、Customer テーブルを pub_contact パブリケーションに追加します。

```
ALTER PUBLICATION pub_contact
  ADD TABLE Customers
```

ALTER SYNCHRONIZATION PROFILE 文 [Ultra Light]

Ultra Light の同期プロファイルを変更します。

構文

ALTER SYNCHRONIZATION PROFILE *sync-profile-name*
MERGE *sync-option* [; ...]

sync-option :
sync-option-name **=** *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

パラメータ

- **sync-profile-name**　同期プロファイルの名前。
- **MERGE 句**　この句を使用して、既存のオプションを変更したり、同期プロファイルに新しいオプションを追加したりします。
- **sync-option**　1 つ以上の同期オプションの値ペアがセミコロンで区切られた文字列。たとえば、'option1=value1;option2=value2' のように記述します。
- **sync-option-name**　同期プロファイルオプションの名前。
- **sync-option-value**　同期プロファイルオプションの値。

備考

同期プロファイルによって、Ultra Light データベースが Mobile Link サーバと同期する方法を定義します。

MERGE 句を使用して、既存の同期プロファイルを変更できます。この句を使用すると、MERGE 句で指定した同期オプションだけが変更されます。同期プロファイルから同期オプションを削

除するには、'option1=;' のような sync-option 文字列を作成します (オプションの値を空に設定します)。

同期プロファイルオプション STREAM は、その値にサブリストが含まれているため他のオプションとは異なります。たとえば、'STREAM=TCPIP{host=192.168.1.1;port=1234}' とあるとします。この例では、'host=192.168.1.1;port=1234' がサブリストです。サブリストの値を追加または削除するには、STREAM の sync-option-name と sub-option-name の間にピリオドを使用します。たとえば、MERGE 'stream.port=5678;stream.host=;compression=zlib' のようにすると、同期プロファイルは stream=TCPIP{port=5678;compression=zlib} になります。ストリームを新しい値に設定しようとすると、ストリーム全体の値が置換されます。たとえば、MERGE 'stream=HTTPS' のようにすると、同期プロファイルは stream=HTTPS{} になります。

関連する動作

なし。

参照

- 「同期プロファイルオプション」227 ページ
- 「DROP SYNCHRONIZATION PROFILE 文 [Ultra Light]」441 ページ
- 「SYNCHRONIZE 文 [Ultra Light]」455 ページ

例

次の例は、ALTER SYNCHRONIZATION PROFILE...REPLACE 文を示します。

```
CREATE SYNCHRONIZATION PROFILE myProfile1;
ALTER SYNCHRONIZATION PROFILE myProfile1
  REPLACE 'publications=p1;uploadonly=on';
```

次の例は、ALTER SYNCHRONIZATION PROFILE...MERGE 文を示します。

```
CREATE SYNCHRONIZATION PROFILE myProfile2 'publications=p1;
ALTER SYNCHRONIZATION PROFILE myProfile2
  MERGE 'publications=p2;uploadonly=on';
```

次の例は、別のオプションで一連の ALTER SYNCHRONIZATION PROFILE コマンドを実行した後に発生する変更を示します。

myProfile1='MobiLinkUID=mary;ScriptVersion=default' とします。

ALTER SYNCHRONIZATION PROFILE myProfile1 REPLACE 'MobiLinkPwd=sql;ScriptVersion=1' を実行した後、myProfile1 は 'MobiLinkPwd=sql;ScriptVersion=1' になります。

ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'MobiLinkUID=mary;STREAM=tcpip' を実行した後、myProfile1 は 'MobiLinkPwd=sql;ScriptVersion=1;MobiLinkUID=mary;STREAM=tcpip' になります。

ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'MobiLinkUID=;STREAM.host=192.168.1.1;STREAM.port=1234;ScriptVersion=;' を実行した後、myProfile1 は 'MobiLinkPwd=sql;STREAM=tcpip{192.168.1.1;port=1234}' になります。

ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'MobiLinkPwd=;Ping=yes;STREAM=HTTP' を実行した後、myProfile1 は 'Ping=yes;STREAM=HTTP' になります。

ALTER SYNCHRONIZATION PROFILE myProfile1 MERGE 'STREAM=HTTP{host=192.168.1.1}' を実行した後、myProfile1 は 'Ping=yes;STREAM=HTTP{host=192.168.1.1}' になります。

ALTER TABLE 文 [Ultra Light]

テーブル定義を変更します。

構文

ALTER TABLE *table-name* {
add-clause
| *modify-clause*
| *drop-clause*
| *rename-clause*
}

add-clause :
ADD { *column-definition* | *table-constraint* }

modify-clause :
ALTER *column-definition* | *sync-constraint*

drop-clause :
DROP { *column-name* | **CONSTRAINT** *constraint-name* }

rename-clause :
RENAME {
new-table-name
| [*old-column-name* **TO**] *new-column-name*
| **CONSTRAINT** *old-constraint-name* **TO** *new-constraint-name* }

column-definition :
column-name data-type
[[**NOT**] **NULL**]
[**DEFAULT** *column-default*]
[**UNIQUE**]

column-default :
GLOBAL AUTOINCREMENT [(*number*)]
| **AUTOINCREMENT**
| **CURRENT DATE**
| **CURRENT TIME**
| **CURRENT TIMESTAMP**
| **NULL**
| **NEWID( )**
| *constant-value*

table-constraint :
[**CONSTRAINT** *constraint-name*]

 Copyright © 2013, SAP AG or an SAP affiliate company. - SAP Sybase SQL Anywhere 16.0

{ *fkey-constraint* | *unique-key-constraint* }
[**WITH MAX HASH SIZE** *integer*]

fkey-constraint :
[**NOT NULL**] **FOREIGN KEY** [*role-name*] **(** *ordered-column-list* **)**
REFERENCES *table-name* **(** *column-name*, ... **)**
[**CHECK ON COMMIT**]

unique-key-constraint :
UNIQUE (*ordered-column-list* **)**

ordered-column-list :
(*column-name* [**ASC** | **DESC**], ... **)**

sync-constraint :**SYNCHRONIZE** {**ON**| **OFF**|**ALL**|**DOWNLOAD**}

パラメータ

add-clause　テーブルに新しいカラム制約またはテーブル制約を追加します。

テーブルに新しいカラム制約またはテーブル制約を追加します。

○ **ADD *column-definition* 句**　テーブルに新しいカラムを追加します。カラムにデフォルト値がある場合は、新しいカラムのすべてのローにそのデフォルト値が移植されます。 この句のキーワードとサブ句の詳細については、「CREATE TABLE 文 [Ultra Light]」432 ページを参照してください。

○ **ADD *table-constraint* 句**　テーブルに制約を追加します。オプションの制約名を使用すると、テーブル全体の制約を修正することなく、後で個々の制約を修正したり削除したりできます。この句のキーワードとサブ句の詳細については、「CREATE TABLE 文 [Ultra Light]」432 ページを参照してください。

新しいユニークな制約を追加する場合は、すべての制約カラムを非 NULL 可にする必要があります。ユニークな制約を追加するには、カラムを NOT NULL に変更してください。

> **注意**
> Ultra Light では、プライマリキーを追加することはできません。

modify-clause　1 つのカラム制約を変更します。*column-definition* が ALTER 文の一部の場合には、プライマリキーは使用できません。必要に応じて、変更されるカラムのデータを新しいデータ型に変換します。変換エラーが発生すると、操作は失敗となり、テーブルは変更されません。*column-definition* の詳細については、「CREATE TABLE 文 [Ultra Light]」432 ページを参照してください。

drop-clause　カラム制約またはテーブル制約を削除します。

○ **DROP *column-name***　テーブルからカラムを削除します。カラムがインデックス、一意性制約、外部キー、またはプライマリキーに含まれている場合は、それらのオブジェクトを削除**しないと**、Ultra Light がカラムを削除できません。

○ **DROP CONSTRAINT *table-constraint*** テーブル定義から指定された制約を削除します。*table-constraint* の詳細については、「CREATE TABLE 文 [Ultra Light]」432 ページを参照してください。

> **注意**
> Ultra Light では、プライマリキーを削除することはできません。

rename-clause テーブル名、カラム名、制約名を変更します。

○ **RENAME *new-table-name*** テーブルの名前を *new-table-name* に変更します。古いテーブル名を使用しているアプリケーションは、修正する必要があります。古いテーブル名を自動的に割り当てられた外部キーは、名前を変更しません。

○ **RENAME *old-column-name* TO *new-column-name*** カラムの名前を *new-column-name* に変更します。古いカラム名を使用しているアプリケーションは、修正する必要があります。

○ **RENAME *old-constraint-name* TO *new-constraint-name*** 制約の名前を *new-constraint-name* に変更します。古い定数名を使用しているアプリケーションは、修正する必要があります。

> **注意**
> Ultra Light では、プライマリキーの名前を変更することはできません。

column-constraint カラム制約は、データベース内のデータの整合性を保つために、カラムに格納できる値を制限します。カラム制約は UNIQUE である必要があります。

UNIQUE テーブル内の各ローをユニークに識別する 1 つまたは複数のカラムを識別します。テーブル内の異なるローが、指定されているすべてのカラムで同じ値を持つことはできません。1 つのテーブルに複数の一意性制約が存在することがあります。

sync-constraint 句 テーブルを同期可能にするかどうかおよびすべてのローをアップロードするか、テーブルへの変更だけをアップロードするか、テーブルへの変更をアップロードしないかを決定するための同期制約を指定します。

○ **SYNCHRONIZE ON** デフォルト設定 - テーブルを同期することができ、テーブルへの変更のみがアップロードで送信されます。

○ **SYNCHRONIZE OFF** テーブルを同期できず、テーブルをパブリケーションに含めるときにエラーが発生します。

○ **SYNCHRONIZE ALL** テーブルを同期することができ、テーブル内のすべてのローがアップロードで送信されます。この制約は、Ultra Light Java Edition データベースではサポートされていません。

○ **SYNCHRONIZE DOWNLOAD** テーブルを統合データベースへの変更と同期できますが、ローカルの変更はアップロードされません。

備考

1 つの ALTER TABLE 文では、1 つの *table-constraint* または *column-constraint* だけを追加、変更、または削除できます。

役割名は外部キーの名前です。*role-name* の主な機能は、同じテーブルに対する 2 つの外部キーを区別することです。また、CONSTRAINT *constraint-name* を使用して外部キーに名前を付けることができます。ただし、両方のメソッドを使用して外部キーに名前を付けないでください。

テーブルまたはカラムの制約は MODIFY (変更) できません。制約を変更するには、古い制約を DELETE (削除) し、新しい制約を ADD (追加) します。

テーブルを対象とした文が、別の要求やクエリですでに参照されている場合、ALTER TABLE は実行できません。同様に、テーブルの変更中は、そのテーブルを参照する要求は処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は ALTER TABLE を実行できません。

Ultra Light.NET の場合 : すべてのデータオブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。ULBulkCopy.Dispose メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』を参照してください。

スキーマの変更が同時に開始された場合、文は解放されません。

参照

- 「Ultra Light データベースのスキーマ」49 ページ
- 「CREATE TABLE 文 [Ultra Light]」432 ページ
- 「DROP TABLE 文 [Ultra Light]」442 ページ
- 「Ultra Light SQL データ型」293 ページ
- 「テーブル変更」『SQL Anywhere サーバ SQL の使用法』
- 「テーブルとカラム制約」『SQL Anywhere サーバ SQL の使用法』
- 「分割サイズ」73 ページ
- 「最後に割り当てられた GLOBAL AUTOINCREMENT 値の調べ方」72 ページ

例

次の文は、MyEmployees という架空のテーブルから Street カラムを削除します。

```
ALTER TABLE MyEmployees
DROP Street
```

次の例は、MyCustomers という架空のテーブルの Street カラムを変更して、約 50 文字を格納するようにします。

```
ALTER TABLE MyCustomers
ALTER Street CHAR(50)
```

ALTER USER 文 [Ultra Light]

ユーザ設定を変更します。

> **注意**
> この文は、Ultra Light Java Edition データベースではサポートされていません。

構文 1

ALTER USER *user-name* [**IDENTIFIED BY** *password*]

パラメータ

user-name　ユーザの名前。

IDENTIFIED BY 句　ユーザのパスワード。

備考

●ユーザ ID には、次に該当する値を指定できません。

○ 空白スペース、一重引用符、または二重引用符で始まる値
○ 空白スペースで終わる値
○ セミコロンを含む値

●パスワードは大文字と小文字が区別され、次に該当する値は指定できません。

○ 空白スペース、一重引用符、または二重引用符で始まる値
○ 空白スペースで終わる値
○ セミコロンを含む値
○ 255 バイト長を超える値

関連する動作

なし。

参照

● 「CREATE USER 文 [Ultra Light]」439 ページ
● 「DROP USER 文 [Ultra Light]」443 ページ

例

次の文は、ユーザ SQLTester を変更します。パスワードは、"welcome" に設定されます。

```
ALTER USER SQLTester IDENTIFIED BY welcome
```

CHECKPOINT 文 [Ultra Light]

データベースにチェックポイントを設定します。

> **注意**
> この文は、Ultra Light Java Edition データベースではサポートされていません。

構文

CHECKPOINT

備考

CHECKPOINT 文をコミットフラッシュのトリガとして使用できます。コミットフラッシュが実行されると、コミットされていないトランザクションが記憶領域に書き込まれます。

Embedded SQL API を使用している場合は、ULCheckpoint メソッドを使用することもできます。C++ コンポーネントアプリケーションを作成している場合は、接続オブジェクトで Checkpoint

メソッドを使用することもできます。その他すべての API では、この文を使用する必要があります。

関連する動作

この文によって、保留中のコミットされたトランザクションが記憶領域にすべてフラッシュされますが、現在のトランザクションはコミットまたはフラッシュされません。

参照

- 「単一のトランザクションまたはグループ化されたトランザクションのフラッシュ」478 ページ
- 「COMMIT 文 [Ultra Light]」427 ページ
- 「Ultra Light COMMIT_FLUSH 接続パラメータ」171 ページ
- ULCheckpoint メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.Checkpoint メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』

例

次の文は、データベースのチェックポイントを実行します。

```
CHECKPOINT
```

COMMIT 文 [Ultra Light]

データベースへの変更を永続的なものします。

構文

COMMIT [WORK]

備考

Ultra Light SQL を使用するとトランザクションが作成されます。トランザクションは、最後の ROLLBACK または COMMIT 後のすべての変更内容 (INSERT、UPDATE、DELETE) から構成されます。COMMIT 文は、現在のトランザクションを終了し、このトランザクション中に行われたすべての変更内容をデータベースに永続化します。

ALTER、CREATE、DROP の各文を使用したデータベースオブジェクトの変更は、自動的にコミットされます。

参照

- 「CHECKPOINT 文 [Ultra Light]」426 ページ
- 「ROLLBACK 文 [Ultra Light]」450 ページ

例

次の文は、現在のトランザクションで行われた変更内容をデータベースに永続化します。

```
COMMIT
```

CREATE INDEX 文 [Ultra Light]

指定したテーブルにインデックスを作成します。

構文

CREATE [**UNIQUE**] **INDEX** [**IF NOT EXISTS**] [*index-name*]
ON *table-name* **(** *ordered-column-list* **)**
[**WITH MAX HASH SIZE** *integer*]

ordered-column-list :
(*column-name* [**ASC** | **DESC**], ... **)**

パラメータ

UNIQUE UNIQUE 属性は、テーブル内の 2 つのローがインデックス内のすべてのカラムで同じ値にならないようにします。各インデックスキーはユニークであるか、少なくとも 1 つのカラムで NULL を持つ必要があります。

テーブル上の一意性制約とユニークインデックスの間には違いがあります。ユニークインデックスのカラムは NULL を使用できますが、一意性制約のカラムには使用できません。外部キーは、プライマリキーまたは一意性制約は参照できますが、ユニークインデックスは参照できません。ユニークインデックスは NULL の複数のインスタンスを含むことがあるからです。

一意性制約内のカラムが更新時に変更され、その一意性制約が外部キーによって参照されている場合、その一意性制約内のローを今後参照しないローは、すべてリモートから削除されます。

IF NOT EXISTS 句 IF NOT EXISTS 属性を指定し、指定したインデックスがすでに存在する場合、変更は行われず、エラーは返されません。

ordered-column-list カラムの順序リストです。インデックスのカラム値は、昇順または降順にソートできません。

WITH MAX HASH SIZE このインデックスのハッシュサイズをバイト単位で設定します。この値は、データベースに対して有効なデフォルトの MaxHashSize プロパティを上書きします。 デフォルトファイルの詳細については、「データベースプロパティの読み取り」39 ページを参照してください。

備考

Ultra Light では、プライマリキーと一意性制約のインデックスは自動的に作成されます。

インデックスによって、Ultra Light が特定のローを検索しやすくし、クエリのパフォーマンスを向上させることができます。ただし、インデックスを管理する必要があるので、同期や INSERT、DELETE、UPDATE の各文の処理は低速になる可能性があります。

インデックスは、データベースに対して発行されたクエリのパフォーマンスを向上させたり、ORDER BY 句を使用してクエリをソートするときに自動的に使用されます。インデックスはいったん作成されると、DROP INDEX を使用して削除するときを除いて、SQL 文では二度と参照されません。

インデックスはデータベース内の領域を占有します。また、インデックスの管理に必要な追加作業は、データ修正操作のパフォーマンスに影響する場合があります。このため、クエリのパフォーマンスが向上しないインデックスの作成は避けてください。

CREATE INDEX 文の処理中は、そのインデックスを参照する要求やクエリは処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は CREATE INDEX を実行できません。

Ultra Light では、実行プランを使用してクエリを最適化することもできます。「Ultra Light の実行プラン」469 ページを参照してください。

Ultra Light.NET の場合 : すべてのデータオブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。ULBulkCopy.Dispose メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』を参照してください。

スキーマの変更が同時に開始された場合、文は解放されません。「Ultra Light データベースのスキーマ」49 ページを参照してください。

関連する動作

- オートコミット

参照

- 「Ultra Light のパフォーマンスに関するヒント」461 ページ
- 「DROP INDEX 文 [Ultra Light]」440 ページ
- 「Ultra Light max_hash_size 作成パラメータ」147 ページ
- 「Ultra Light インデックス」56 ページ

例

次の文は、Employees テーブルで 2 カラムのインデックスを作成します。

```
CREATE INDEX employee_name_index
ON Employees ( Surname, GivenName )
```

次の文は、ProductID カラム用に SalesOrderItems テーブル上でインデックスを作成します。

```
CREATE INDEX item_prod
ON SalesOrderItems ( ProductID )
```

次に、Ultra Light Java Edition データベースに対する MAX HASH SIZE の影響を示すシナリオを示します。Employees テーブルの Initials カラムは VARCHAR(3)、EmployeeID カラムは TINY です。

次の文は、ASCII7 文字のみが使用されると、すべての値を完全にハッシュします。

```
CREATE INDEX ascii_a ON Employees( Initials ) WITH MAX HASH SIZE 3
```

次の文は、使用される文字に関係なく、すべての値を完全にハッシュします。

```
CREATE INDEX unicode_a ON Employees( Initials ) WITH MAX HASH SIZE 9
```

次の文は、ASCII のみが使用されていても、Initials の値しかハッシュしません。Initials の最初の 9 バイトは予約されているためです。

```
CREATE INDEX compound_1 ON Employees( Initials, EmployeeID ) WITH MAX HASH SIZE 9
```

次の文は Initials と EmployeeID の値を両方とも完全にハッシュします。

```
CREATE INDEX compound_2 ON Employees( Initials, EmployeeID ) WITH MAX HASH SIZE 10
```

CREATE PUBLICATION 文 [Ultra Light]

パブリケーションを作成します。

構文

CREATE PUBLICATION [**IF NOT EXISTS**]*publication-name*
(**TABLE** *table-name* [**WHERE** *search-condition*], ... **)**

パラメータ

- **IF NOT EXISTS 句**　IF NOT EXISTS 句を指定し、指定したパブリケーションがすでに存在する場合、変更は行われず、エラーは返されません。

- **TABLE 句**　このテーブルは、パブリケーションに TABLE を含めるときに使用します。TABLE 句の数に制限はありません。

- **WHERE 句**　WHERE 句を指定した場合、同期時に、*search-condition* を満たすローだけが、関連付けられたテーブルからのアップロード対象になります。

  WHERE 句を指定しなかった場合は、最後の同期後にテーブル内で変更されたすべてのローがアップロード対象になります。

備考

パブリケーションは、Ultra Light リモートデータベース内の同期されたデータを示します。

パブリケーションは、単一の同期操作で同期されるテーブルを設定し、Mobile Link サーバにアップロードされるデータを決定します。Mobile Link サーバのダウンロードセッション中には、これらのテーブルだけのローが送り返される可能性がありますが、ダウンロードされるローは、テーブルの WHERE 句を満たす必要はありません。

テーブル全体だけをパブリッシュできます。Ultra Light では、テーブルの特定のカラムはパブリッシュできません。

関連する動作

- オートコミット

参照

- 「Ultra Light の探索条件」281 ページ
- 「Ultra Light クライアント」69 ページ
- 「DROP PUBLICATION 文 [Ultra Light]」441 ページ
- 「ALTER PUBLICATION 文 [Ultra Light]」419 ページ
- 「Ultra Light の探索条件」281 ページ

例

次の文は、2 つのテーブルのすべてのカラムとローをパブリッシュします。

```
CREATE PUBLICATION pub_contact (
  TABLE Contacts,
  TABLE Customers
)
```

次の文は、Customers テーブルで、State カラムに MN が含まれているローのみをパブリッシュします。

```
CREATE PUBLICATION pub_customer (
  TABLE Customers
  WHERE State = 'MN'
)
```

CREATE SYNCHRONIZATION PROFILE 文 [Ultra Light]

Ultra Light の同期プロファイルを作成または置換します。

構文

CREATE [OR REPLACE] SYNCHRONIZATION PROFILE *sync-profile-name sync-option* **[;...]**

sync-option :
sync-option-name **=** *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

パラメータ

- **OR REPLACE 句**　指定の同期プロファイルがすでに存在する場合は、そのプロファイルが置換されます。プロファイルが存在しない場合は作成されます。
- **sync-profile-name**　同期プロファイルの名前。
- **sync-option**　1 つ以上の同期オプションの値ペアがセミコロンで区切られた文字列。たとえば、'option1=value1;option2=value2' のように記述します。
- **sync-option-name**　同期プロファイルオプションの名前。
- **sync-option-value**　同期プロファイルオプションの値。

備考

同期プロファイルによって、Ultra Light データベースが Mobile Link サーバと同期する方法を定義します。

REPLACE 句を使用して、既存の同期プロファイルを変更できます。この句により、同期プロファイルの内容が、新しい sync-option 文字列に含まれている内容に置換されます。この方法は、同期プロファイルを削除してから、新しい文字列を使用して同じ名前のプロファイルを作成する

方法と同じです。したがって、同期プロファイルに完全な同期定義を含める必要はありません。同期の実行時にパラメータをマージまたは上書きできるからです。

同期プロファイルオプション STREAM は、その値にサブリストが含まれているため他のオプションとは異なります。たとえば、'STREAM=TCPIP{host=192.168.1.1;port=1234}' とあるとします。この例では、'host=192.168.1.1;port=1234' がサブリストです。サブリストの値を追加または削除するには、STREAM の sync-option-name と sub-option-name の間にピリオドを使用します。たとえば、MERGE 'stream.port=5678;stream.host=;compression=zlib' のようにすると、同期プロファイルは stream=TCPIP{port=5678;compression=zlib} になります。ストリームを新しい値に設定しようとすると、ストリーム全体の値が置換されます。たとえば、MERGE 'stream=HTTPS' のようにすると、同期プロファイルは stream=HTTPS{} になります。

関連する動作

なし。

参照

- 「同期プロファイルオプション」227 ページ
- 「ALTER SYNCHRONIZATION PROFILE 文 [Ultra Light]」420 ページ
- 「DROP SYNCHRONIZATION PROFILE 文 [Ultra Light]」441 ページ
- 「SYNCHRONIZE 文 [Ultra Light]」455 ページ

例

次の文は、Test1 という同期プロファイルを作成します。

```
CREATE SYNCHRONIZATION PROFILE Test1
'MobiLinkUid=mary;Stream=TCPIP{host=192.168.1.1;port=1234}'
```

CREATE TABLE 文 [Ultra Light]

テーブルを作成します。

構文

CREATE TABLE [**IF NOT EXISTS**] *table-name* **(**
{ *column-definition* | *table-constraint*| *sync-constraint* }, ...
)

column-definition :
column-name *data-type*
[[**NOT**] **NULL**]
[**DEFAULT** *column-default*]
[**STORE AS FILE** (*file-name-column*) [**CASCADE DELETE**]
[*column-constraint*]

column-default :
AUTOFILENAME(*prefix*,*extension*)
| **GLOBAL AUTOINCREMENT** [**(** *number* **)**]
| **AUTOINCREMENT**
| **CURRENT DATE**
| **CURRENT TIME**
| **CURRENT TIMESTAMP**

| **CURRENT UTC TIMESTAMP**
| **NULL**
| **NEWID( )**
| *constant-value*

file-name
"*filename*"

column-constraint :
PRIMARY KEY
| **UNIQUE**

table-constraint :
{ [**CONSTRAINT** *constraint-name*]
pkey-constraint
| *fkey-constraint*
| *unique-key-constraint* }
[**WITH MAX HASH SIZE** *integer*]

pkey-constraint :
PRIMARY KEY [*ordered-column-list*]

fkey-constraint :
[**NOT NULL**] **FOREIGN KEY** [*role-name*] **(** *ordered-column-list* **)**
REFERENCES *table-name* **(** *column-name*, ... **)**
[**CHECK ON COMMIT**]

unique-key-constraint :
UNIQUE (*ordered-column-list* **)**

ordered-column-list :
(*column-name* [**ASC** | **DESC**], ... **)**

sync-constraint :**SYNCHRONIZE** {**ON**| **OFF**| **ALL**| **DOWNLOAD**}

パラメータ

IF NOT EXISTS 句　この句は、テーブルを作成するために使用します。指定された名前のテーブルがすでに存在する場合、変更は行われず、エラーは返されません。

column-definition　テーブル内のカラムを定義します。この句には、次のパラメータを使用できます。

○ **column-name**　カラム名は識別子です。同じテーブル内の 2 つのカラムが同じ名前を持つことはできません。「Ultra Light の識別子」269 ページを参照してください。

Ultra Light Java Edition データベースは、データベースファイルの分割をサポートしているので、たとえば、外部ファイルを使用して大きな BLOB 値を保存し、そのファイルを 2 つのカラムで参照し、最初のカラムには、データ型が CHAR(size) ... AUTOFILENAME(...) のファイル名を格納し、2 番目のカラムはファイルコンテンツへのアクセスに使用して、LONG BINARY STORE AS ... データ型として宣言するといったことが可能です。ファイルコンテンツのカラムは読み取り専用です。

○ **data-type**　カラムのデータ型です。「Ultra Light SQL データ型」293 ページを参照してください。

○ **[NOT] NULL**　NOT NULL が指定されているか、カラムが PRIMARY KEY または UNIQUE 制約下にある場合、カラムはいずれのローでも NULL を持つことはできません。それ以外の場合は、NULL は許可されます。

○ **column-default**　カラムのデフォルト値を設定します。DEFAULT 値を指定する場合、カラムの値を指定しない INSERT 文のカラムの値としてこのデフォルト値が使用されます。INSERT 文はカラムの値を指定しません。DEFAULT を指定しない場合、DEFAULT NULL と等しくなります。デフォルトのオプションを以下に示します。

- **AUTOFILENAME**　この句では、分割された Ultra Light Java Edition データベースへの外部 BLOB ファイルの格納がサポートされています。

  データベースを分割するときに、ファイル名を格納するように指定されたカラムには、AUTOFILENAME(prefix,extensio) 句が必要です。この句では、ダウンロードされた BLOB 値用の新しいファイル名をどのように生成するかを指定します。prefix 値と extension 値は文字列リテラル定数です。Connection.OPTION_BLOB_FILE_BASE_DIR 変数 [BlackBerry] [Ultra Light J]『Ultra Light Java プログラミング』を参照してください。

- **AUTOINCREMENT**　AUTOINCREMENT を使用する場合、カラムは整数データ型の 1 つ、または真数値型にします。テーブルに挿入する場合、AUTOINCREMENT カラムの値を指定しないと、カラム内の任意の値より大きいユニーク値が生成されます。INSERT で、カラムの現在の最大値より大きい値を指定した場合、この値が後続の挿入処理の開始ポイントとして使用されます。

  ヒント
  Ultra Light では、テーブルが作成された時点でのオートインクリメントの初期値は 0 ではありません。カラムに符号付きデータ型が指定されている場合は、オートインクリメントによって負の値が生成されます。このため、オートインクリメントを適用するカラムを符号なし整数として宣言し、負の値が生成されないようにしてください。

- **GLOBAL AUTOINCREMENT**　ドメインが分割されるという点を除いて、AUTOINCREMENT と同じです。各分割には同じ数の値が含まれます。データベースの各コピーにユニークなグローバルデータベース ID 番号を割り当てます。Ultra Light では、データベースのデフォルト値は、そのデータベース番号でユニークに識別された分割からのみ設定されます。

  ヒント
  カラムの型が BIGINT または UNSIGNED BIGINT である場合、デフォルトの分割サイズは 2^32 = 4294967296 です。それ以外の型のカラムの場合、デフォルトの分割サイズは 2^16 = 65536 です。特に、カラムの型が INT または BIGINT ではない場合は、これらのデフォルト値が適切ではないことがあるため、分割サイズを明示的に指定するのが最も賢明です。

  「Ultra Light の GLOBAL AUTOINCREMENT カラム」71 ページと「Ultra Light global_database_id オプション」194 ページを参照してください。

- **[NOT] NULL**　カラムに NULL を格納できるかどうかを制御します。

- **NEWID()**　ユニークな識別子の値を生成する関数です。「NEWID 関数 [その他]」374 ページを参照してください。

- **CURRENT TIMESTAMP**　CURRENT DATE と CURRENT TIME を結合して、TIMESTAMP 値を形成します。年、月、日、時、分、秒、秒の小数位で構成されます。秒は小数第 3 位まで格納されます。精度はシステムクロックの精度によって制限されます。「CURRENT TIMESTAMP 特別値」273 ページを参照してください。

- **| CURRENT UTC TIMESTAMP**　TIMESTAMP WITH TIMEZONE の値には、年、月、日、時、分、秒、秒の小数位、タイムゾーンで構成される協定世界時 (UTC: Coordinated Universal Time) が含まれます。秒は小数第 3 位まで格納されます。精度はシステムクロックの精度によって制限されます。「CURRENT UTC TIMESTAMP 特別値」273 ページを参照してください。

- **CURRENT DATE**　現在の年、月、日を格納します。「CURRENT DATE 特別値」271 ページを参照してください。

- **CURRENT TIME**　現在の時、分、秒 (小数位あり) で構成される時刻を格納します。「CURRENT TIME 特別値」272 ページを参照してください。

- **constant-value**　カラムのデータ型の定数です。通常、この定数は数値または文字列です。

○ **STORE AS FILE (*file-name-column*) [CASCADE DELETE]**　Ultra Light Java Edition データベースでのみサポートされます。

外部保存される LONG BINARY カラム (データベース分割) を指定し、外部保存される BLOB 値のファイル名の保存に使用されるカラムを命名する **file-name-column** を指定します。この句を持つカラムは LONG BINARY 型で読み取り専用カラムとして動作する必要があります。

○ **column-constraint 句**　カラム制約を指定して、カラムに許容される値を制限します。カラム制約は次のいずれかです。

- **PRIMARY KEY**　*column-constraint* の一部として設定するときに、PRIMARY KEY 句はテーブルのプライマリキーとしてカラムを設定します。プライマリキーは、テーブル内の各ローをユニークに識別します。デフォルトでは、プライマリキーに含まれるカラムには NULL を使用できません。

- **UNIQUE**　テーブル内の各ローをユニークに識別する 1 つまたは複数のカラムを識別します。テーブル内の異なるローが、指定されているすべてのカラムで同じ値を持つことはできません。1 つのテーブルに複数の一意性制約が存在することがあります。NULL 値は使用できません。

table-constraint 句　テーブル制約を指定して、テーブル内の 1 つまたは複数のカラムに格納できる値を制限します。CONSTRAINT 句を使用して、テーブル制約の識別子を指定します。以下に定義するように、テーブル制約は、プライマリキー制約、外部キー制約、一意性制約のいずれかの形式にできます。

○ **pkey-constraint 句**　テーブルのプライマリキーとして、指定したカラムを設定します。プライマリキーは、テーブル内の各ローをユニークに識別します。プライマリキーに含まれるカラムには NULL を使用できません。

○ **fkey-constraint 句**　外部キー制約を指定して、1 つまたは複数のカラムの値を、別のテーブルのプライマリキー (または一意性制約) の値と一致するように制限します。

- **NOT NULL 句**　NOT NULL を指定して、外部キーカラムを NULL 入力不可にします。外部キー内の NULL は、外部テーブルのこのローに該当するローがプライマリテーブル内にないことを意味します。複数カラムの外部キー内の少なくとも 1 つの値が NULL である場合、キーの他のカラムに保持できる値に関する制限はありません。

- **role-name 句**　*role-name* を指定して、外部キーに名前を付けます。*role-name* は、同じテーブル内で外部キーを区別する場合に使用します。また、CONSTRAINT *constraint-name* を使用して外部キーに名前を付けることができます。ただし、両方のメソッドを使用して外部キーに名前を付けないでください。

- **REFERENCES 句**　REFERENCES 句を指定し、プライマリテーブルで、外部キー制約として使用する 1 つまたは複数のカラムを定義します。REFERENCES カラム制約に *column-name* を指定する場合、一意性制約またはプライマリキーの制約を受ける、プライマリテーブルのカラム名を指定します。

- **CHECK ON COMMIT**　Ultra Light Java Edition データベースではサポートされていません。データベースサーバで、COMMIT を待ってから外部キー制約を実行するように、CHECK ON COMMIT を指定します。デフォルトでは、挿入、更新、削除の各操作中に、外部キー制約がただちに実行されます。ただし、CHECK ON COMMIT を設定すると、データベースの変更が外部キー制約に違反している場合でも、次の COMMIT 前にデータの不整合が解決されればデータベースの変更を任意の順序で実行できます。

○ **unique-key-constraint 句**　一意性制約を指定して、テーブル内の各ローをユニークに識別する 1 つ以上のカラムを指定します。テーブル内の異なるローが、指定されているすべてのカラムで同じ値を持つことはできません。1 つのテーブルに複数の一意性制約が存在することがあります。

○ **WITH MAX HASH SIZE**　このインデックスのハッシュサイズをバイト単位で設定します。この値は、データベースに対して有効なデフォルトの MaxHashSize プロパティを上書きします。 デフォルトのサイズの詳細については、「データベースプロパティの読み取り」39 ページを参照してください。「Ultra Light max_hash_size 作成パラメータ」147 ページを参照してください。

sync-constraint 句　テーブルを同期可能にするかどうかおよびすべてのローをアップロードするか、テーブルへの変更だけをアップロードするか、テーブルへの変更をアップロードしないかを決定するための同期制約を指定します。

○ **SYNCHRONIZE ON**　デフォルト設定 - テーブルを同期することができ、テーブルへの変更のみがアップロードで送信されます。

○ **SYNCHRONIZE OFF**　テーブルを同期できず、テーブルをパブリケーションに含めるときにエラーが発生します。

○ **SYNCHRONIZE ALL**　テーブルを同期することができ、テーブル内のすべてのローがアップロードで送信されます。この制約は、Ultra Light Java Edition データベースではサポートされていません。

○ **SYNCHRONIZE DOWNLOAD**　テーブルを統合データベースへの変更と同期できますが、ローカルの変更はアップロードされません。

備考

通常、カラム制約を使うのは、制約がテーブル内で複数の他カラムを参照しない場合です。複数のカラムを参照する場合は、テーブル制約を使用します。この文によって制約違反が発生した場合、文の実行は完了しません。エラーの検出前に文が行った変更が完了していない場合は、エラーがレポートされます。

テーブル内の各ローは、ユニークなプライマリキー値を持ちます。

役割名を指定しない場合、役割名は以下のように設定されます。

1. テーブル名と同じ役割名を含む外部キーが存在しない場合、テーブル名が役割名として割り当てられます。
2. テーブル名がすでに使用されている場合、役割名は、0 が埋め込まれた、テーブルに固有の 3 桁の数字と結合されたテーブル名になります。

スキーマの変更　スキーマの変更が同時に開始された場合、文は解放されません。「Ultra Light データベースのスキーマ」49 ページを参照してください。

CREATE TABLE 文の処理中は、そのテーブルを参照する要求やクエリは処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は CREATE TABLE を実行できません。

Ultra Light.NET の場合 : すべてのデータオブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。ULBulkCopy.Dispose メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』を参照してください。

外部 BLOB カラムの同期 (Ultra Light J のみ)　統合データベースでは、ファイル名カラムは通常の CHAR カラムとして格納され、BLOB ファイルカラムは通常の BLOB (LONG BINARY) カラムとして格納されます。ダウンロード時には、ファイル名カラムは無視され、データベースオプション (Connection.OPTION_BLOB_FILE_BASE_DIR) と、DEFAULT AUTOFILENAME 句で指定したプレフィクスと拡張子の文字列を使用して新しいファイル名が生成されます。J2SE の場合、ファイル名カラムに保存された値は構文 <database_option_blob_file_base_dir><prefix><auto generated integer value>.<extension> に一致し、BlackBerry の場合、保存された値は構文 <prefix><auto generated integer value>.<extension> に一致します。このため、BlackBerry の場合、生成されるファイル名は常に相対的になります。

外部 BLOB カラムへのアクセス (Ultra Light J のみ)　外部 BLOB 値を含むファイルは、クライアントアプリケーションがカラム値を読み取ろうとする場合にのみ開かれます。その時点で、STORE AS FILE 句で指定されたカラムに保存されたファイル名は無効なファイル名になります。BlackBerry では、相対ファイル名はデータベースオプション OPTION_BLOB_FILE_BASE_DIR に対して解決されます。Ultra Light でファイル名がプレフィクス "file://" で始まっていないと判断された場合、そのファイルが開かれる前に、ファイル名の先頭に OPTION_BLOB_FILE_BASE_DIR オプションの値が付加されます。

BLOB ファイルは、**file-name-column** カラムのファイル名を指定することによって、データベースに挿入されます。挿入されたファイル名は有効なファイル名である必要があり、Blackberry の

場合、IETF RFCs 1738 & 2396 のファイル URL 形式で記述された完全修飾の絶対パスファイル名の形式に従う必要があります (BlackBerry JDE API ドキュメントのパッケージの javax.microedition.io.file の説明を参照)。挿入時に、STORE AS FILE カラムの値が指定されない場合があります。

ファイルがデータベースに挿入されると、データベースはファイルに対してフルコントロール権を持ち、外部変更が行われないと見なします。

関連する動作

オートコミット

参照

- 「Ultra Light の式」275 ページ
- 「DROP TABLE 文 [Ultra Light]」442 ページ
- 「CREATE TABLE 文」『SQL Anywhere サーバ SQL リファレンス』
- 「Ultra Light SQL データ型」293 ページ
- 「分割サイズ」73 ページ

例

次の文は、図書データベース用にテーブルを作成し、図書情報を保持します。

```
CREATE TABLE library_books (
  isbn CHAR(20)      PRIMARY KEY,
  copyright_date     DATE,
  title              CHAR(100),
  author             CHAR(50),
  location           CHAR(50),
  FOREIGN KEY location REFERENCES room
)
```

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

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

次の文は、販売データベース用にテーブルを作成し、注文と注文項目情報を保持します。

```
CREATE TABLE Orders (
  order_num INTEGER NOT NULL PRIMARY KEY,
  date_ordered DATE,
  name CHAR(80)
);
CREATE TABLE Order_item (
  order_num       INTEGER NOT NULL,
  item_num        SMALLINT NOT NULL,
  PRIMARY KEY (order_num, item_num),
  FOREIGN KEY (order_num)
  REFERENCES Orders (order_num)
)
```

CREATE USER 文 [Ultra Light]

データベースユーザまたはグループを作成します。

> **注意**
> この文は、Ultra Light Java Edition データベースではサポートされていません。

構文

CREATE USER *user-name* **IDENTIFIED BY** *password*

パラメータ

user-name　作成するユーザの名前。

password　作成するユーザのパスワード。

備考

●ユーザ ID には、次に該当する値を指定できません。

○ 空白スペース、一重引用符、または二重引用符で始まる値
○ 空白スペースで終わる値
○ セミコロンを含む値

●パスワードは大文字と小文字が区別され、次に該当する値は指定できません。

○ 空白スペース、一重引用符、または二重引用符で始まる値
○ 空白スペースで終わる値
○ セミコロンを含む値
○ 255 バイト長を超える値

関連する動作

なし。

参照

● 「ALTER USER 文 [Ultra Light]」425 ページ
● 「DROP USER 文 [Ultra Light]」443 ページ

例

次の例は、ユーザ SQLTester を作成し、パスワードを welcome に設定します。

```
CREATE USER SQLTester IDENTIFIED BY welcome
```

DELETE 文 [Ultra Light]

データベースのテーブルからローを削除します。

構文

DELETE [**FROM**] *table-name*[[**AS**] *correlation-name*]
[**WHERE** *search-condition*]

パラメータ

correlation-name　文の他の部分からテーブルを参照するときに使用する識別子。

WHERE 句　WHERE 句を指定すると、*search-condition* を満たすローだけが削除されます。

WHERE 句では、非確定の関数 (RAND など) や変数を使用できません。この句では、カラムも制限されません。カラムをサブクエリで使用する場合、別のテーブルを参照する必要がある場合があります。

備考

Ultra Light では、独自の方法でローのステータスがトレースされます。削除やローのステータスの意味を理解している必要があります。

参照

- 「Ultra Light の探索条件」281 ページ
- 「Ultra Light データベースのローのステータス管理」476 ページ
- 「START SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ
- 「STOP SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ

例

次の文は、Employees テーブルから従業員 105 を削除します。

```
DELETE
FROM Employees
WHERE EmployeeID = 105
```

次の文は、FinancialData テーブルから、2000 年より前のデータをすべて削除します。

```
DELETE
FROM FinancialData
WHERE Year < 2000
```

DROP INDEX 文 [Ultra Light]

インデックスを削除します。

構文

DROP INDEX[**IF EXISTS**] [*table-name*.]*index-name*

備考

テーブルのプライマリインデックスを削除することはできません。

DROP INDEX 文の処理中は、そのインデックスを参照する要求やクエリは処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は DROP INDEX を実行できません。

IF EXISTS 句は、存在しないインデックスを DROP INDEX 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

IF EXISTS 句を指定し、指定のテーブルが見つからない場合、エラーが返されます。

Ultra Light.NET の場合 : すべてのデータオブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。ULBulkCopy.Dispose メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』を参照してください。

スキーマの変更が同時に開始された場合、文は解放されません。「Ultra Light データベースのスキーマ」49 ページを参照してください。

参照

- 「CREATE INDEX 文 [Ultra Light]」428 ページ
- 「Ultra Light インデックス」56 ページ

例

次の文は、FinancialData テーブルで、架空のインデックス fin_codes_idx を削除します。

```
DROP INDEX FinancialData.fin_codes_idx
```

DROP PUBLICATION 文 [Ultra Light]

パブリケーションを削除します。

構文

DROP PUBLICATION[**IF EXISTS**] *publication-name*, ...

備考

IF EXISTS 句は、存在しないパブリケーションを DROP PUBLICATION 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

参照

- 「Ultra Light クライアントの同期設計」74 ページ
- 「ALTER PUBLICATION 文 [Ultra Light]」419 ページ
- 「CREATE PUBLICATION 文 [Ultra Light]」430 ページ

例

次の文は、pub_contact パブリケーションを削除します。

```
DROP PUBLICATION pub_contact
```

DROP SYNCHRONIZATION PROFILE 文 [Ultra Light]

同期プロファイルを削除します。

構文

DROP SYNCHRONIZATION PROFILE [**IF EXISTS**] *sync-profile-name*

パラメータ

- **sync-profile-name**　同期プロファイルの名前。

備考

同期プロファイルによって、Ultra Light または Ultra Light Java Edition データベースが Mobile Link サーバと同期する方法を定義します。

IF EXISTS 句は、存在しない同期プロファイルを DROP SYNCHRONIZATION PROFILE 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

関連する動作

なし

参照

- 「CREATE SYNCHRONIZATION PROFILE 文 [Ultra Light]」431 ページ
- 「ALTER SYNCHRONIZATION PROFILE 文 [Ultra Light]」420 ページ
- 「SYNCHRONIZE 文 [Ultra Light]」455 ページ

例

次の例は、Test1 という同期プロファイルを削除するための構文を示します。

```
DROP SYNCHRONIZATION PROFILE Test1
```

DROP TABLE 文 [Ultra Light]

テーブルとそのすべてのデータをデータベースから削除します。

構文

DROP TABLE [**IF EXISTS**] *table-name*

備考

DROP TABLE 文は、データベースから指定したテーブルを削除します。テーブル内にあるすべてのデータ、およびすべてのインデックスとキーも削除されます。

DROP TABLE 文の処理中は、そのテーブルやテーブルのインデックスを参照する要求やクエリは処理されません。また、アクティブなクエリやコミットされていないトランザクションがある場合は DROP TABLE を実行できません。

IF EXISTS 句は、存在しないテーブルを DROP TABLE 文が削除しようとしたときにエラーを返さないようにする場合に使用します。

Ultra Light.NET の場合、すべてのデータオブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。ULBulkCopy.Dispose メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』を参照してください。

スキーマの変更が同時に開始された場合、文は解放されません。「Ultra Light データベースのスキーマ」49 ページを参照してください。

参照

- 「ALTER TABLE 文 [Ultra Light]」422 ページ
- 「CREATE TABLE 文 [Ultra Light]」432 ページ

例

次の文は、データベースから架空のテーブル EmployeeBenefits を削除します。

```
DROP TABLE EmployeeBenefits
```

DROP USER 文 [Ultra Light]

ユーザを削除します。

> **注意**
> この文は、Ultra Light Java Edition データベースではサポートされていません。

構文

DROP USER *userid* **IDENTIFIED BY** *password*

パラメータ

- **userid**　削除するユーザのユーザ ID。
- **password**　ユーザのパスワード。

備考

なし

関連する動作

なし

参照

- 「ALTER USER 文 [Ultra Light]」425 ページ
- 「CREATE USER 文 [Ultra Light]」439 ページ

例

次の例は、データベースからユーザ SQLTester を削除します。

```
DROP USER SQLTester
```

FROM 句 [Ultra Light]

この句は、SELECT 文に必要なテーブルまたはビューを指定するために使用します。

> **注意**
> この文は、Ultra Light Java Edition データベースではサポートされていません。

構文

FROM *table-expression*, ...

table-expression :
table-name [[**AS**] *correlation-name*]
| **(** *select-list* **)** [**AS**] *derived-table-name* **(** *column-name*, ... **)**
| **(** *table-expression* **)**
| *table-expression join-operator table-expression* [**ON** *search-condition*] ...

join-operator :
,
| **INNER JOIN**
| **CROSS JOIN**
| **LEFT OUTER JOIN**
| **JOIN**

パラメータ

table-name　ベーステーブルまたはテンポラリテーブル。Ultra Light では、異なるユーザがテーブルを所有できません。ユーザ ID でテーブルを修飾した場合、ID は無視されます。

correlation-name　文の他の部分からテーブルを参照するときに使用する識別子。たとえば、次の文で、a は Contacts テーブルの相関名、b は Customers テーブルの相関名として定義されています。

```
SELECT *
FROM Contacts a, Customers b
WHERE a.CustomerID=b.ID
```

derived-table-name　派生テーブルは、FROM 句でネストされている SELECT 文です。

派生テーブルの SELECT リストの項目は、派生テーブル名 (オプション) に続いてピリオド (.) とカラム名を指定して参照します。カラム名があいまいではない場合は、カラム名だけを使用できます。

SELECT 文内から派生テーブルを参照することはできません。

join-operator　ジョインのタイプを指定します。カンマ (,) または CROSS JOIN を指定した場合は、ON サブ句を指定できません。JOIN を指定した場合は、ON サブ句を指定する必要があります。INNER JOIN と LEFT OUTER JOIN の場合は、ON サブ句はオプションです。

備考

FROM 句を指定しない場合、SELECT 文の式は定数式である必要があります。

> **注意**
> この説明はテーブルについてのものですが、特に注意書きがなければ派生テーブルにも適用します。

FROM 句は、指定した全テーブルのすべてのカラムで構成される結果セットを作成します。最初に、指定したテーブルのすべてのローの組み合わせが結果セットの中に入ります。次に、JOIN 条件か WHERE 条件、またはその両方の分だけ、通常は組み合わせの数が減ります。

ジョインのタイプを指定しないで、カンマ区切りリストとしてテーブルをリストする場合、デフォルトでは CROSS JOIN が使用されます。

INNER ジョインの場合、ON 句または WHERE 句を使用してジョインの結果を制限すると、同じ結果が返されます。OUTER ジョインの場合、これらは同じではありません。

注意
Ultra Light では、KEY JOIN と NATURAL JOIN の各ジョインはサポートされていません。

参照

- 「式のサブクエリ」279 ページ
- 「ジョイン：複数テーブルからのデータ検索」『SQL Anywhere サーバ SQL の使用法』
- 「DELETE 文 [Ultra Light]」439 ページ
- 「SELECT 文 [Ultra Light]」451 ページ
- 「UPDATE 文 [Ultra Light]」459 ページ

例

次は、有効な FROM 句です。

```
...
FROM Employees
...

...
FROM Customers
CROSS JOIN SalesOrders
CROSS JOIN SalesOrderItems
CROSS JOIN Products
...
```

次のクエリは、派生テーブルを使用して、SalesOrders テーブルに 4 件以上の注文がある Customers テーブルの顧客名を返します。

```
SELECT Surname, GivenName, number_of_orders
FROM Customers JOIN
    ( SELECT CustomerID, COUNT(*)
      FROM SalesOrders
      GROUP BY CustomerID )
    AS sales_order_counts( CustomerID, number_of_orders )
ON ( Customers.id = sales_order_counts.CustomerID )
WHERE number_of_orders > 3
```

INSERT 文 [Ultra Light]

テーブルにローを挿入します。

構文

INSERT [**INTO**]
table-name [**(** *column-name*, ... **)**]
{ **VALUES (** *expression, ...* **)** | *select-statement* }

備考

INSERT 文を使用すると、1 つのローを挿入するか、クエリの結果セットから複数のローを挿入できます。

カラムを指定した場合は、指定したカラムに 1 対 1 で値が挿入されます。カラム名のリストを指定しないと、テーブルでの表示順 (SELECT * を使用して取り出すときと同じ順序) に値がテーブルカラムに挿入されます。ローは、任意の順序でテーブルに挿入されます。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字の状態のままで格納されます。

参照

● 「SELECT 文 [Ultra Light]」 451 ページ

例

次の文は、データベースに Eastern Sales 部を追加します。

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 230, 'Eastern Sales' )
```

次の文は、otertable の c の値が 10 を超えている場合、othertable から mytable に a と b の値を追加します。

```
INSERT INTO mytable( col1, col2 ) SELECT a, b FROM othertable WHERE c > 10
```

LOAD TABLE 文 [Ultra Light]

外部ファイルからデータベーステーブルにバルクデータをインポートします。

構文

LOAD [**INTO**] **TABLE** [*owner.*]*table-name*
(*column-name*, ... **)**
FROM *stringfilename*
[*load-option* ...]

load-option :
CHECK CONSTRAINTS { **ON** | **OFF** }
| **COMPUTES** { **ON** | **OFF**}
| **DEFAULTS** { **ON** | **OFF** }
| **DELIMITED BY** *string*
| **ENCODING** *encoding*
| **ESCAPES** { **ON** }
| **FORMAT** { **ASCII** | **TEXT**}
| **ORDER** { **ON** | **OFF**}
| **QUOTES** { **ON** | **OFF** }

| **SKIP** *integer*
| **STRIP** { **ON** | **OFF** | **BOTH** }
| **WITH CHECKPOINT** { **ON** | **OFF** }

comment-prefix : string

encoding : string

パラメータ

- **column-name**　この句は、データのロード先となる 1 つ以上のカラムを指定するために使用します。DEFAULTS が OFF に設定されている場合、カラムリストにないカラムは NULL になります。DEFAULTS オプションが ON で、カラムにデフォルト値が入っている場合は、その値が使用されます。DEFAULTS オプションが OFF で、NULL 入力不可のカラムがカラムリストから省かれている場合は、データベースサーバは、空の文字列をカラムの型に変換しようとします。

  カラムリストが指定されていると、ファイルに存在すると思われるカラムと、想定されるファイル内でのカラムの順序がカラムリストにリストされます。カラム名を繰り返すことはできません。

- **FROM string-filename**　この句は、データのロード元となるファイルを指定するために使用します。*string-filename* は文字列としてデータベースサーバに渡されます。したがって、文字列は他の SQL 文字列と同じデータベースのフォーマット要件に従います。特に、次の点に注意してください。

  ○フォルダパスを示すには、円記号 (¥) を 2 つの円記号で表してください。したがって、ファイル *c:¥temp¥input.dat* から Employees テーブルにデータをロードする文は、次のようになります。

  ```
  LOAD TABLE Employees
  FROM 'c:¥¥temp¥¥input.dat' ...
  ```

  ○パス名はデータベースサーバを基準にした相対パスを指定します。クライアントアプリケーションではありません。

  ○UNC パス名を使用すると、データベースサーバ以外のコンピュータ上のファイルからデータをロードできます。

- **load-option 句**　さまざまなロードオプションを使用して、データのロード方法を制御できます。次のリストは、サポートされているロードオプションを示します。

  ○ **CHECK CONSTRAINTS 句**　この句は、ロード中に制約をチェックするかどうかを制御します。CHECK CONSTRAINTS はデフォルトで ON に設定されますが、アンロードユーティリティ (ulunload) は CHECK CONSTRAINTS を OFF に設定して LOAD TABLE 文を書き出します。CHECK CONSTRAINTS を OFF に設定すると、データベースの再構築などの場合に役立つ検査制約が無効になります。

  ○ **COMPUTES 句**　このオプションは処理されますが、Ultra Light では無視されます。

  ○ **DEFAULTS 句**　デフォルトでは、DEFAULTS は OFF に設定されています。DEFAULTS が OFF の場合は、カラムリストにないカラムすべてに NULL が割り当てられます。

DEFAULTS が OFF に設定されており、NULL 入力不可のカラムがカラムリストから省かれている場合は、データベースサーバは、空の文字列をカラムの型に変換しようとします。DEFAULTS が ON に設定されており、カラムにデフォルト値が入っている場合は、その値が使用されます。

○ **DELIMITED BY 句**　この句は、カラムデリミタ文字を指定するために使用します。デフォルトのカラムデリミタ文字列はカンマです。ただし、最長で 255 バイトの文字列を指定できます。たとえば、... DELIMITED BY '###' ... などです。他の SQL 文字列と同じフォーマット要件が適用されます。タブで区切った値を指定する場合、タブ文字を表す 16 進のエスケープシーケンス (9) を使用して、... DELIMITED BY '¥x09' ... のように指定します。

○ **ENCODING 句**　この句は、データベースにロードするデータに使用する文字エンコードを指定します。Ultra Light は文字セット変換を実行しません。データファイルのエンコードがデータベースと一致している必要があります。

○ **ESCAPES 句**　ESCAPES は常に ON であるため、データベースサーバによって円記号に続く文字が認識され、特殊文字として解釈されます。改行文字は ¥n との組み合わせとしてインクルードされ、他の文字はタブ文字の ¥x09 のような 16 進の ASCII のコードとしてデータにインクルードされます。2 つの円記号 (¥) は 1 つの円記号として解釈されます。円記号 (¥) の後に n、x、X、¥ 以外の文字がある場合、それらは別々の文字と解釈されます。たとえば、¥q であれば、円記号と q が挿入されます。

○ **FORMAT 句**　この句は、データのロード元となるデータソースのフォーマットを指定します。TEXT の場合、(ENCODING オプションで定義したとおり) 入力行は文字であり、1 行あたり 1 つのローで構成され、カラムデリミタ文字列によって値が区切られているものとみなされます。ASCII もサポートされています。

○ **QUOTES 句**　この句は、文字列を引用符で囲むかどうかを指定します。Ultra Light は ON のみをサポートしているため、LOAD TABLE 文は引用符文字で囲まれた文字列を探します。引用符文字はアポストロフィ (一重引用符) です。文字列の中で最初に出てくるこのような文字は、その文字列の引用符文字として処理されます。文字列の終わりには先頭にあるものと同じ引用符が必要です。

カラムデリミタ文字列をカラム値の中に入れることができます。また、引用符文字は値の一部とはみなされません。したがって、次の行はアドレスにカンマがあるかどうかは関係ありません。また、アドレスを囲む引用符は、データベースへは挿入されません。

```
'123 High Street, Anytown',(715)398-2354
```

値の中に引用符文字をインクルードするには、2 つの引用符を使用する必要があります。次の行は、第 3 カラムの中へ一重引用符文字である値を入れます。

```
'123 High Street, Anytown','(715)398-2354',""
```

○ **SKIP 句**　この句は、ファイルの最初の数行を無視するかどうかを指定するために使用します。*integer* 引数では、スキップする行数を指定します。たとえば、この句を使用して、カラム見出しを含む行をスキップできます。

○ **STRIP 句**　この句は処理されますが、無視されます。この句は、引用符がない値に、値を挿入する前に削除された先行ブランクまたは後続ブランクがあるかどうかを指定します。STRIP オプションは次のオプションを受け入れます。

- **STRIP ON**　先行ブランクを削除します。
- **STRIP OFF**　先行ブランクまたは後続ブランクを削除しません。
- **STRIP BOTH**　先行ブランクと後続ブランクの両方を削除します。

○ **WITH CHECKPOINT 句**　この句は、チェックポイントを実行するかどうかを指定するために使用します。デフォルト設定は OFF です。この句を ON に設定すると、文が正常に完了した後、チェックポイントが発行されます。

備考

またこの文は、SQL Anywhere の dbunload ユーティリティ (reload.sql ファイル) の出力処理もサポートします。LOAD TABLE は、Windows 上で DBISQL を使用してのみ使用できます。

> **注意**
> この文は、Ultra Light Java Edition データベースではサポートされていません。

Ultra Light データベースをアンロードまたは再ロードするための推奨メソッドは、ulunload および ulload ユーティリティです。ulinit ユーティリティはスキーマとデータを直接 SQL Anywhere データベースからロードできることにも注意してください。

LOAD TABLE を使うと、ファイルからデータベーステーブルの中へ効率よく大量の挿入を行えます。この文は主に、SQL Anywhere の dbunload ユーティリティ (reload.sql ファイル) の出力処理をサポートする手段です。

FORMAT TEXT の場合、値を指定しないことが、NULL 値を指定することになります。たとえば、1,,'Fred', という値を含むファイルに 3 つの値が予想される場合、挿入される値は 1、NULL、Fred です。ファイルに 1,2, が含まれる場合、値 1、2、NULL が挿入されます。空白のみで構成される値は、NULL 値と扱われます。ファイルに 1, ,'Fred', が含まれる場合、値 1、NULL、Fred が挿入されます。他の値はすべて NULL 以外と扱われます。たとえば、'' (一重引用符が 2 つ続く) は空の文字列です。'NULL' は 4 文字の文字列です。

LOAD TABLE でロードしていないカラムが NULL 値を許容しておらず、ファイル値が NULL の場合、数値カラムには値 0 (ゼロ)、文字カラムには空の文字列 ('') が指定されます。LOAD TABLE でロードされたカラムが NULL 値を許容し、ファイル値が NULL の場合、カラム値は (すべての型で) NULL になります。

テーブルに a、b、c の各列が含まれ、入力データに a、b、c が含まれているときに、LOAD 文がデータのロード先として a と b のみを指定した場合は、カラム c に次の値が挿入されます。

- DEFAULTS ON が指定されると、カラム c にデフォルト値が入っている場合は、デフォルト値が使用されます。
- カラム c にデフォルト値がなく、NULL が許容されている場合は、NULL が使用されます。

- カラム c にデフォルト値がなく、NULL が許容されていない場合は、カラムのデータ型に応じて、ゼロ (0) または空の文字列 ('') が使用されるかエラーが返されます。

関連する動作

オートコミット

参照

- 「INSERT 文 [Ultra Light]」445 ページ
- 「Ultra Light データベースのアンロードユーティリティ (ulunload)」232 ページ
- 「アンロードユーティリティ (dbunload)」『SQL Anywhere サーバ データベース管理』

例

次に、LOAD TABLE の例を示します。まずテーブルを作成し、次に *input.txt* というファイルを使用してテーブルにデータをロードします。

```
CREATE TABLE t( a CHAR(100) primary key, let_me_default INT DEFAULT 1, c CHAR(100) )
```

次に、ファイル *input.txt* の内容を示します。

```
'this_is_for_column_c', 'this_is_for_column_a', ignore_me
```

次の LOAD 文は、*input.txt* というファイルをロードします。

```
LOAD TABLE T ( c, a ) FROM 'input.txt' FORMAT TEXT DEFAULTS ON
```

コマンド SELECT * FROM t は、次の結果セットを返します。

a	let_me_default	c
this_is_for_column_a	1	this_is_for_column_c

ROLLBACK 文 [Ultra Light]

トランザクションを終了し、COMMIT 文または ROLLBACK 文が最後に実行されて以降の変更を元に戻します。

構文

ROLLBACK [WORK]

備考

Ultra Light SQL を使用するとトランザクションが作成されます。トランザクションは、最後の ROLLBACK または COMMIT 後のすべての変更内容 (INSERT、UPDATE、DELETE) から構成されます。ROLLBACK 文は、現在のトランザクションを終了し、前回の COMMIT または ROLLBACK 以降にデータベースに対して行われたすべての変更を元に戻します。

参照

- 「COMMIT 文 [Ultra Light]」427 ページ

例

次の文は、前回コミット時の状態にデータベースをロールバックします。

```
ROLLBACK
```

SELECT 文 [Ultra Light]

データベースから情報を取得します。

構文

SELECT [**DISTINCT**] [*row-limitation*]
select-list
[**FROM** *table-expression*, ...]
[**WHERE** *search-condition*]
[**GROUP BY** *group-by-expression*, ...]
[**ORDER BY** *order-by-expression*, ...]
[**FOR** { **UPDATE** | **READ ONLY** }]
[**OPTION** (**FORCE ORDER**)]

row-limitation :
FIRST
| **TOP** *n* [**START AT** *m*]

select-list :
expression [[**AS**] *alias-name*], ...

order-by-expression :
{ *integer* | *expression* } [**ASC** | **DESC**]

パラメータ

DISTINCT 句　DISTINCT を指定して、重複するローを結果から削除します。DISTINCT を指定しない場合は、SELECT 文の句を満たすすべてのローを返します。ここには重複するローも含まれます。多くの場合、DISTINCT を指定すると、文の実行時間が非常に長くなります。したがって、DISTINCT を使用するのは、必要な場合だけにしてください。

row-limitation 句　ロー制限を使用して、結果のサブセットを返します。たとえば、結果セットの最初のローを取得するには、FIRST を指定します。結果の最初の *n* 個のローを返すには、TOP*n* を使用します。TOP*n* 個のローを取得するときの開始ローの位置を制御するには、START AT*m* を指定します。これらの句が意味のある結果を返すようにローの順序を指定するには、SELECT 文に ORDER BY 句を指定します。

select-list　式のリストであり、データベースから何を取り出すかを指定します。通常、SELECT リスト内の式はカラム名です。ただし、関数のように、他のタイプの式にすることもできます。アスタリスク (*) を使用すると、FROM 句にリストされた全テーブルのすべてのカラムを選択できます。オプションとして、*select-list* 内のそれぞれの式に対して、エイリアスを定義できます。エイリアスを使用すると、クエリ内の別の位置 (WHERE 句や ORDER BY 句など) から、*select-list* の式を参照できるようになります。

FROM 句　*table-expression* の中で指定されるテーブルとビューからローを取り出します。
「FROM 句 [Ultra Light]」443 ページを参照してください。

WHERE 句　WHERE 句を指定すると、*search-condition* を満たすローだけが選択されます。「Ultra Light の探索条件」281 ページを参照してください。

GROUP BY 句　GROUP BY 句を使用したクエリの結果には、GROUP BY 式の中の個別の値の各セットに対し 1 つのローが入ります。テーブルリストのローの各グループに対する結果にはローが 1 つずつ含まれるため、結果ローはグループとして頻繁に参照されます。これらのグループ内のローには、集合関数を適用できます。NULL があった場合、ユニークな値と見なされます。

ORDER BY 句　この句で指定した式に従って、クエリの結果がソートされます。ORDER BY 句の各式は、昇順 (ASC) または降順 (DESC) (デフォルト) でソートすることができます。式が整数 *n* である場合、クエリの結果は SELECT リストの *n* 番目の式でソートされます。

特定の順序でローが返されるようにする唯一の方法は ORDER BY を使用することです。ORDER BY 句がない場合は、Ultra Light が最も効率のよい順序でローを返します。

Ultra Light は、LONG VARCHAR 値や LONG BINARY 値の順序をサポートしていません。

FOR 句　この句には、クエリの動作を制御する 2 つの形式があります。

○ **FOR READ ONLY**　この句は、クエリが更新に使用されないことを示します。クエリが更新に使用されないことがわかっている場合は、Ultra Light のパフォーマンスが向上することがあるため、この句はできるだけ指定してください。たとえば、読み込み専用アクセスが必要であることがわかると、Ultra Light はダイレクトテーブルスキャンを実行できます。FOR READ ONLY はデフォルトの動作です。「ダイレクトページスキャン」471 ページを参照してください。

○ **FOR UPDATE**　この句によってクエリを更新に使用できるようになります。この句は、明示的に指定する必要があります。明示的に指定しないと、更新は許可されません (FOR READ ONLY がデフォルトの動作)。

OPTION (FORCE ORDER) 句　この句は一般的には使用しないことをおすすめします。この句は、Ultra Light で選択されたテーブルアクセス順序を上書きし、クエリに出現する順序でテーブルにアクセスするよう Ultra Light に要求します。この句は、クエリの順序が Ultra Light の順序よりも確実に効率的である場合にのみ使用してください。

Ultra Light では、実行プランを使用してクエリを最適化することもできます。「Ultra Light の実行プラン」469 ページを参照してください。

備考

クエリは必ず閉じてください。そうしないと、メモリが解放されず、存在し続けるテンポラリテーブルの数が不必要に増加することになります。

参照

- 「Ultra Light のパフォーマンスに関するヒント」461 ページ
- 「SELECT 文」『SQL Anywhere サーバ SQL リファレンス』
- 「クエリ」『SQL Anywhere サーバ SQL の使用法』

例

次の文は、Employees テーブルから従業員の数を選択します。

```
SELECT COUNT(*)
FROM Employees
```

次の文は、Employees テーブルで 40 番目から 49 番目までの 10 個のローを選択します。

```
SELECT TOP 10 START AT 40 * FROM Employees
```

SET OPTION 文 [Ultra Light]

データベースオプションの値を変更します。

構文

SET OPTION *option-name*=[*option-value*]

option-name: identifier

option-value: string, identifier, または number

備考

この文を使用して、Ultra Light データベース上でオプションを設定できます。ほとんどの Ultra Light オプションはデータベースが最初に作成されたときに設定され、後で変更することはできません。

オプションが永続的かどうかの指定はできません。Ultra Light は永続的なオプションか一時的なオプションかを判別します。永続的なオプションはデータベースに格納されます。一時的なオプションが使用されるのは、接続またはデータベースの実行が停止するまでです。

Ultra Light は、次の永続的なオプションが設定されているとコミット操作を実行します。**global_database_id** と **ml_remote_id**。Ultra Light は一時的なオプションや接続ベースのオプションでは、コミット操作を実行しません。

設定解除が可能なデータベースオプションは **ml_remote_id** だけです。次に例を示します。

```
SET OPTION ml_remote_id=
```

この結果、ID が NULL に設定されます。この場合、Ultra Light は次の同期時に自動的に新しい値を生成します。

参照

- 「Ultra Light global_database_id オプション」194 ページ
- 「Ultra Light ml_remote_id オプション」195 ページ
- 「Ultra Light データベースオプション」191 ページ
- 「DB_PROPERTY 関数 [システム]」348 ページ

例

次の文を実行すると、global_database_id オプションが 100 に設定されます。

```
SET OPTION global_database_id=100
```

START SYNCHRONIZATION DELETE 文 [Ultra Light]

Mobile Link 同期で削除されたローのロギングを再起動します。

構文

START SYNCHRONIZATION DELETE

備考

Ultra Light データベースでは、同期が必要なローへの変更のログを自動的に取ります。これらの変更は次の同期時に統合データベースにアップロードされます。この文を使用すると、その前に STOP SYNCHRONIZATION DELETE 文で停止されていた、削除されたローのロギングを再起動できます。

STOP SYNCHRONIZATION DELETE 文を実行すると、その接続に対してそれ以降に実施された削除操作は同期されません。この効果は、START SYNCHRONIZATION DELETE 文が実行されるまで継続します。

アプリケーションでデータを同期しない場合は、START SYNCHRONIZATION DELETE を使用しないでください。

Ultra Light では、独自の方法でローのステータスがトレースされます。削除やローのステータスの意味を理解している必要があります。

参照

- 「Ultra Light データベースのローのステータス管理」476 ページ
- 「STOP SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ

例

次の一連の SQL 文は、START SYNCHRONIZATION DELETE と STOP SYNCHRONIZATION DELETE の使用方法を示します。

```
STOP SYNCHRONIZATION DELETE;
DELETE FROM PROPOSAL
  WHERE last_modified < months( CURRENT TIMESTAMP, -1 );
START SYNCHRONIZATION DELETE;
COMMIT;
```

STOP SYNCHRONIZATION DELETE 文 [Ultra Light]

Mobile Link 同期で削除されたローのロギングを停止します。

構文

STOP SYNCHRONIZATION DELETE

備考

Ultra Light データベースでは、同期が必要なローへの変更のログを自動的に取ります。これらの変更は次の同期時に統合データベースにアップロードされます。この文を使用すると、その前に START SYNCHRONIZATION DELETE 文で起動されていた、削除されたローのロギングを停止できます。このコマンドは、リモートデータベースに対して訂正を行うには便利ですが、Mobile Link 同期を事実上無効にしてしまうので、使用の際には注意してください。削除のロギングを停止するのは、一時的のみにしてください。

STOP SYNCHRONIZATION DELETE 文を実行すると、その接続に対してそれ以降に実施された削除操作は同期されません。この効果は、START SYNCHRONIZATION DELETE 文が実行されるまで継続します。

アプリケーションでデータを同期しない場合は、STOP SYNCHRONIZATION DELETE を使用しないでください。

Ultra Light では、独自の方法でローのステータスがトレースされます。削除やローのステータスの意味を理解している必要があります。

参照

- 「Ultra Light データベースのローのステータス管理」476 ページ
- 「START SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ

例

次の一連の SQL 文は、START SYNCHRONIZATION DELETE と STOP SYNCHRONIZATION DELETE の使用方法を示します。

```
STOP SYNCHRONIZATION DELETE;
DELETE FROM PROPOSAL
WHERE last_modified < months( CURRENT TIMESTAMP, -1 );
START SYNCHRONIZATION DELETE;
COMMIT;
```

SYNCHRONIZE 文 [Ultra Light]

Mobile Link サーバによって Ultra Light または Ultra Light Java Edition データベースを同期します。

構文

SYNCHRONIZE {
PROFILE *sync-profile-name* [**MERGE** *sync-option* [**;**...]]
| **USING** *sync-option* [**;**...]
}

sync-option :
sync-option-name **=** *sync-option-value*

sync-option-name : *string*

sync-option-value : *string*

パラメータ

- **sync-profile-name**　同期プロファイルの名前。
- **MERGE 句**　この句は、同期プロファイルに指定されているオプションを追加または上書きするために使用します。
- **USING 句**　この句は、同期プロファイルを参照しないで同期オプションを指定するために使用します。
- **sync-option**　1 つ以上の同期オプションの値ペアがセミコロンで区切られた文字列。たとえば、'option1=value1;option2=value2' のように記述します。
- **sync-option-name**　同期オプションの名前
- **sync-option-value**　同期オプションの値。

備考

同期は、同期プロファイルのパラメータに応じて設定します。または、文自体の中でパラメータを指定できます。

同期のオプションをマージできると、開発者は、(MobiLinkPwd などのように) 一部のオプションをデータベースに格納する作業を省略できます。

同期コールバック関数が定義され、Ultra Light に登録されている場合は、SYNCHRONIZE 文を実行するたびに、その同期の進行状況情報がコールバック関数に渡されます。コールバック関数が登録されていない場合、進行状況情報は渡されません。

関連する動作

なし

参照

- 「CREATE SYNCHRONIZATION PROFILE 文 [Ultra Light]」431 ページ
- 「ALTER SYNCHRONIZATION PROFILE 文 [Ultra Light]」420 ページ
- 「DROP SYNCHRONIZATION PROFILE 文 [Ultra Light]」441 ページ
- ULSetSynchronizationCallback メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』

例

次の例は、Test1 という同期プロファイルを削除するための構文を示します。ここでは、MobiLinkPwd がプロファイルの一部として格納されていません。

```
SYNCHRONIZE PROFILE Test1 MERGE 'MobiLinkPwd=sql'
```

次の例は、Test1 という同期プロファイルに、publication オプションと uploadonly オプションを追加するための構文を示します。

```
SYNCHRONIZE PROFILE Test1
  MERGE 'publications=p2;uploadonly=on'
```

次の例は、USING の使用方法を示します。

```
SYNCHRONIZE USING
'MobiLinkUid=joe;MobiLinkPwd=sql;ScriptVersion=1;Stream=TCPIP{host=localhost}'
```

次の例は、publication オプションと uploadonly オプションを同期するための構文を示します。

```
SYNCHRONIZE
  USING 'publications=p2;uploadonly=on'
```

TRUNCATE TABLE 文 [Ultra Light]

テーブルを削除しないで、そのテーブルのすべてのローを削除します。

構文

TRUNCATE TABLE *table-name*

備考

TRUNCATE TABLE 文によってテーブル内のローがすべて削除され、この後の同期時に Mobile Link サーバにこの削除について通知されません。これは、次の文と同義です。

```
STOP SYNCHRONIZATION DELETE;
DELETE FROM TABLE;
START SYNCHRONIZATION DELETE;
```

> **注意**
> この文は、同期またはレプリケーション対象のデータベースに対して慎重に使用する必要があります。Mobile Link サーバに通知されないので、この削除によって整合性が失われ、その結果、同期またはレプリケーションに失敗する可能性があります。

TRUNCATE TABLE 文の実行後、テーブル構造体、すべてのインデックス、制約およびカラム定義は存在し続けます。削除されるのはデータのみです。

テーブルを対象とした文が、別の要求やクエリですでに参照されている場合、TRUNCATE TABLE は実行できません。同様に、テーブルの変更中は、そのテーブルを参照する要求は処理されません。また、データベースにアクティブなクエリやコミットされていないトランザクションがある場合は TRUNCATE TABLE を実行できません。

Ultra Light.NET の場合 : すべてのデータオブジェクト (たとえば ULDataReader) に対して Dispose メソッドも呼び出さないと、この文を実行できません。ULBulkCopy.Dispose メソッド [Ultra Light.NET]『Ultra Light .NET プログラミング』を参照してください。

スキーマの変更　スキーマの変更が同時に開始された場合、文は解放されません。

関連する動作

テーブルに DEFAULT AUTOINCREMENT または DEFAULT GLOBAL AUTOINCREMENT と定義されたカラムがある場合、TRUNCATE TABLE はそのカラムの次に使用可能な値をリセットします。

TRUNCATE TABLE でローが削除済みとマーク付けされると、この処理を実行したユーザは、ROLLBACK 文を発行しないかぎり、ローにアクセスできなくなります。ただし、他の接続から

はアクセスできます。COMMIT を使用すると削除が確定し、すべての接続からデータにアクセスできなくなります。

トランケート対象のテーブルを同期すると、テーブルに適用されているすべての INSERT 文が、TRUNCATE TABLE 文より優先されます。

参照

- 「Ultra Light データベースのスキーマ」49 ページ
- 「DELETE 文 [Ultra Light]」439 ページ
- 「START SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ
- 「STOP SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ

例

次の文は Departments テーブルからすべてのローを削除します。

```
TRUNCATE TABLE Departments
```

この例を実行する場合は、ROLLBACK 文を実行して、変更内容を元に戻すようにしてください。

UNION 文 [Ultra Light]

2 つ以上の SELECT 文の結果を結合します。

> **注意**
> この文は、Ultra Light Java Edition データベースではサポートされていません。

構文

select-statement-without-ordering
[**UNION** [**ALL** | **DISTINCT**] *select-statement-without-ordering*]...
[**ORDER BY** [*number* [**ASC** | **DESC**] , ...]

備考

いくつかの SELECT 文の結果を、UNION を使ってより大きな結果へと結合することができます。各 SELECT 文には、それぞれの SELECT リストに同じ数の式を指定します。各文には ORDER BY 句を含めることはできません。

UNION ALL の結果は、統合された SELECT 文の結果を結合したものです。ローが重複しないようにして結果を取得するには、UNION または UNION DISTINCT を指定します。ただし、重複するローを削除すると、文の合計実行時間が増えます。ローの重複を許容するには、UNION ALL を指定します。

データ型の異なる対応する式を結合しようとすると、Ultra Light は、結合された値を表すデータ型を検索しようとします。この検索ができない場合、統合操作は失敗し、エラーが返されます (例 :「値 'Surname' をデータ型数値に変換できません。」)。

結果に表示されるカラム名は、最初の SELECT 文に対して表示されるカラム名 (またはエイリアス) です。

UNION の ORDER BY は整数フォーマットに制限されています。ORDER BY 句では、整数を使用して順序が確立されます。ここで整数は結果をソートするクエリ式を示します。

参照

- 「SELECT 文 [Ultra Light]」451 ページ

例

次の例は、結合された Employees テーブルと Customers テーブルで見つかったそれぞれの姓すべてをリストします。

```
SELECT Surname FROM Employees
UNION
SELECT Surname FROM Customers
```

UPDATE 文 [Ultra Light]

テーブルのローを変更します。

構文

UPDATE *table-name*[[**AS**] *correlation-name*]
SET *column-name* **=** *expression*, ...
[**WHERE** *search-condition*]

パラメータ

table-name　*table-name* は、更新するテーブルの名前を指定します。使用できるのは単一テーブルのみです。

correlation-name　文の他の部分からテーブルを参照するときに使用する識別子。

SET 句　指定した各カラムに、等号の右側の式の値を設定します。式には制限がありません。式が *column-name* である場合は、古い値が使われます。

SET 句で指定されたカラムの値のみ変更されます。特に、UPDATE を使用して、カラムの値をデフォルトに設定することはできません。

WHERE 句　WHERE 句を指定すると、*search-condition* を満たすローだけが更新されます。

備考

UPDATE 文は、テーブル内の値を修正します。

テーブルに挿入した文字列は、データベースが大文字と小文字を区別するかどうかに関係なく、常に入力された大文字と小文字の状態のままで格納されます。

参照

- 「INSERT 文 [Ultra Light]」445 ページ
- 「DELETE 文 [Ultra Light]」439 ページ
- 「Ultra Light の探索条件」281 ページ

例

次の文は、従業員 Philip Chin (従業員 129) を販売部からマーケティング部 (部門 400) に移動します。

```
UPDATE Employees
SET DepartmentID = 400
WHERE EmployeeID = 129
```

correlation-name を使用した例を次に示します。

```
UPDATE Employee E
SET salary = salary * 1.05
WHERE EXISTS( SELECT 1 FROM Sales S HAVING E.Sales > Avg( S.sales)
GROUP BY S.dept_no )
```

 Copyright © 2013, SAP AG or an SAP affiliate company.

Ultra Light のパフォーマンスに関するヒント

この項では、さまざまな Ultra Light の機能を調整して、クエリ、挿入、更新のパフォーマンスを向上させる方法について説明します。

Ultra Light データベースのキャッシュサイズの調整

キャッシュサイズを明示的に調整する必要はありませんが、モバイルデバイス上のオペレーティングシステムによって、Ultra Light データベースアプリケーションのメモリ使用量を減らすように要求された場合は、キャッシュサイズを調整してください。

Ultra Light データベースのキャッシュサイズは、使用可能なデバイスのメモリが指定したパラメータの範囲内であれば、データ操作に応じて動的に増加されます。通常は、パラメータを指定する必要はありません。データベースが大きい (400 MB など) 場合は、CACHE_MAX_SIZE パラメータを指定して、最大許容サイズをデフォルト値よりも大きくすることができます。Ultra Light は、最大キャッシュサイズに基づいて、いくつかのデータ構造を割り当てます。そのため、デフォルト値はさほど大きく設定されていません。メモリのオーバヘッドを追加するには、より大きな最大値を明示的に要求する必要があります。最大キャッシュサイズに、実際のデータベースファイルの最大サイズより大きい値を指定しても、利点はありません。

Ultra Light のキャッシュサイズは自動的には縮小されません。データベースのキャッシュサイズを制御するには、アプリケーションの cache_allocation データベースオプションを使用する必要があります。オペレーティングシステムによって発行されたメモリ容量低下イベントへの対応として、データベースへの接続後、cache_allocation データベースオプションを調整します。

注意
動的なキャッシュサイズの増加は、Ultra Light Java Edition データベースではサポートされていません。 Ultra Light Java Edition データベースのキャッシュサイズの詳細については、「Ultra Light Java Edition データベースのキャッシュサイズ」462 ページを参照してください。

例

次の UltraLite C++ コードサンプルは、接続文字列の更新によって最大キャッシュサイズを 100 MB に設定する方法を示しています。

```
static ul_char const * ConnectionParms =
   "UID=DBA;PWD=sql;DBF=sample.udb;CACHE_MAX_SIZE=100m";
```

次の UltraLite C++ コードサンプルは、キャッシュ割り当てを半分に減らしてキャッシュサイズを変更する方法を示しています。

```
ULConnection * conn = ULDatabaseManager::OpenConnection(ConnectionParms);
ul_u_long percent;
percent = conn->GetDatabasePropertyInt( "cache_allocation" );
conn->SetDatabaseOptionInt( "cache_allocation", percent / 2 );
```

参照

- 「Ultra Light cache_allocation オプション」191 ページ
- 「Ultra Light CACHE_SIZE 接続パラメータ」168 ページ
- 「Ultra Light CACHE_MIN_SIZE 接続パラメータ」167 ページ
- 「UltraLite CACHE_MAX_SIZE 接続パラメータ」166 ページ

Ultra Light Java Edition データベースのキャッシュサイズ

Ultra Light データベースは、すべてのプラットフォームの一連のページに格納されます。ページのワーキングセットは、キャッシュに保持され、先入れ先出し (FIFO) 方式で管理されます。現在使用中のページは、スワップアウトされないようにキャッシュ内でロックされます。

大規模なデータベースの場合は、データベースを開くたびにデータベースを設定し、メモリに一度に格納するロー数とインデックスページ数を制限できます。

ベンチマークテストを実行すると、Ultra Light Java Edition データベースのパフォーマンスをグラフ化し、ビジネスソリューションに最適なキャッシュサイズを判別できます。

さまざまなキャッシュサイズでテストを実行することによって、パフォーマンスの急激な変動を観察します。キャッシュは、十分なページのワーキングセットを保持できる大きさにする必要があります。キャッシュにストレスをかけるには、次の項目を検討してください。

- テーブルに複数のインデックスを作成し、外部キーを追加する。
- ローをランダムに (インデックスの順序とは異なる順で) 挿入する。
- データベースページサイズの 25 % 以上となる多数のローを作成する。
- インデックスのハッシュを 0 以外の値に設定する。このようにサイズを増やすと、必要とされるページアクセスも増えます。
- 最小のキャッシュサイズからパフォーマンスのグラフ作成を開始する。たとえば、Windows の場合は 256 KB (このプラットフォームの最小許容キャッシュ)、その他のすべてのプラットフォームの場合は 64 KB です。
- データベースを暗号化または難読化する。難読化は、強力な暗号化に比べて使用するコードが少なく、実行する計算も少なくなります。単純暗号化のパフォーマンスは、暗号化をまったく行わない場合よりも若干遅いだけです。ただし、最終的には、強力な暗号化を使用するかどうかは、セキュリティの要件に従って決定する必要があります。

参照

- 「インデックスのハッシュ」464 ページ

クエリのパフォーマンスに関するヒント

この項では、Ultra Light データベースで、既存のクエリのパフォーマンスを向上させる方法について説明します。

インデックススキャンの作成と保持

1 つ以上のインデックスを作成すると、クエリのパフォーマンスが向上します。インデックスのタイプによっては、ローの値がユニークであることを保証できます。

インデックスは、一部またはすべてのカラムの値を基に、テーブルのローに順序を付けます。インデックスを作成する場合、インデックス付けするカラムを指定する順序が、インデックスでカラムが実際に出現する順序になります。インデックスを戦略的に使用すると、インデック付けしたカラムの検索パフォーマンスを大幅に向上できます。

クエリのパフォーマンスを向上するには、次の推奨する手順に従ってください。

- 次のようなカラムにインデックスを作成してください。
  - 定期的に検索する値
  - クエリでテーブルのジョインに使用するカラム
  - ORDER BY 句、GROUP BY 句、または WHERE 句で頻繁に使用するカラム
- 複合インデックスを作成し、インデックスの最初のカラムが、クエリの作成時に述部で最も頻繁に使用されるようにします。
- インデックスによって生じる更新管理オーバヘッドが、デバイスのメモリに負荷をかけすぎていないか確認します。
- 不必要に、インデックスを作成したり保持したりしないでください。インデックスは、カラムのデータが変更されたときに更新する必要があります。したがって、挿入、更新、削除の各操作はインデックスにも実行されます。
- 大規模なテーブルでインデックスを作成してください。
- 冗長なインデックスは作成しないでください。たとえば、カラム (x、y) に対するインデックスをテーブル T に作成した場合に、カラム (x、y、z) に対する別の既存インデックスが T にあると、冗長的になります。

参照

- 「Ultra Light インデックス」56 ページ
- 「テンポラリテーブルの管理」470 ページ
- 「ダイレクトページスキャン」471 ページ
- 「実行プランの表示」472 ページ
- 「複合インデックスについて」57 ページ
- 「EXPLANATION 関数 [その他]」352 ページ
- ULPreparedStatement.GetPlan メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』
- ULCommand.Plan プロパティ [Ultra Light.NET]『Ultra Light .NET プログラミング』
- PreparedStatement.getPlan メソッド [Ultra Light J]『Ultra Light Java プログラミング』

インデックスのハッシュ

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

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

ハッシュキーに含まれるローデータの量は、次の要素によって決まります。

●設定した最大ハッシュサイズプロパティ。

●カラムのデータ型に実際に必要とされるデータサイズ。

ハッシュの例

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

●Anders

●Anderseck

●Andersen

●Anderson

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

●Anders

●Anders

●Anders

●Anders

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

次の文を考えてみます。

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

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

1. 目的のロー ID に一致するテーブルローを探す。

2. データをロードしてアンパックし、値を評価する。

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

●Anders

●Anderse

●Anderse

●Anderso

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

参照

- 「最適なハッシュサイズ制限」466 ページ
- 「Ultra Light インデックスの追加」59 ページ
- 「Ultra Light max_hash_size 作成パラメータ」147 ページ

最適なハッシュサイズ制限

Ultra Light のデフォルトの最大ハッシュサイズである 4 バイトは、ほとんどの場合に適しています。値を大きくしてロー ID とともに含めるデータを増やすことができます。ただし、そうすると、インデックスのサイズが大きくなり、複数ページに断片化される場合があります。この変更によって、結果的にデータベースのサイズが大きくなる可能性があります。最大ハッシュサイズを大きくした場合の影響は、テーブル内のローの数によって異なります。たとえば、ローの数が 2 つか 3 つ程度であれば、インデックスハッシュキーが大きくてもインデックスページに収まります。その場合、インデックスの断片化は起こりません。

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

最適なハッシュサイズを選択したかどうかを確認する唯一の方法は、ターゲットデバイスで Ultra Light クライアントアプリケーションに対してベンチマークテストを実行することです。データベースサイズ自体の変化に加え、さまざまなハッシュサイズがアプリケーションとクエリのパフォーマンスに及ぼす影響についても確認する必要があります。

インデックス化されるカラムに、共通のプレフィクスがない文字列など、適切な値の分散がある場合は、インデックス構造が大きくなっても、インデックスのハッシュによって挿入、更新、削除、検索のパフォーマンスが改善されます。ハッシュされたインデックスは、ハッシュを使用して最初にローを特定してから、ダイレクトロー比較で同じハッシュ値のローを区別します。ハッシュサイズが十分に大きければ、ハッシュをユニークに識別できるので、ローの読み出しと比較は必要ありません。ただし、ページサイズに比べてハッシュサイズが大きすぎると、インデックスに多数のデータベースページが必要になる可能性があります。

データ型

カラム内の値全体をハッシュする場合は、次の表で、データ型ごとに必要なサイズを確認してください。Ultra Light は、必要な場合にのみ最大ハッシュサイズを使用し、指定した最大ハッシュサイズを超えることはありません。カラムのデータ型でフルバイト制限が使用されていなければ、小さい方のハッシュサイズが使用されます。

データ型	値全体のハッシュに使用されるバイト数
LONG VARCHAR、DOUBLE、FLOAT、REAL、LONG BINARY、ST_GEOMETRY	ハッシュなし
BIT、TINYINT	1
SMALLINT	2
INTEGER、DATE	4

データ型	値全体のハッシュに使用されるバイト数
BIGINT、DATETIME、TIME、TIMESTAMP、TIMESTAMP WITH TIMEZONE	8
DECIMAL、NUMERIC	ほぼ正確に 2 等分されます。
CHAR、VARCHAR	文字列全体をハッシュするには、バイト単位の最大ハッシュサイズが、カラムの宣言されたサイズと一致する必要があります。UTF-8 でエンコードされたデータベースでは、宣言されたサイズを常に 2 倍にします。ただし、最大値は 32 バイトです。 たとえば、UTF-8 でエンコードされていないデータベースでカラムを VARCHAR(10) と宣言した場合は、必要なサイズは 10 バイトです。しかし、同じカラムを UTF-8 でエンコードされたデータベースで宣言した場合は、文字列全体のハッシュに使用されるサイズは 20 バイトです。 UltraLite Java Edition データベースでは、各 VARCHAR 文字が UTF-8 表現を使用してハッシュされるので、1 ～ 3 文字の長さになることがあります。Ultra Light Java Edition は (3 * n) バイトを使用して VARCHAR(n) カラムをハッシュします。ただし、VARCHAR(n) カラムに ASCII7 文字しか含まれていない場合は、n バイトのハッシュサイズを指定してすべての値をハッシュできます。
BINARY、VARBINARY	バイト単位の最大ハッシュサイズは、カラムの宣言されたサイズと一致する必要があります。 たとえば、カラムを BINARY(30) と宣言した場合は、必要なサイズは 30 バイトです。
UNIQUEIDENTIFIER	16

たとえば、それぞれ INTEGER と BINARY (20) を宣言した 2 つのカラムの複合インデックスに対して最大ハッシュサイズを 6 バイトに設定した場合、データ型のサイズ要件に基づいて、次の変更が生じます。

- INTEGER カラムのローの値全体がハッシュされ、インデックスに格納されます。これは、整数のデータ型のハッシュに必要なのは 4 バイトだけだからです。

- ハッシュの最初の 4 バイトは INTEGER カラムで使用されるので、BINARY カラムは最初の 2 バイトだけがハッシュされ、インデックスに格納されます。残りの 2 バイトで、BINARY カラムの適切な量がハッシュされない場合は、最大ハッシュサイズを大きくしてください。

ローのデータ

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

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

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

データベースサイズ

各インデックスページには固定のオーバヘッドがありますが、ページのほとんどの領域は実際のインデックスエントリに使用されます。ハッシュサイズを大きくすると、各インデックスエントリが大きくなり、1 ページに収まるエントリの数が少なくなります。大規模なテーブルになると、大規模なハッシュを使用するインデックスの方が、小規模のハッシュを使用するテーブルやハッシュを使用しないテーブルより、使用するページ数が多くなります。必要なページ数が増えると、データベースサイズが増加し、パフォーマンスが低下します。通常、このようなパフォーマンスの低下が生じるのは、キャッシュで保持できるページ数が固定されていて、Ultra Light でページのスワップが発生するためです。

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

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

ハッシュサイズの設定

最大ハッシュサイズは次の 2 つの方法で設定できます。

- データベースでの最大サイズのデフォルトを格納するには、データベースの作成時に max_hash_size 作成パラメータを設定します。デフォルトでインデックスをハッシュしない場合は、この値を 0 に設定します。インデックスをハッシュする場合は、この値を 32 バイト以内の任意の値に変更するか、Ultra Light のデフォルト値 4 バイトを使用できます。

- 新しいインデックスの作成時に特定のハッシュサイズを設定してデフォルト値を上書きします。次のいずれかの方法を使用します。

  - Sybase Central で、新しいインデックスを作成するときに [最大ハッシュサイズ] プロパティを設定します。

○SQL で、CREATE TABLE 文または CREATE INDEX 文で WITH MAX HASH SIZE 句を使用します。

参照

- 「Ultra Light インデックスの追加」59 ページ
- 「Ultra Light SQL データ型」293 ページ
- 「Ultra Light max_hash_size 作成パラメータ」147 ページ
- 「CREATE INDEX 文 [Ultra Light]」428 ページ
- 「CREATE TABLE 文 [Ultra Light]」432 ページ

Ultra Light の実行プラン

Ultra Light の実行プランは、クエリ実行時のテーブルとインデックスへのアクセス方法を示します。Ultra Light には、「クエリオプティマイザ」が含まれています。このオプティマイザは、クエリの効率的なプランの生成を試みる、Ultra Light ランタイムの内部コンポーネントです。オプティマイザは、テンポラリテーブルを使用して中間結果が格納されないようにしたり、クエリで 2 つのテーブルがジョインされるときに、テーブルの関連するサブセットだけがアクセスされるようにしたりします。

オプティマイザの上書き

オプティマイザでは、常に最も効率的なアクセスプランが目標とされますが、特に多数の可能性がある複雑なクエリでは、この目標の達成は保証されません。極端な場合、OPTION (FORCE ORDER) 句をクエリに追加することによって、Ultra Light が選択するテーブル順序を上書きして、クエリに出現する順序でテーブルにアクセスすることもできます。**このオプションは一般的には使用しないことをお勧めします。**パフォーマンスが低い場合、通常は、適切なインデックスを作成して実行速度を上げるようにしてください。

ヒント
クエリを使用してデータを更新しない場合は、クエリで FOR READ ONLY 句を指定してください。こうすると、パフォーマンスが向上することがあります。 「SELECT 文 [Ultra Light]」451 ページを参照してください。

オプティマイザで使用するアクセス方法の決定

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

クエリを最適化する場合、オプティマイザはクエリの要件を確認し、パフォーマンスの向上に使用できるインデックスがないかどうかをチェックします。どのインデックスを使用してもパフォーマンスの改善が望めない場合、オプティマイザはインデックスをスキャンせず、テンポラ

リテーブルかダイレクトページスキャンを使用します。したがって、インデックスを使用して実験し、生成された実行プランを頻繁にチェックして、次の事項を確認する必要があります。

●オプティマイザによって使用されないインデックスを保持していないこと。

●作成されるテンポラリテーブルの数を最小限に抑えていること。

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

実行プランのチェック

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

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

```
scan(T)
```

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

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

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

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

参照

●「テンポラリテーブルの管理」470 ページ

テンポラリテーブルの管理

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

テンポラリテーブルは、アクセスプランによって、実行中にデータを格納するために、一時的なテーブル、つまりテンポラリワークテーブルとして使用されます。このテーブルが存在するのは、アクセスプランの実行中のみです。一般的に、テンポラリテーブルが使用されるのは、使用可能なメモリに中間結果が収まらない場合です。たとえば、次のような場合です。

●サブクエリをアクセスプランの早い段階で評価する必要がある場合

●テンポラリテーブルのデータが単一の接続に対してのみ保持される場合

●クエリにインデックス以外のカラムに対する ORDER BY 句が含まれている場合

●クエリにインデックス以外のカラムに対する GROUP BY 句が含まれている場合

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

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

参照

- 「Ultra Light TEMP_DIR 接続パラメータ」185 ページ
- 「オプティマイザで使用するアクセス方法の決定」469 ページ
- 「実行プランの解釈」473 ページ
- 「実行プランを確認する場合」472 ページ

ダイレクトページスキャン

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

●結果を効率的に返すことができる既存のインデックスがない。

●更新の実行にクエリが使用されない。たとえば、文が FOR READ ONLY (FOR 句が指定されていない場合のデフォルト設定) であると宣言されている場合、またはデータが更新されないことが明らかなようにクエリが記述されている場合です。

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

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

Ultra Light がいつページを直接スキャンするか、または結果を返すのにどのインデックスが使用されたかを確認できます。

参照

- 「オプティマイザで使用するアクセス方法の決定」469 ページ
- 「インデックススキャンの作成と保持」463 ページ

実行プランを確認する場合

次の情報が必要な場合に、Interactive SQL で実行プランを確認します。

- 結果を返すために使用されるインデックス。インデックススキャンオブジェクトにテーブルの名前と、そのテーブルで使用されているインデックスが含まれます。
- 結果を返すためにテンポラリテーブルが使用されるかどうか。テンポラリテーブルは、Ultra Light のテンポラリファイルに書き込まれます。
- テーブルがジョインされる順序。この情報により、パフォーマンスへの影響を確認できます。
- クエリの実行が遅い理由、またはクエリの実行が遅くならないかどうか。

参照

- 「Ultra Light TEMP_DIR 接続パラメータ」185 ページ

実行プランの表示

Interactive SQL を使用して、準備文の実行方法をまとめた Ultra Light プランを表示します。テキストプランは、Interactive SQL のプランビューアに表示されます。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

Ultra Light では、実行プランはプランをテキスト形式で短くまとめたものです。他のプランのタイプはサポートされていません。ただし、プランは短く、情報が 1 行にまとめてあるので、簡単に比較できます。

♦ タスク

1. **[ツール]** » [プランビューアを開く] をクリックします。
2. **[SQL]** ウィンドウ枠で、クエリを入力します。
3. **[プランの取得]** をクリックして、指定した SQL 文のプランを生成します。

結果

プランビューアの下ウィンドウ枠にテキストプランが表示されます。

例

次の文を考えてみます。

```
SELECT I.inv_no, I.name, T.quantity, T.prod_no
FROM Invoice I, Transactions T
WHERE I.inv_no = T.inv_no
```

この文により、下記のようなプランが生成されます。

```
join[scan(Invoice,primary),index-scan(Transactions,secondary)]
```

このプランは、primary というインデックスを使用して Invoice テーブルのすべてのローを読み込むことによって、ジョイン操作を実行することを示します。次に、Transaction テーブルの secondary というインデックスを使用して、inv_no カラムが一致するローのみを読み込みます。

参照

- 「Interactive SQL ユーティリティ (dbisql)」『SQL Anywhere サーバ データベース管理』
- 「実行プランの解釈」473 ページ

実行プランの解釈

Ultra Light のプランは、クエリのアクセス方法をテキスト形式で短くまとめたものなので、テーブルのジョイン操作またはスキャン操作が実装される方法を理解している必要があります。

- **スキャン操作の場合**　1 つのオペランドで表されます。1 つのテーブルのみに適用され、インデックスが使用されます。テーブル名とインデックス名は操作名の後に丸カッコ ((と)) で表されます。

- **その他の操作**　1 つまたは複数のオペランドで表されます。このオペランド自体もプランの場合があります。Ultra Light では、これらのオペランドは、角カッコ ([と]) で囲まれた、カンマ区切りのリストです。

操作リスト

次の表に、Ultra Light でサポートされている操作を示します。

操作	説明
count(*)	テーブルのローの数を数えます。
distinct[*plan*]	クエリの DISTINCT 部分を実装し、重複するローを比較、排除します。基本となるプランでローがソートされるときに使用され、重複、連続するローが排除されます。2 つの連続するローが一致する場合、最初のローだけが結果セットに追加されます。
dummy	操作は実行されません。次の 2 つの場合にのみ使用されます。 ●FROM 句で DUMMY を指定した場合。 ●クエリに FROM 句がない場合。
filter[*plan*]	基本となるプランによって指定される各ローに探索条件を実行します。true と評価されたローだけが、結果セットの一部として転送されます。

操作	説明
group-by[*plan*]	GROUP BY の結果を集約したものを作成し、グループ化されたデータの複数の行をソートします。ローは、発生する順序で表示され、連続するローを比較してグループ化されます。
group-single[*plan*]	1 行だけが返されることがわかっている場合にのみ、GROUP BY の結果を集約したものを作成します。
keyset[*plan*]	テンポラリテーブル内のローの作成に使用されたローを記録し、Ultra Light で元の行を更新できるようにします。これらのローを更新しない場合は、クエリで FOR READ ONLY 句を使用してこの操作をなくします。
index-scan(*table-name*, *index-name*)	テーブルの一部だけを読み出します。開始ローはインデックスを使用して検索します。
join[*plan*, *plan*]	2 つのプラン間で内部ジョインを行います。
lojoin[*plan*, *plan*]	2 つのプラン間で左外部ジョインを行います。
like-scan(*table-name*, *index-name*)	テーブルの一部だけを読み出します。開始ローはインデックスを使用してパターン一致で検索します。
rowlimit[*plan*]	移植されたローに対してローの制限処理を行います。ローの制限は、SELECT 文の TOP n または FIRST 句によって設定されます。
scan(*table-name*, *index-name*)	インデックスが示す順序でテーブル全体を読み出します。
sub-query[*plan*]	サブクエリの開始を示します。
temp[*plan*]	基本となるプラン内のローからテンポラリテーブルを作成します。Ultra Light では、基本となるローをソートする必要があり、そのために使用できるインデックスが見つからなかった場合にテンポラリテーブルが使用されます。 インデックスを追加して、テンポラリテーブルを不要にできます。ただし、インデックスを追加すると、インデックス対象のテーブル内のローの挿入または同期に必要な時間が長くなります。
union-all[*plan*, ..., *plan*]	基本となるプランで生成されるローに対して UNION ALL 操作を行います。

挿入と更新のパフォーマンスのヒント

この項では、Ultra Light データベースで、データの挿入と更新の実行パフォーマンスを向上させる方法について説明します。

トランザクションとローのステータス管理

Ultra Light は、データベース内でデータのほかにステータス情報を保持します。ステータス情報を追跡して格納することにより、次の項目が管理されます。

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

参照

- 「Ultra Light の同時実行性」475 ページ
- 「Ultra Light 同期クライアント」2 ページ
- 「Ultra Light データベースのローのステータス管理」476 ページ
- 「Ultra Light でのトランザクション処理」477 ページ
- 「Ultra Light および Ultra Light Java Edition データベースのバックアップとリカバリ」46 ページ

Ultra Light の同時実行性

Ultra Light では、次の方法により、データベースへの同時アクセスが自動的に管理されます。

- **複数の Ultra Light データベースへのアクセス**　1 つのアプリケーションで、複数のデータベースへの接続を開くことができます。Ultra Light Java Edition では、同時アクセスはサポートされていません。
- **複数のアプリケーション**　Ultra Light または Ultra Light Java Edition データベースは、一度に 1 つのプロセスによってのみ開くことができます。
- **複数のスレッド**　Ultra Light では、マルチスレッドアプリケーションがサポートされています。複数のスレッドを使用し、各スレッドが同じまたは異なるデータベースに接続できる、単一のアプリケーションを作成できます。

- **複数のトランザクション／要求**　各接続は、進行中のトランザクションを一度に 1 つ使用できます。トランザクションは、1 つの要求または複数の要求で構成されます。トランザクションの実行中に行われたデータの変更は、トランザクションがコミットされるまでデータベースで永続化されません。トランザクションで行われたデータの変更は、すべてコミットされるか、すべてロールバックされるかのどちらかです。

- **同期**　アップロードおよびダウンロードでは、読み込み／書き込みアクセスが許可されます。ただし、アプリケーションがローを変更してからダウンロードによってこのローが変更されようとすると、ダウンロードが失敗し、ロールバックが行われます。同期時のデータへのアクセスを無効にするには、Disable Concurrency 同期パラメータを使用します。

  同期に失敗した場合、Ultra Light では、すべてのプラットフォームで再開可能なダウンロードがサポートされます。

参照

- 「ダウンロードの失敗」『Mobile Link サーバ管理』
- 「Ultra Light および Ultra Light Java Edition データベースの制限事項」9 ページ
- 「Ultra Light でのトランザクション処理」477 ページ
- 「Ultra Light クライアント」69 ページ
- 「Additional Parameters 同期パラメータ」90 ページ

Ultra Light データベースのローのステータス管理

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

> **注意**
> Ultra Light Java Edition では、ローのステータス管理はサポートされていません。Ultra Light Java Edition データベースは、トランザクションログを使用して、同期が必要な変更を追跡します。

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

- **削除が実行された場合**　対象の各ローのステータスが、削除されたことを反映して変更されます。ロールバックによって削除が実行されない場合は、ローの元のステータスがリストアされます。

- **削除がコミットされた場合**　対象のローがメモリから削除されない場合もあります。一度も同期が行われていないローは削除されます。同期が実行されたローは削除されません。削除操作を行うには、まず統合データベースと同期させる必要があるからです。次の同期が実行された後、ローはメモリから削除されます。

- **ローが更新された場合**　ローの新しいバージョンが作成されます。古いローと新しいローのステータスが変更され、古いローは表示されなくなり、新しいローが表示されます。

- **ローの更新がコミットされた場合**　トランザクションがコミットされると、トランザクションの影響を受けるすべてのローのステータスが変更され、コミットが反映されます。更新が同期されると、ローの古いバージョンと新しいバージョンの両方で競合が検出され、解決が行われます。その後、古いローがデータベースから削除され、新しいローが通常のローになります。

- **ローが追加された場合**　ローがデータベースに追加され、コミット未実行のマークが付きます。

- **追加されたローがコミットされた場合**　ローにコミット済みのマークが付き、統合データベースとの同期が必要であると通知されます。

参照

- 「Ultra Light および Ultra Light Java Edition データベースのバックアップとリカバリ」46 ページ
- 「単一のトランザクションまたはグループ化されたトランザクションのフラッシュ」478 ページ
- 「Ultra Light でのトランザクション処理」477 ページ

Ultra Light でのトランザクション処理

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

- トランザクションがブロックされること (デッドロック)。Ultra Light が既存のローロックに基づいて要求をブロックすることはありません。このような事態になった場合、Ultra Light はすぐにエラーを返します。

- 未処理の変更を上書きすること。トランザクションは、他のトランザクションの未処理の変更を上書きすることはできません。トランザクションによってローが変更されると、Ultra Light は、トランザクションが「コミット」または「ロールバック」されるまで、このローをロックします。ロックによって、他のトランザクションはローを読み込むことはできても、ローを変更できなくなります。

ヒント
Ultra Light J および C++ の API を除くすべての Ultra Light API は、「オートコミット」モードで動作します。

オートコミットモードでは、Ultra Light は各操作の後にコミットを実行します。一部の API はデフォルトでオートコミットを使用します。これらのインタフェースのいずれかを使用している場合、複数操作トランザクションを使用するには、オートコミットをオフに設定します。オートコミットをオフに設定する方法は、使用しているプログラミングインタフェースによって異なります。ほとんどのインタフェースでは、これは接続オブジェクトのプロパティです。

次の項を参照してください。

- Ultra Light C++ : 「トランザクション管理」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「トランザクション管理」『Ultra Light .NET プログラミング』
- Ultra Light J : 「ロー操作の管理」『Ultra Light Java プログラミング』

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

参照

- Ultra Light C++ : 「エラー処理」『Ultra Light C/C++ プログラミング』
- Ultra Light.NET : 「エラー処理」『Ultra Light .NET プログラミング』
- Ultra Light J : 「エラー処理」『Ultra Light Java プログラミング』

単一のトランザクションまたはグループ化されたトランザクションのフラッシュ

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

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

- **チェックポイントでの制御**　独自のチェックポイントを設定し、これを使用して、それまでの期間に実行された作業を解放します。チェックポイントは、単一トランザクション内にも、複数のトランザクションにわたるものでも、必要に応じていくつでも使用できます。

- **グループ化して制御**　トランザクションカウントスレッショルドかタイムアウトスレッショルドまたはその両方を使用して、実行した作業を解放できます。

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

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

参照

- 「Ultra Light COMMIT_FLUSH 接続パラメータ」171 ページ
- 「Ultra Light commit_flush_count オプション [テンポラリ]」192 ページ
- 「Ultra Light commit_flush_timeout オプション [テンポラリ]」193 ページ
- 「CHECKPOINT 文 [Ultra Light]」426 ページ
- ULCheckpoint メソッド [Ultra Light Embedded SQL]『Ultra Light C/C++ プログラミング』
- ULConnection.Checkpoint メソッド [Ultra Light C++]『Ultra Light C/C++ プログラミング』

Ultra Light のベンチマークに関するヒント

ベンチマークテストは、一般的に、プリケーション開発サイクルの運用段階に入る前に行われます。このフェーズでは、アプリケーションで Ultra Light データベースをテストして、両方のコンポーネントができるだけ効率的に連携することを確認する必要があります。テストで期待どおりのパフォーマンス効率が達成されなかった場合は、データベースのチューニングかアプリケーションの最適化、またはその両方を実行して、ベンチマーク結果を向上させることができます。

注意
Ultra Light の配備が Mobile Link 同期環境内に含まれている場合、必ず同期パフォーマンスもテストしてください。「Mobile Link のパフォーマンスのチューニング」『Mobile Link サーバ管理』を参照してください。

ベンチマークテストの種類

ベンチマークテストを実行して、次のパフォーマンスをチェックできます。

- SQL 文

Ultra Light データベースは、SQL クエリを効率的に処理して、結果をすばやく返すように最適化されています。それでも、データベースのパフォーマンスを向上させるには、重要なクエリが適切に実行されているかどうかを確認する必要があります。

- 同期

  Mobile Link 同期のスループットを最大限にするには、複数の同期が同時に発生し、効率よく実行することが重要になります。

- インデックス

- テーブル設計

- アプリケーションコード

- デバイス設定

  たとえば、Ultra Light の配備ロケーションとして、外部フラッシュメモリとデバイスの内部メモリをそれぞれ使用して比較します。

- データベース設定

  たとえば、キャッシュサイズ、ページサイズ、予約サイズ、インデックス、ハッシュサイズなどを変えてみます。

- データのスループット (トランザクション数／秒に基づく)

  一般的に、Ultra Light は大量データ入力処理用のデータベースではありませんが、業務の要件によっては、このベンチマークについてのテストを実行する場合があります。

- ソフトウェアの変更

  2 つの異なるバージョンの Ultra Light 間、または 2 つの異なるバージョンのアプリケーション間で、ソフトウェアの変更による影響をテストする必要があります。

参照

- 「SQL クエリのテスト」480 ページ
- 「Mobile Link のパフォーマンスのチューニング」『Mobile Link サーバ管理』

SQL クエリのテスト

SQL クエリには、単純なものも複雑なものもあります。クエリの性質と重要性に応じて、2 種類のクエリベンチマークテストを使用できます。

代表 SQL ベンチマーク

この種類のテストでは、日常的な業務においてアプリケーションによって実行される、一般的なトランザクションの代表的な文を選択してテストする必要があります。各アプリケーションにはさまざまな業務用途があるため、アプリケーションごとに異なるベンチマークテストが必要となります。たとえば、計測アプリケーションでは、1 つの基本的な INSERT 文をテストするのみですむ場合があります。一方、モバイル営業支援アプリケーションでは、複数の INSERT 文以外に、複数の SELECT 文と、必要に応じて UPDATE 文をテストする場合があります。

アプリケーションでのクエリの量によって、現実的にテスト可能な範囲が制限されます。クエリ処理が多すぎる場合は、目的の SQL ベンチマークテストの実行が制限されることがあります。

対象を限定した SQL ベンチマーク

アプリケーションで多数の文を使用する場合は、必要に応じて、次の文の一部またはすべてを含めて、テストの範囲を絞ることができます。

- 最も頻繁に使用する文。
- 大量のデータを処理する文。
- 緊急の要件を持つ文。
- アプリケーションを使用する業務状況で最も重要となる文。
- 最も複雑な文 : たとえば、最大数のテーブルジョインが含まれる文や、多くのサブクエリが使用される文です。これらの種類の文ででは、大量のデバイスリソースが使用されることがあります。これらの文については、使用頻度が低くても、デバイスの容量を超えていないことを確認してください。
- インデックスでサポートされていない文。
- 大量のメモリリソースを使用する文。

手法

1. 「準備フェーズ」

   ベンチマークテストを開始する前に、データベース設計を最終決定し、アプリケーション開発の条件を整備します。

2. 「作成フェーズ」

   Ultra Light の配備において予測されるエンドユーザの動作を模倣するカスタムプログラムを構築できます。

3. 「実行および分析フェーズ」

   データベースの各種要素を微調整し、その変更の結果を記録して分析できるようにします。すべての変更から最大限の効果を得ることができるまで、テストを繰り返します。

準備フェーズ

準備フェーズでは、データベースとアプリケーションを適切なベンチマーク候補と見なせる状態にし、テストで達成したい目標を決定できます。次の作業を行います。

1. データベースの論理設計を完了します。

   次のことを確認します。

- テーブルを作成し、代表的なデータを移植していること。
- これらのテーブル内のデータをより効果的に取得するためのインデックスを作成していること。

2. データベースとアプリケーションの両方の物理配備環境を準備します。配備する環境は、最終的な運用環境が正確に再現されている必要があります。つまり、ラボと運用環境には、同じプラットフォームまたはデバイスタイプを使用し、メモリとディスクの構成も同じにしてください。

3. アプリケーションプログラミングフェーズを進める条件が整っていることを確認します。パフォーマンスの欠陥ではなく最適化を追求するようにしてください。ただし、テストの結果として欠陥が見つかる場合もあります。

   すべてのデータベースクエリがデータにアクセスでき、必要な量のデータが返される必要があります。運用環境でデータのソートが必要な場合は、クエリにもこのデータが含まれていることを確認してください。そうしないと、アプリケーションで代表的なメモリ要件を正確にテストできません。

4. データベースとアプリケーションのコピーを最終的なディスクロケーションに配備します。

5. 確認して調整したいデータベースパフォーマンスの要素を決定します。

これで、データベースやアプリケーションに対するベンチマークテストを実行できます。

参照

- 「ベンチマークテストの種類」479 ページ
- 「手法」481 ページ

作成フェーズ

信頼できる結果が得られるテストを作成する必要があります。信頼できるテストでない場合、一定期間における複数のテスト結果を適切に比較できません。

効果的で信頼性の高いベンチマークテストを作成するには、次の要因を考慮します。

- **目標**　パフォーマンス比率を取得しますか。それとも、データベースに対するコマンドの処理に必要な期間を確認しますか。前者については、SQL パフォーマンスをテスト場合、設定された時間間隔が経過するまで、1 つ以上の文を繰り返し実行することがあります。このテストでは、スループット比率を確認できます。スループット比率を簡単に表すと次のようになります。

  statement-number / time-interval = throughput ratio

- **環境**　基準とするテスト環境を確立し、その設計とスコープを記録します。同じ条件で同じテストを実行できないと、そのテストの結果を適切に比較できません。また、ベンチマークテストの一環でラボで使用するハードウェアとソフトウェアが運用環境のものと一致している必要があります。

- **ステータス**　信頼できるベンチマークテストでは、各反復を常に同じアクションで開始します。サードパーティアプリケーションを Ultra Light と同時に動作させる必要があるかどうかを判断してください。これらのアプリケーションがパフォーマンスに影響を及ぼす場合は、ベンチマークテストの設計に追加します。実行する必要のないサードパーティアプリケーションは、必ず完全に終了してください。アプリケーション／プロセスが最小限に抑えられ、アイドル状態になっていても、メモリは引き続き使用されるため、適切な結果を得ることができない場合があります。

- **結果**　ベンチマークの結果は、テストの各反復の後に一貫した方法で残します。一定期間に収集した結果が特定の傾向を示すことがあり、これにより、データベースかアプリケーション、またはその両方にどのような変更を加えると Ultra Light のパフォーマンスを改善できるかを特定しやすくなることがあります。

- **タイミングメカニズム**　ベンチマークテストでは、ユーザアクションをシミュレートします。したがって、通常、これらのアクションで経過した実行時間を追跡します。タイミングメカニズムが体系化されており、実行時間がテストの結果に正確に反映されていることを確認してください。

参照

- 「手法」481 ページ
- 「SQL クエリのテスト」480 ページ

実行および分析フェーズ

ベンチマークの実行は、テストを繰り返し実行してデータベースをチューニングするフェーズです。このフェーズでは、データベースに何らか (たとえば、1 つ以上のデータベースプロパティまたは接続パラメータの値) の変更を加えた後、テストを再実行して変更の結果を確認します。

前提条件

この作業を実行するための前提条件はありません。

内容と備考

次の手順は、効果を最大化するために、さまざまなデータベースプロパティや接続パラメータをテストすることを想定しています。テストが必要なすべてのパラメータのテストが完了するまで、この手順を繰り返してください。

> **ヒント**
> Ultra Light の配備の負荷および目的に対して重要な意味を持つプロパティまたはパラメータのみを選択してください。

♦ タスク

1. テストの最初の反復を実行することによって、基準を作成します。この場合、さまざまなデータベースプロパティや接続パラメータをテストするため、可能な範囲で Ultra Light のデフォルト値を使用します。

2. 標準的なテストの実行を開始します。1 回のテストでチューニングするデータベースプロパティまたは接続パラメータは 1 つのみとします。このように制限することによって、収集する結果がそのアプローチの中で体系化され、チューニング作業において効果が最大化されたタイミングを判断しやすくなります。

3. ベンチマークプログラムからの出力には、次の内容が含まれます。

   ●各テストの識別子またはラベル

   ●プログラムの実行の反復

   ●チェック対象の要素の名前とその要素をどのように変更したか

   ●記録された経過時間

   たとえば、他のデータベースパラメータをテストできたとしても、ページサイズ、キャッシュサイズ、予約サイズのみを変更するようにテストを制限した場合、次の例に示すようなテーブルに出力が保存されます。

```
PROP/PARM          VALUES
TEST NUMBER          001     002     003
page_size        1     2       8
CACHE_SIZE           128     256     512
RESERVE_SIZE          128     256     512

STMT ID            EXECUTION (seconds)
 01              01.55   01.50   01.49
 02              02.01   02.20   01.59
 03              00.33   00.55   00.44
```

4. 反復が完了したら、それ以降の実行結果に意図しない影響を及ぼさないよう、データベースを基準のステータスに戻します。

結果

ベンチマークテストの結果に応じて、次のいずれかを実行します。

●パフォーマンスが向上した場合は、同じプロパティまたはパラメータの値を変更して、テストを再実行します。パフォーマンスがそれ以上向上しなくなるまで、この値のチューニングを続けてください。

●パフォーマンスが低下した場合は、プロパティまたはパラメータの値を前の値に戻します。

次の手順

別の新しいプロパティまたはパラメータをテストします。

参照

●「手法」481 ページ

Ultra Light トラブルシューティング

Ultra Light エンジンを起動できない

現象

START 接続パラメータを使用し、次の定義で Ultra Light エンジンを起動しましたが、クライアントが SQLE_UNABLE_TO_CONNECT_OR_START を返します。

```
START="¥Program Files¥uleng16.exe"
```

説明

引用符の位置が正しくありません。

推奨事項

このパラメータを機能させるには、最初の引用符の前に ¥ 記号が必要です。たとえば、次のようにしてこのパス内のスペースを明確に記述します。

```
START=¥"Program Files¥uleng16.exe"
```

または

```
START="'¥Program Files¥uleng16.exe'"
```

アップグレード後にデータベースに接続できない

現象

Ultra Light をアップグレードしました。管理ツールを使用して空のデータベースを作成できることがわかりました。しかし、Sybase Central を使用して空のデータベースまたはその他のデータベース (*CustDB.udb* を含む) に接続しようとすると、エラーが発生します。ただし、SQL Anywhere データベースとの接続は問題なく動作します。

説明

SQL Anywhere のすべてのアプリケーションとプロセスを閉じていませんでした。このため、Ultra Light のプラグインが正しくインストールされていません。

推奨事項

SQL Anywhere を削除して再インストールします。

1. Sybase Central、Interactive SQL、実行中のすべてのデータベースエンジンを閉じます。
2. 次のコマンドを実行します。

   ```
   dbisql -terminate
   scjview -terminate
   ```

3. Windows の **[タスク マネージャ]** を開き、*scjview.exe* プロセスと *dbisql.exe* プロセスを閉じます。

4. 最新バージョンの Ultra Light を再インストールします。

参照

- 「Ultra Light のアップグレード」『SQL Anywhere 16 変更点とアップグレード』

Ultra Light データベースの破損

現象

次の場合は、Ultra Light データベースが破損している可能性があります。

- 次のエラーが生成される。
  - SQLE_DEVICE_ERROR
  - SQLE_DATABASE_ERROR (他の問題による現象の可能性もあり)
  - SQLE_MEMORY_ERROR (他の問題による現象の可能性もあり)
- クラッシュするか、無効なクエリ結果が返される。

説明

一般的な破損の原因は 2 つあります。

- 頻繁の高い原因としては、ファイルの格納時にデバイスに問題が発生したため、ファイルの内容が誤って変更されることを挙げることができます。通常、この問題が発生すると、Ultra Light データベースはすぐに機能を停止します。
- 頻度の低い原因としては、Ultra Light コードのエラーによってインデックスの正常な管理に失敗することを挙げることができます。クエリの結果に対する変更を予期することが難しいため、これらの問題は長期間発見されない可能性があります。

推奨事項

チェックサムは、Ultra Light データベースのオフラインの破損を検出するために使用します。チェックサムによって、重要なページの破損が原因で他のデータが破損するのを防ぐことができます。チェックサムが一致しなかった場合、Ultra Light データベースでページがロードされるときに、Ultra Light によってすぐにデータベースが停止され、致命的なエラーがレポートされます。このエラーは訂正できません。代わりに、次の手順を実行してください。

1. エラーをレポートします。これは、破損の原因となる一連のイベントがわかっている場合、およびエラーに再現性がある場合に役立ちます。

2. データが必要な場合は、Ultra Light データベースの内容をファイルにアンロードします。

3. 新しい Ultra Light データベースを作成します。

4. アンロードしたデータを同期またはロードすることによって、データを再移植します。

参照

- 「Ultra Light checksum_level 作成パラメータ」140 ページ

データベースのサイズが安定しない

現象

アプリケーションが複数のクライアントデバイス間のラージバイナリオブジェクトを数多く収集して、この情報を統合データベースに同期し、その後、各クライアントデバイスから同期されたデータが削除されます。ただし、データベースからデータが削除されても、データベースのサイズは大きいままです。このことは問題となります。デバイスのリソースは限られているため、ファイルサイズは慎重に管理する必要があります。

説明

データベースのサイズが大きくなるのは、データベース内のデータ量が増える場合のみです。ただし、いったん拡大すると、データベースファイルはそのサイズを維持し、自動的に縮小しません。ファイルの空き領域は、内部的に管理されます。

推奨事項

同期しないテーブルに STOP SYNCHRONIZATION DELETE 文または TRUNCATE 文を使用しないようにします。代わりに、同期しないテーブルには FROM *table-name* 句を含む DELETE 文を使用します。

同期後のデータベースを再作成します。

1. デバイスに配備するデータベースを作成します。
2. クライアントデバイスで必要とされるスキーマを定義する DDL 文の SQL スクリプトを作成します。「Ultra Light データベーススキーマのアップグレードの配備」122 ページを参照してください。
3. データを同期します。
4. データベースを削除します。
5. 新しい空のデータベースを作成し、ALTER DATABASE SCHEMA FROM FILE 文で標準のデータベーススキーマを使用します。

参照

- 「STOP SYNCHRONIZATION DELETE 文 [Ultra Light]」454 ページ
- 「TRUNCATE TABLE 文 [Ultra Light]」457 ページ
- 「DELETE 文 [Ultra Light]」439 ページ
- 「ALTER DATABASE SCHEMA FROM FILE 文 [Ultra Light]」418 ページ

新しい Ultra Light データベースへの ASCII データのインポート

現象

新しい Ultra Light データベースを作成しましたが、インポートできない *.csv* ASCII データファイルがあります。

説明

Ultra Light の管理ツールでは、*.csv* フォーマットはサポートされていません。

推奨事項

次のいずれかの方法を試すことができます。

- Interactive SQL (dbisql) を使用してデータをインポートします。Ultra Light データベースに接続し、**[データ]** » **[インポート]** をクリックします。または、Ultra Light データベースに接続し、INPUT 文を実行します (この文は、Ultra Light の PreparedStatement オブジェクトでは使用できません)。

  > **注意**
  > Ultra Light にはプライマリキーが必要です。Interactive SQL ではユーザのテーブルを作成できますが、そのプライマリキーは自動的に作成されません。この目的のために作成した空の Ultra Light データベースに必ず接続してください。

- バッチプロセスの一部としてこの機能を組み込む場合は、独自のコードを作成してください。

参照

- 「INPUT 文 [Interactive SQL]」『SQL Anywhere サーバ SQL リファレンス』
- 「Ultra Light 用 Interactive SQL ユーティリティ (dbisql)」197 ページ

以前のバージョンのユーティリティが動作する

現象

Ultra Light 16 のインストールを完了しました。しかし、Ultra Light のユーティリティを実行しようとすると、以前のバージョンが起動します。

説明

Ultra Light の複数のバージョンが同じコンピュータにインストールされている場合は、管理ツールを使用するとき、システムのパスに注意してください。インストールでは、最も新しくインストールされたバージョンの実行プログラムのフォルダがシステムパスの最後に追加されるため、ソフトウェアの最新バージョンをインストールしたにもかかわらず、以前にインストールしたバージョンが実行される場合があります。

推奨事項

この問題にはさまざまな対処方法があります。

参照

- 「複数のバージョンのユーティリティがインストールされている場合に正しいバージョンを実行していることを確認する方法」『SQL Anywhere 16 変更点とアップグレード』

結果セットに予測できない変更が生じる

現象

クエリを実行すると、実行するたびに結果セットが変わります。

説明

得られる結果セットを注意深く確認します。セットの結果は本当に違いますか。または、実行のたびに最も効率的な順序で結果が返されているだけではありませんか。ローに最後にアクセスした日時やその他の要因によって、クエリを実行するたびに選択される順序が変わることがあります。

推奨事項

予測可能な順序または一貫性のある順序で結果セットが返されるようにする場合は、SELECT 文に ORDER BY 句が含まれていることを確認します。結果セットで返される結果が依然として不適切な場合は、データベースが破損している可能性があります。

参照

- 「SELECT 文 [Ultra Light]」451 ページ
- 「Ultra Light データベースの破損」486 ページ

Ultra Light エンジンクライアントでエラー -764 が発生する

適用対象

Windows Mobile

現象

Windows Mobile デバイスで Ultra Light エンジンを実行しようとして、クライアントから -764 エラーが返されます。

説明

-764 のエラーは、エンジンが見つからないため、起動できなかったことを意味します。

推奨事項

次のいずれかのアクションを検討します。

- 推奨される配備ロケーション *¥Windows* フォルダにエンジンを再配備することを検討します。Ultra Light は、自動的にこのロケーションでエンジンのファイルを検索します。

- 別のロケーションにエンジンをインストールした場合は、接続コードに START 接続パラメータを使用していることを確認します。

- START 接続パラメータを使用しており、エンジンへのパスが正しい場合は、パス名の特殊文字に正しいエスケープシーケンスを使用していることを確認します。

  たとえば、次のコードは変更が必要となる場合があります。

  ```
  ULConnection conn = new ULConnection(@"dbf=¥Program Files¥HelloEngine¥HelloEngine.udb;
  START=¥Windows¥uleng16.exe")
  ```

  場合によっては、次のようなコードに変更します。

  ```
  ULConnection conn = new ULConnection(@"dbf=¥¥¥"Program Files ¥"¥¥HelloEngine¥
  ¥HelloEngine.udb;
  START=¥¥Windows¥¥uleng16.exe");
  ```

参照

- 「Ultra Light エンジンの起動」124 ページ
- 「Ultra Light START 接続パラメータ」184 ページ

索引

記号

?
Ultra Light 入力パラメータ, 280
-? オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
// コメントインジケータ
Ultra Light 説明, 270
/* コメントインジケータ
Ultra Light 説明, 270
^
Ultra Light ビット処理演算子, 292
~
Ultra Light ビット処理演算子, 292
@data オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
Ultra Light 情報ユーティリティ (ulinfo), 210
Ultra Light データベース検証ユーティリティ (ulvalid), 235
Ultra Light データベース消去ユーティリティ (ulerase), 209
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
Ultra Light 同期ユーティリティ (ulsync), 224
&
Ultra Light ビット処理演算子, 292
% 演算子
モジュロ関数、Ultra Light, 371
|
Ultra Light ビット処理演算子, 292
0 埋め込み
timestamp_with_time_zone_format オプションによる制御, 162
Ultra Light date_format 作成パラメータ, 144
Ultra Light timestamp_format 作成パラメータ, 159
10054
Ultra Light 同期ストリームシステムエラー, 104
10 進数精度
Ultra Light precision, 153
130 エラー
Ultra Light スキーマの更新の SQL コード, 122
16 進文字列
Ultra Light 説明, 354
256 ビットの強力な暗号化
Ultra Light 使用法, 29
Ultra Light 接続パラメータ, 175
-a オプション
ulinit 作成パラメータ, 25
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
-b オプション
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
--case オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--checksum_level オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--collation オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-c オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベース情報ユーティリティ (uljinfo), 251
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light 情報ユーティリティ (ulinfo), 210
Ultra Light データベース検証ユーティリティ (ulvalid), 235
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232

Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
Ultra Light 同期ユーティリティ (ulsync), 224
-d1 オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
--databaseid オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--datacopy オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--date_format オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--date_order オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--dba オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-d オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
-ek オプション
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベース情報ユーティリティ (uljinfo), 251
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
---exactschema オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--exclude オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-e オプション
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース検証ユーティリティ (ulvalid), 235
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
-E オプション
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
--fips オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-f オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
-g オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
--help オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-h オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
--identity-file オプション

Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
--identity--password オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
--insertforupload オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-i オプション
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
--key オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-k オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース消去ユーティリティ (ulerase), 209
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-K オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--list_collation オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--log オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light 同期ユーティリティ (ulsync), 224
-l オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
--max_hash_size オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--mirror_file オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-m オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-nearest_century オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--no_warnings オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-nogui オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
-n オプション
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
--obfuscate オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-onerror オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
--overwrite オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-o オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220

--page_size オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--precision オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--prompt オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--publication オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-p オプション
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベース情報ユーティリティ (uljinfo), 251
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light データベース消去ユーティリティ (ulerase), 209
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
Ultra Light 同期ユーティリティ (ulsync), 224
--quiet オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-q オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベース情報ユーティリティ (uljinfo), 251
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light 情報ユーティリティ (ulinfo), 210
Ultra Light データベース検証ユーティリティ (ulvalid), 235
Ultra Light データベース消去ユーティリティ (ulerase), 209
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
Ultra Light 同期ユーティリティ (ulsync), 224
--reserve_size オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--rootcert オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-r オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light 同期ユーティリティ (ulsync), 224
--SAconnect オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--scale オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--sql オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-SQL コマンド
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
--sync_publication オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-s オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
-S オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--time_format オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
--timestamp_increment オプション

Ultra Light データベース初期化ユーティリティ (ulinit), 211
--timestamp_with_time_zone_format オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-t オプション
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
-ud オプション
Ultra Light エンジンユーティリティ (uleng16), 207
-ul オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
--utf8_encoding オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-u オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース消去ユーティリティ (ulerase), 209
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-version
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
-v オプション
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベース情報ユーティリティ (uljinfo), 251
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light データベース検証ユーティリティ (ulvalid), 235
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
Ultra Light 同期ユーティリティ (ulsync), 224
-w オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-x オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
-y オプション
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload), 254
Ultra Light Java Edition データベースロードユーティリティ (uljload), 252
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
-z オプション
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light データベース初期化ユーティリティ (ulinit), 211
-Z オプション
Ultra Light データベース初期化ユーティリティ (ulinit), 211
- コメントインジケータ
Ultra Light 説明, 270

A

ABS 関数
Ultra Light の構文, 321
ACOS 関数
Ultra Light の構文, 321
ActiveSync
Mobile Link Ultra Light アプリケーションの配備, 87
Ultra Light クライアントへのアプリケーションの登録, 127
Ultra Light プロバイダファイルの配備, 125

Ultra Light 用の ActiveSync プロバイダの配備, 125
ActiveSync プロバイダインストールユーティリティ (mlasinst)
Ultra Light クライアントへのアプリケーションの登録, 127
Additional Parameters
Ultra Light 同期パラメータ, 90
AES 暗号化アルゴリズム
Ultra Light fips 作成パラメータ, 146
Ultra Light fips の使用法, 29
Ultra Light 使用法, 29
Ultra Light 配備手順, 18
AllowDownloadDupRows
Ultra Light 同期パラメータ, 90
allsync テーブル
Ultra Light 同期テーブル, 77
ALL 探索条件
Ultra Light SQL, 284
ALTER DATABASE SCHEMA FROM FILE 文
Ultra Light 構文, 418
Ultra Light スキーマ変更の影響, 49
使用方法, 122
ALTER PUBLICATION 文
Ultra Light 構文, 419
ALTER SYNCHRONIZATION PROFILE 文
Ultra Light 構文, 420
ALTER TABLE 文
Ultra Light 構文, 422
ALTER USER 文
Ultra Light 構文, 425
AND
Ultra Light ビット処理演算子, 292
Ultra Light 論理演算子, 283
Android
暗号化と難読化, 29
ANY 探索条件
Ultra Light SQL, 285
ApplyFile メソッド
スキーマのアップグレードの Ultra Light 代替, 122
ARGN 関数
Ultra Light の構文, 322
ASCII
Ultra Light ソート, 26
Ultra Light の構文, 323
ASCII ファイル
Ultra Light インポート, 488
ASIN 関数
Ultra Light の構文, 323
ATAN2 関数
Ultra Light の構文, 325
ATAN 関数
Ultra Light の構文, 324
AUTOINCREMENT
Ultra Light J 構文, 433
Ultra Light 構文, 433
AVG 関数
Ultra Light の構文, 325

B

BETWEEN 探索条件
Ultra Light SQL, 286
BIGINT データ型
Ultra Light, 296
BINARY データ型
Ultra Light, 308
BIT データ型
Ultra Light, 297
BlackBerry
ULjDbTServ ユーティリティ, 257
uljinfo ユーティリティ, 251
uljload ユーティリティ, 252
uljunload ユーティリティ, 254
Ultra Light Java Edition システムテーブル, 262
暗号化と難読化, 29
オブジェクトストアの制限事項, 13
ユーティリティ, 251
blob_file_base_dir データベースオプション
Ultra Light Java Edition の説明, 246
blob_file_base_dir プロパティ
Ultra Light Java Edition の説明, 243
BYTE_LENGTH 関数
Ultra Light の構文, 326
BYTE_SUBSTR 関数
Ultra Light の構文, 327

C

C++ API
(*参照* Ultra Light C/C++ API)
C++ アプリケーション
(*参照* Ultra Light C/C++)
cache_allocation データベースオプション
Ultra Light, 191
CACHE_MAX_SIZE 接続パラメータ

Ultra Light 構文, 166
CACHE_MIN_SIZE 接続パラメータ
Ultra Light 構文, 167
CACHE_SIZE 接続パラメータ
Ultra Light 構文, 168
case 作成パラメータ
Ultra Light 説明, 139
CASE 式
Ultra Light NULLIF 関数, 376
Ultra Light SQL 構文, 277
case プロパティ
Ultra Light 説明, 187
CAST 関数
Ultra Light の構文, 328
CE_FILE 接続パラメータ
Ultra Light 構文, 170
CEILING 関数
Ultra Light の構文, 330
Certicom
Ultra Light TLS が有効化された同期, 113
Ultra Light 暗号化モジュール, 29
CHAR_LENGTH 関数
Ultra Light の構文, 332
char_set プロパティ
Ultra Light 説明, 187
CHARINDEX 関数
Ultra Light の構文, 332
CHAR 関数
Ultra Light の構文, 331
CHAR データ型
Ultra Light, 294
CHECK CONSTRAINTS 句
Ultra Light LOAD TABLE 文, 447
CheckpointStore
Ultra Light 同期パラメータ, 90
CHECKPOINT 文
Ultra Light 構文, 426
checksum_level 作成パラメータ
Ultra Light 説明, 140
checksum_level パラメータ
Ultra Light 説明, 187
COALESCE 関数
Ultra Light の構文, 333
collation 作成パラメータ
Ultra Light 説明, 141
collation プロパティ
Ultra Light 説明, 187
commit_flush_count データベースオプション
Ultra Light, 192
commit_flush_count プロパティ
Ultra Light 説明, 187
commit_flush_timeout データベースオプション
Ultra Light, 193
commit_flush_timeout プロパティ
Ultra Light 説明, 187
COMMIT_FLUSH 接続パラメータ
Ultra Light 構文, 171
COMMIT 文
Ultra Light 構文, 427
COMPUTES 句
Ultra Light LOAD TABLE 文, 447
connCount プロパティ
Ultra Light 説明, 187
CONVERT 関数
Ultra Light の構文, 334
CON 接続パラメータ
Ultra Light 構文, 172
COS 関数
Ultra Light の構文, 336
COT 関数
Ultra Light の構文, 337
COUNT_UPLOAD_ROWS 関数
構文, 338
COUNT 関数
Ultra Light の構文, 338
count 操作
Ultra Light 実行プラン, 473
CPU
Ultra Light 制限事項, 9
CREATE INDEX 文
Ultra Light 構文, 428
Ultra Light の例, 59
UNIQUE パラメータ, 428
CREATE PUBLICATION 文
Ultra Light 構文, 430
CREATE SYNCHRONIZATION PROFILE 文
Ultra Light 構文, 431
CREATE TABLE 文
Ultra Light J 構文, 432
Ultra Light 構文, 432
CREATE USER 文
Ultra Light 動的 SQL 構文, 439
CROSS JOIN 句
Ultra Light 構文, 443
CSV ファイル
Ultra Light インポート, 488

CURRENT_TIMESTAMP 特別値
Ultra Light 構文, 273
CURRENT DATE 関数
Ultra Light TODAY 関数, 408
CURRENT DATE 特別値
Ultra Light 構文, 271
CURRENT TIMESTAMP 特別値
Ultra Light 機能比較, 3
Ultra Light 構文, 273
CURRENT TIME 特別値
Ultra Light 構文, 272
CURRENT UTC TIMESTAMP 特別値
Ultra Light 構文, 273
CustDB
Ultra Light アプリケーション readme ファイル, 16
Ultra Light サンプルファイルのロケーション, 16
Ultra Light の説明, 14
custdb.db
ロケーション, 16
custdb.sql
同期スクリプトの呼び出し, 129
custdb.udb
Ultra Light ロケーション, 16

D

database_name プロパティ
Ultra Light Java Edition の説明, 243
DATALENGTH 関数
Ultra Light の構文, 339
Data Manager
Ultra Light データベースの記憶領域, 37
date_format 作成パラメータ
Ultra Light 説明, 142
date_format データベースオプション
Ultra Light Java Edition の説明, 246
date_format プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
date_order 作成パラメータ
Ultra Light 説明, 145
date_order データベースオプション
Ultra Light Java Edition の説明, 247
date_order プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
DATEADD 関数
Ultra Light の構文, 341
DATEDIFF 関数
Ultra Light の構文, 342
DATEFORMAT 関数
Ultra Light の構文, 343
DATENAME 関数
Ultra Light の構文, 344
DATEPART 関数
Ultra Light の構文, 345
datetime
Ultra Light の変換関数, 314
DATETIME 関数
Ultra Light の構文, 345
DATETIME データ型
Ultra Light, 304
DATE 関数
Ultra Light の構文, 340
DATE データ型
Ultra Light, 303
DAYNAME 関数
Ultra Light の構文, 346
DAYS 関数
Ultra Light の構文, 347
DAY 関数
Ultra Light の構文, 346
DB_PROPERTY 関数
Ultra Light の構文, 348
DBF 接続パラメータ
Ultra Light 構文, 173
dbisql.com
説明, 202
dbisql.exe
インストール前の停止, 485
説明, 202
dbisql ユーティリティ
Ultra Light 構文, 197
Ultra Light 終了コード, 202
Ultra Light データインポートのトラブルシューティング, 488
DBKEY 接続パラメータ
Ultra Light 構文, 175
dblgen16.dill
Ultra Light での ActiveSync コンジットの配備, 125
DBN 接続パラメータ
Ultra Light 構文, 176
DDL

Ultra Light スキーマの変更, 49
DECIMAL データ型
Ultra Light, 297
DEFAULTS 句
Ultra Light LOAD TABLE 文, 447
DEFAULT TIMESTAMP カラム
Ultra Light J 構文, 433
Ultra Light 構文, 433
DEGREES 関数
構文, 349
DELIMITED BY 句
Ultra Light LOAD TABLE 文, 448
device 接続パラメータ
Ultra Light 構文, 178
DIFFERENCE 関数
Ultra Light の構文, 350
DisableConcurrency
Ultra Light 同期パラメータ, 90
Disable Concurrency
Ultra Light 同期パラメータの概要, 475
DISTINCT キーワード
Ultra Light SQL, 451
distinct 操作
Ultra Light 実行プラン, 473
DOUBLE データ型
Ultra Light, 298
download_only 同期パラメータ
Ultra Light リファレンス, 94
Download Only
Ultra Light 同期パラメータ, 94
DOW 関数
Ultra Light の構文, 350
DROP INDEX 文
Ultra Light J 構文, 440
Ultra Light 構文, 440
DROP PUBLICATION 文
Ultra Light 構文, 441
DROP SYNCHRONIZATION PROFILE 文
Ultra Light J 構文, 441
Ultra Light 構文, 441
DROP TABLE 文
Ultra Light J 構文, 442
Ultra Light 構文, 442
DROP USER 文
Ultra Light 構文, 443
dummy 操作
Ultra Light 実行プラン, 473

E

ELSE
Ultra Light CASE 式, 277
Ultra Light IF 式, 277
Embedded SQL
(*参照* Ultra Light Embedded SQL)
Ultra Light NULL 値, 271
Embedded SQL ライブラリ関数 (*参照* Ultra Light Embedded SQL ライブラリ関数)
ENCODING 句
Ultra Light LOAD TABLE 文, 448
encryption プロパティ
Ultra Light 説明, 187
END
Ultra Light CASE 式, 277
ENDIF
Ultra Light IF 式, 277
ERRORLEVEL 環境変数
Ultra Light Interactive SQL リターンコード, 202
ER 図
Ultra Light 説明, 56
[ER 図] タブ
Ultra Light 説明, 56
ER タブ
Ultra Light 使用, 56
ESCAPES オプション
Ultra Light LOAD TABLE 文, 448
ESQL (*参照* Ultra Light SQL)
ESQL (*参照* Embedded SQL)
EXISTS 探索条件
Ultra Light SQL, 286
Ultra Light 探索条件, 281
EXPLANATION 関数
Ultra Light の構文, 352
EXP 関数
Ultra Light の構文, 351

F

file プロパティ
Ultra Light 説明, 187
filter 操作
Ultra Light 実行プラン, 473
FIPS
Ultra Light fips 作成パラメータ, 146
Ultra Light fips プロパティの使用法, 29
Ultra Light TLS が有効化された同期, 113

Ultra Light 暗号化されたデータベースの配備, 18
Ultra Light 設定と配備, 147
FIPS 作成パラメータ
Ultra Light 説明, 146
fips データベースプロパティ
Ultra Light 使用法, 29
Ultra Light 配備手順, 18
fips ネットワークプロトコルオプション
Ultra Light TLS が有効化された同期, 113
FIRST 句
Ultra Light J SELECT 文, 451
Ultra Light SELECT 文, 451
FLOAT データ型
Ultra Light, 299
FLOOR 関数
Ultra Light の構文, 352
FORCE ORDER 句
Ultra Light SELECT 文, 452
FORMAT 句
Ultra Light LOAD TABLE 文, 448
FOR READ ONLY 句
Ultra Light ダイレクトページスキャン, 471
FOR 句
Ultra Light SELECT 文, 452
FROM 句
Ultra Light LOAD TABLE 文, 447
Ultra Light SELECT 文, 451
Ultra Light 構文, 443

G

GETDATE 関数
Ultra Light の構文, 353
現在のシステム日付のクエリ, 354
GetLastIdentity メソッド
Ultra Light 同期, 72
getScriptVersion メソッド
Ultra Light 例, 111
getStream メソッド
Ultra Light 例, 105
getUploadOK メソッド
Ultra Light 例, 108
global_database_id オプション
Ultra Light, 194
Ultra Light CREATE TABLE 文, 433
Ultra Light J CREATE TABLE 文, 433
Ultra Light 設定, 71
global_database_id プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
GREATER 関数
Ultra Light の構文, 354
GROUP BY 句
Ultra Light SELECT 文, 452
group-by 操作
Ultra Light 実行プラン, 473
GUID
Mobile Link システム内の Ultra Light クライアント, 70
Ultra Light NEWID 関数の SQL 構文, 374
Ultra Light STRTOUUID 関数の SQL 構文, 402
Ultra Light UUIDTOSTR 関数の SQL 構文, 411

H

HAVING 句
Ultra Light 探索条件, 281
HEXTOINT 関数
Ultra Light の構文, 354
HOURS 関数
Ultra Light の構文, 356
HOUR 関数
Ultra Light の構文, 355

I

ID
ml_remote_id オプション, 195
Ultra Light グローバルデータベース, 194
IDENTIFIED BY password 句
ALTER USER 文, 426
IF NOT EXISTS 句
CREATE PUBLICATION 文 [Ultra Light], 430
DROP INDEX 文 [Ultra Light] [Ultra Light J], 440
DROP PUBLICATION 文 [Ultra Light] [Ultra Light J], 440
DROP SYNCHRONIZATION PROFILE 文 [Ultra Light] [Ultra Light J], 440
DROP TABLE 文 [Ultra Light] [Ultra Light J], 440
Ultra Light、CREATE TEXT INDEX 文, 428
IFNULL 関数
Ultra Light の構文, 357
IF 式
Ultra Light SQL の構文, 277
ignored_rows 同期パラメータ

Ultra Light リファレンス, 95
Ignored Rows
Ultra Light 同期パラメータ, 95
index-scan 操作
Ultra Light 実行プラン, 473
INNER JOIN 句
Ultra Light 構文, 443
INPUT 文
Ultra Light トラブルシューティング, 488
INSERTSTR 関数
Ultra Light の構文, 358
INSERT 文
Ultra Light Interactive SQL の例, 55
Ultra Light J 構文, 445
Ultra Light 構文, 445
INTEGER データ型
Ultra Light, 300
Interactive SQL
Ultra Light コマンドライン, 197
Ultra Light テキストプラン, 472
Ultra Light データインポートのトラブルシューティング, 488
Ultra Light プランの解釈, 473
Ultra Light プランの操作, 473
Ultra Light プランの表示, 469
コマンドライン, 202
Interactive SQL ユーティリティ (dbisql)
Ultra Light 構文, 197
Ultra Light 終了コード, 202
INTTOHEX 関数
Ultra Light の構文, 358
IN 探索条件
Ultra Light SQL, 287
iPad (*参照* iOS)
iPhone (*参照* iOS)
IS
Ultra Light 論理演算子, 283
ISDATE 関数
Ultra Light の構文, 359
ISNULL 関数
Ultra Light の構文, 359
isolation_level オプション
Ultra Light, 195

J

join 操作
Ultra Light 実行プラン, 473

K

Keep Partial Download 同期パラメータ
Ultra Light リファレンス, 96
KEY JOIN 句
Ultra Light 構文, 443
keyset 操作
Ultra Light 実行プラン, 473
key 接続パラメータ
Ultra Light 構文, 175

L

LCASE 関数
Ultra Light の構文, 360
LEFT OUTER JOIN 句
Ultra Light 構文, 443
LEFT 関数
Ultra Light の構文, 361
LENGTH 関数
Ultra Light の構文, 361
LESSER 関数
Ultra Light の構文, 362
libulbase.lib
Ultra Light C++ 開発, 113
libulrsa.lib
Ultra Light C++ 開発, 113
libulrt.lib
Ultra Light C++ 開発, 113
like-scan 操作
Ultra Light 実行プラン, 473
LIKE 探索条件
Ultra Light SQL, 287
Ultra Light 大文字と小文字の区別, 288
Ultra Light 照合, 288
Ultra Light 探索条件, 281
Ultra Light パターン長, 288
Linux
Ultra Light ランタイムライブラリ, 113
LIST 関数
Ultra Light の構文, 363
LOAD TABLE 文
Ultra Light 構文, 446
LOCATE 関数
Ultra Light の構文, 363
LOG10 関数
Ultra Light の構文, 365
LOG 関数
Ultra Light の構文, 364

lojoin 操作
Ultra Light 実行プラン, 473
LONG BINARY データ型
Ultra Light, 309
LONG VARCHAR データ型
Ultra Light, 294
LOWER 関数
Ultra Light の構文, 366
LTRIM 関数
Ultra Light の構文, 366

M

max_hash_size 作成パラメータ
Ultra Light 説明, 147
max_hash_size プロパティ
Ultra Light 説明, 187
MAX 関数
Ultra Light の構文, 367
MINUTES 関数
Ultra Light の構文, 369
MINUTE 関数
Ultra Light の構文, 368
MIN 関数
Ultra Light の構文, 368
MIRROR_FILE 接続パラメータ
Ultra Light 構文, 179
ml_add_connection_script システムプロシージャ
追加, 129
ml_add_table_script システムプロシージャ
追加, 129
ML_GET_SERVER_NOTIFICATION
構文, 370
ml_remote_id オプション
Ultra Light, 195
Ultra Light プロパティの設定, 210
ml_remote_id プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
mlasdesk.dll
Ultra Light アプリケーションの配備, 125
mlasdev.dll
Ultra Light アプリケーションの配備, 125
mlasinst ユーティリティ
Ultra Light DLL ファイルとともに配備, 125
Ultra Light クライアントへのアプリケーションの登録, 127
Mobile Link
Ultra Light CREATE PUBLICATION 文, 430
Ultra Light SQL 文, 416
Ultra Light クライアント, 69
Ultra Light ユーザ ID の一意性, 195
Mobile Link ActiveSync プロバイダインストールユーティリティ (mlasinst)
Ultra Light クライアントへのアプリケーションの登録, 127
Mobile Link クライアント
Ultra Light 進行状況のカウンタ, 3
Mobile Link 同期
timestamp_increment 作成パラメータの設定, 160
Ultra Light クライアント, 69
Mobile Link ファイル転送
Ultra Light クライアントの概要, 82
Mobile Link プロジェクト
Ultra Light サンプル, 129
MOD 関数
Ultra Light の構文, 371
MONEY データ型
Ultra Light で相当する型, 312
MONTHNAME 関数
Ultra Light の構文, 372
MONTHS 関数
Ultra Light の構文, 373
MONTH 関数
Ultra Light の構文, 372

N

name プロパティ
Ultra Light 説明, 187
NATURAL JOIN 句
Ultra Light 構文, 443
nearest_century 作成パラメータ
Ultra Light 説明, 149
nearest_century データベースオプション
Ultra Light Java Edition の説明, 247
nearest_century プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
NEWID 関数
Ultra Light の構文, 374
newmobilinkpwd 同期パラメータ
Ultra Light リファレンス, 96
New Password
Ultra Light 同期パラメータ, 96

nosync テーブル
Ultra Light 非同期テーブル, 76
NOT
Ultra Light ビット処理演算子, 292
Ultra Light 論理演算子, 283
NOW 関数
Ultra Light の構文, 375
NT_FILE 接続パラメータ
Ultra Light 構文, 181
NULL
NULL 引数が指定されている場合に関数から返される, 313
Ultra Light ISNULL 関数, 359
NULLIF 関数
Ultra Light CASE 式での使用, 278
Ultra Light 説明, 376
NULL 値
Ultra Light SQL, 271
Number of Authentication Parameters
Ultra Light 同期パラメータ, 97
NUMERIC データ型
Ultra Light, 301

O

obfuscate データベース作成パラメータ
Ultra Light 説明, 150
obfuscate データベースプロパティ
Ultra Light 使用法, 29
Observer 同期パラメータ
Ultra Light 説明, 98
ON フレーズ
Ultra Light 探索条件, 281
OR
Ultra Light ビット処理演算子, 292
Ultra Light 論理演算子, 283
Oracle 統合データベース
Ultra Light の問題, 69
ORDER BY 句
Ultra Light SELECT 文, 452
Ultra Light トラブルシューティング, 489

P

page_size 作成パラメータ
Ultra Light 説明, 151
page_size プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
Partial Download Retained 同期パラメータ
Ultra Light リファレンス, 98
Password
Ultra Light 同期パラメータ, 99
Password 接続パラメータ
Ultra Light 構文, 182
PATINDEX 関数
Ultra Light の構文, 377
Ping
Ultra Light 同期パラメータ, 100
PI 関数
Ultra Light の構文, 379
POWER 関数
Ultra Light の構文, 379
precision 作成パラメータ
Ultra Light 説明, 153
precision データベースオプション
Ultra Light Java Edition の説明, 248
precision プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light せつめい, 187
PWD 接続パラメータ
Ultra Light 構文, 182

Q

QUARTER 関数
Ultra Light の構文, 380
QUOTES オプション
Ultra Light LOAD TABLE 文, 448

R

RADIANS 関数
Ultra Light の構文, 381
RAND 関数
Ultra Light J 構文, 381
構文, 381
readme ファイル
Ultra Light CustDB アプリケーション, 16
REAL データ型
Ultra Light, 301
reload.sql
Ultra Light ロード, 446
REMAINDER 関数
Ultra Light の構文, 382
REPEAT 関数
Ultra Light の構文, 383
REPLACE 関数

Ultra Light の構文, 384
REPLICATE 関数
Ultra Light の構文, 385
RESERVE_SIZE 接続パラメータ
Ultra Light 構文, 183
Resume Partial Download 同期パラメータ
Ultra Light リファレンス, 102
RIGHT OUTER JOIN 句
Ultra Light 構文, 443
RIGHT 関数
Ultra Light の構文, 385
ROLLBACK 文
Ultra Light J 構文, 450
Ultra Light 構文, 450
ROUND 関数
Ultra Light の構文, 386
rowlimit 操作
Ultra Light 実行プラン, 473
RSA 暗号化アルゴリズム
Ultra Light TLS が有効化された同期, 113
RTRIM 関数
Ultra Light の構文, 387

S

SAP HANA
Interactive SQL ユーティリティ (dbisql), 197
scale 作成パラメータ
Ultra Light 説明, 154
scale データベースオプション
Ultra Light Java Edition の説明, 248
scale プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
scan 操作
Ultra Light 実行プラン, 473
scjview
インストール前の終了, 485
SECONDS 関数
Ultra Light の構文, 388
SECOND 関数
Ultra Light の構文, 387
SELECT 文
Ultra Light J 構文, 451
Ultra Light J 動的 SQL 構文, 439
Ultra Light 結果セットのコピー, 55
Ultra Light 結果セットのトラブルシューティング, 489
Ultra Light 構文, 451
Ultra Light 動的 SQL 構文, 439
send_download_ack 同期パラメータ
Ultra Light リファレンス, 103
Send Download Acknowledgement
Ultra Light 同期パラメータ, 103
setObserver メソッド
Ultra Light 例, 98
SET OPTION 文
Ultra Light 構文, 453
setScriptVersion メソッド
Ultra Light 例, 111
setStreamParms メソッド
Ultra Light 例, 106
setStream メソッド
Ultra Light 例, 105
setUserData メソッド
Ultra Light 例, 109
SHORT_PLAN 関数
Ultra Light の構文, 389
SIGN 関数
Ultra Light の構文, 390
SIMILAR 関数
Ultra Light の構文, 391
SIN 関数
Ultra Light の構文, 391
SKIP 句
Ultra Light LOAD TABLE 文, 448
SMALLINT データ型
Ultra Light, 301
SMALLMONEY データ型
Ultra Light で相当する型, 312
SOUNDEX 関数
Ultra Light の構文, 392
SPACE 関数
Ultra Light の構文, 393
SQL
(*参照* Ultra Light SQL)
Ultra Light 演算子, 290
Ultra Light キーワード, 269
Ultra Light 式, 275
Ultra Light 識別子, 269
Ultra Light 数値, 271
Ultra Light スキーマのアップグレードエラー, 124
Ultra Light スキーマの変更, 49
Ultra Light 探索条件, 281
Ultra Light データ型, 293

Ultra Light 比較演算子, 282
Ultra Light 文のタイプ, 417
Ultra Light 変数, 293
Ultra Light 文字列, 270
Ultra Light 予約語, 269
SQL Anywhere データベース
Ultra Light とのデータベース比較, 3
SQL Anywhere データベースの変換
説明, 33
SQLCODE
Ultra Light 同時実行性の確認, 477
Ultra Light 特別値, 274
SQLCODE SQLE_LOCKED
Ultra Light 同時実行性エラー, 477
SQLE_CONVERSION_ERROR
Ultra Light アップグレードに関する警告, 124
SQLE_DATABASE_ERROR
Ultra Light データ破損, 486
SQLE_DEVICE_ERROR
Ultra Light データ破損, 486
SQLE_DOWNLOAD_CONFLICT エラー
Ultra Light 同期, 80
SQLE_MAX_ROW_SIZE_EXCEEDED
Ultra Light エラー, 50
SQLE_MEMORY_ERROR
Ultra Light データ破損, 486
SQLE_NOTFOUND
Ultra Light 同時実行性エラー, 477
SQLE_ROW_DROPPED_DURING_SCHEMA_UP
GRADE
Ultra Light アップグレードに関する警告, 122
SQLE_UNABLE_TO_CONNECT_OR_START
トラブルシューティング, 485
SQL FLAGGER
Ultra Light 使用, 203
sqlpp ユーティリティ
Ultra Light 構文, 203
SQL 関数
NULL 引数を指定した場合は NULL を返す, 313
Ultra Light 概要, 313
Ultra Light システム, 320
Ultra Light 集合, 313
Ultra Light 数値, 318
Ultra Light その他, 317
Ultra Light データ型変換, 314
Ultra Light の関数のタイプ, 313
Ultra Light 日付と時刻, 314
Ultra Light 文字列, 319
SQL 構文
Ultra Light CASE 式, 277
Ultra Light IF 式, 277
Ultra Light SQLCODE 特別値, 274
Ultra Light カラム名, 276
Ultra Light コメント, 270
Ultra Light 述部, 281
Ultra Light 定数, 276
Ultra Light 特別値, 271
Ultra Light 入力パラメータ, 280
Ultra Light の関数, 313
SQL プリプロセッサユーティリティ (sqlpp)
Ultra Light 構文, 203
SQL 文
Ultra Light, 416
Ultra Light 文のアルファベット順リスト, 416
SQRT 関数
Ultra Light の構文, 393
ST_AsBinary
関数、Ultra Lite, 394
ST_AsExtText
関数、Ultra Light, 394
ST_AsText
関数、Ultra Light, 395
ST_Distance
関数、Ultra Light, 395
ST_Equals
関数、Ultra Light, 396
ST_GEOMETRY
Ultra Light の型, 311
ST_IntersectsRect
関数、Ultra Light, 396
ST_Point
関数、Ultra Light, 397
ST_PointFromExtText
関数、Ultra Light, 397
ST_PointFromText
関数、Ultra Light, 398
ST_PointFromWKB
関数、Ultra Light, 399
ST_SRID
関数、Ultra Light, 399
ST_X
関数、Ultra Light, 400
ST_Y
関数、Ultra Light, 400
START SYNCHRONIZATION DELETE 文

Ultra Light J 構文, 454
Ultra Light 構文, 454
START 接続パラメータ
Ultra Light Windows Mobile 上の uleng16, 18
Ultra Light 構文, 184
STOP SYNCHRONIZATION DELETE 文
Ultra Light J 構文, 454
Ultra Light 構文, 454
同期パラメータ
Ultra Light stream_error, 104
Ultra Light ul_stream_error 構造体, 104
Ultra Light リファレンス, 104
stream_parms 同期パラメータ
Ultra Light リファレンス, 106
Stream Error
Ultra Light 同期パラメータ, 104
Stream Parameters
Ultra Light 同期パラメータ, 106
Stream Type
Ultra Light 同期パラメータ, 105
STRING 関数
Ultra Light の構文, 401
STRIP 句
Ultra Light LOAD TABLE 文, 449
STRTOUUID 関数
Ultra Light の構文, 402
STR 関数
Ultra Light の構文, 400
STUFF 関数
Ultra Light の構文, 403
subquery 操作
Ultra Light 実行プラン, 473
SUBSCRIBE BY 句
Ultra Light 同期の制限事項, 78
SUBSTRING 関数
Ultra Light の構文, 403
SUBSTR 関数
Ultra Light の構文, 403
SUM 関数
Ultra Light の構文, 405
SWITCHOFFSET 関数
構文, 406
Sybase Central
Ultra Light 接続のとらぶるしゅーてぃんぐ, 485
SYNC_PROFILE_OPTION_VALUE 関数
構文, 406
SYNCHRONIZE 文
Ultra Light J 構文, 455
Ultra Light 構文, 455
Sync Result
Ultra Light 同期パラメータ, 107
SYS
Ultra Light システムテーブル, 237
sysarticles システムテーブル [Ultra Light Java Edition データベース]
説明, 262
sysarticle システムテーブル [Ultra Light]
説明, 237
syscolumn システムテーブル [Ultra Light]
説明, 237
syscolumn システムテーブル [Ultra Light Java Edition データベース]
説明, 262
sysfkcol システムテーブル [Ultra Light Java Edition データベース]
説明, 264
sysforeignkey システムテーブル [Ultra Light Java Edition データベース]
説明, 263
sysindexcolumn システムテーブル [Ultra Light Java Edition データベース]
説明, 265
sysindex システムテーブル [Ultra Light]
説明, 238
sysindex システムテーブル [Ultra Light Java Edition データベース]
説明, 264
sysixcol システムテーブル [Ultra Light]
説明, 239
syspublications システムテーブル [Ultra Light Java Edition データベース]
説明, 266
syspublication システムテーブル [Ultra Light]
説明, 240
syssyncresult システムテーブル [Ultra Light]
説明, 241
systable システムテーブル [Ultra Light]
説明, 242
systable システムテーブル [Ultra Light Java Edition データベース]
説明, 266
system_error_code 値
Ultra Light 同期ストリームエラー, 104
sysuldata システムテーブル [Ultra Light Java Edition データベース]

説明, 267

T

TableOrder
Ultra Light 同期パラメータ, 90
TAN 関数
Ultra Light の構文, 407
TCP/IP
(*参照* TCP/IP 同期)
TEMP_DIR 接続パラメータ
Ultra Light 構文, 185
temp 操作
Ultra Light 実行プラン, 473
TEXT データ型
Ultra Light で相当する型, 312
THEN
Ultra Light IF 式, 277
time_format 作成パラメータ
Ultra Light 説明, 155
time_format データベースオプション
Ultra Light Java Edition の説明, 249
time_format プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
timestamp_format 作成パラメータ
Ultra Light 説明, 157
timestamp_format データベースオプション
Ultra Light Java Edition の説明, 249
timestamp_format プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
timestamp_increment 作成パラメータ
Mobile Link 同期用, 160
Ultra Light 説明, 160
timestamp_increment データベースオプション
Ultra Light Java Edition の説明, 250
timestamp_increment プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
timestamp_with_time_zone_format 作成パラメータ
Ultra Light, 161
timestamp_with_time_zone_format データベースオプション
Ultra Light Java Edition の説明, 250
timestamp_with_time_zone プロパティ
Ultra Light Java Edition の説明, 243
Ultra Light 説明, 187
TIMESTAMP WITH TIME ZONE データ型
Ultra Light, 306
TIMESTAMP データ型
Ultra Light, 305
Ultra Light カラムの制限事項, 3
TIMESTAMP 特別値
Ultra Light J TIMESTAMP カラム, 433
Ultra Light TIMESTAMP カラム, 433
Ultra Light カラムの制限事項, 3
TIME データ型
Ultra Light, 304
TINYINT データ型
Ultra Light, 302
TODATETIMEOFFSET 関数
構文, 408
TODAY 関数
Ultra Light の構文, 408
TOP 句
Ultra Light J SELECT 文, 451
Ultra Light SELECT 文, 451
TRIM 関数
Ultra Light の構文, 409
TRUNCATE TABLE 文
Ultra Light J 構文, 457
Ultra Light 構文, 457
TRUNCATE 関数
Ultra Lite 構文, 409
TRUNCNUM 関数
Ultra Light の構文, 409

U

UCASE 関数
Ultra Light の構文, 410
UDB
(*参照* Ultra Light データベース)
UID 接続パラメータ
Ultra Light 構文, 186
ul_stream_error 構造体
Ultra Light 例, 104
UL_SYNC_ALL_PUBS マクロ
Ultra Light パブリケーションリスト, 101
UL_SYNC_ALL マクロ
Ultra Light パブリケーションリスト, 101
uleng16 ユーティリティ
-ud オプション, 207
Windows Mobile への配備, 18
インプロセスデータベースサポート, 207

構文, 207
ulerase ユーティリティ
構文, 209
ULHTTPSStream 関数 [UL ESQL]
Ultra Light 同期ストリーム, 106
ULHTTPStream 関数 [UL ESQL]
Ultra Light 同期ストリーム, 106
ulinfo ユーティリティ
構文, 210
ulinit ユーティリティ
構文, 211
サポート対象外の照合の対処方法, 34
使用, 22
ULjDbTServ ユーティリティ
起動, 257
構文, 257
データベース情報の表示, 261
データベースの削除, 260
データベースの受信, 259
データベースの転送, 258
ログファイルの表示, 261
uljinfo ユーティリティ
構文, 251
uljload ユーティリティ
構文, 252
uljunload ユーティリティ
構文, 254
ulload ユーティリティ
構文, 219
使用, 23
ULSocketStream 関数
Ultra Light 同期ストリーム, 106
ULSQLCONNECT 環境変数
説明, 37
ulstop ユーティリティ
構文, 208
ulsync ユーティリティ
構文, 224
同期プロファイルオプション, 227
Ultra Light
(*参照* Ultra Light API)
(*参照* Ultra Light Embedded SQL)
(*参照* Ultra Light SQL)
(*参照* Ultra Light アプリケーション)
(*参照* Ultra Light データベース)
(*参照* Ultra Light ユーティリティ)
CustDB サンプル, 129
ER 図, 56
SQLE_MAX_ROW_SIZE_EXCEEDED エラー, 50
SQL 関数、集合, 313
SQL 関数、タイプ, 313
SQL 関数、データ型変換, 314
SQL 文のリファレンス, 416
timestamp_with_time_zone_format 作成パラメータ, 161
Unicode 文字の UTF8BIN 照合, 27
Windows Mobile の考慮事項, 17
アプリケーションとデータベースのインストール, 113
アプリケーションとデータベースの配備, 113
アーキテクチャ, 1
エラーコード, 197
エンジンとランタイム, 18
エンジンの起動, 207
エンジンの停止, 208
環境変数, 35
関数、ST_Distance, 395
関数、ST_PointFromText, 398
空間関数、ST_AsBinary, 394
空間関数、ST_AsExtText, 394
空間関数、ST_AsText, 395
空間関数、Ultra Light, 396
空間関数、ST_IntersectsRect, 396
空間関数、ST_Point, 397
空間関数、ST_PointFromExtText, 397
空間関数、ST_PointFromWKB, 399
空間関数、ST_SRID, 399
空間関数、ST_X, 400
空間関数、ST_Y, 400
サポート対象外の照合, 34
サポート対象のネットワークプロトコル, 3
システム関数, 320
説明, 1
データ変換, 314
データベース作成パラメータ, 137
データベースの配備, 121
テーブルの所有者, 269
同時実行性, 475
トラブルシューティング, 485
ユーザ, 61
ユーティリティリファレンス, 196
Ultra Light.NET
CustDB アプリケーションの構築, 17
CustDB サンプルと readme ファイル, 17
Ultra Light エンジンとランタイムサポート, 18

Ultra Light API (*参照* Ultra Light.NET API) (*参照* Ultra Light C/C++ API) (*参照* Ultra Light for AppForge API)
Ultra Light C/C++
CustDB アプリケーションの構築, 16
エンジンとランタイムサポート, 18
Ultra Light Embedded SQL
(*参照* Ultra Light Embedded SQL ライブラリ関数)
行番号, 203
プリプロセッサ, 203
文字列, 203
Ultra Light J
(*参照* Ultra Light Java Edition データベース)
(*参照* Ultra Light Java Edition 管理システム)
暗号化と難読化, 29
Ultra Light Java Edition
説明, 1
Ultra Light Java Edition 管理システム
(*参照* Ultra Light J)
(*参照* Ultra Light Java Edition データベース)
Ultra Light Java Edition データベース
(*参照* Ultra Light J)
ULjDbTServ ユーティリティ, 257
uljinfo ユーティリティ, 251
uljload ユーティリティ, 252
uljunload ユーティリティ, 254
インデックスの格納, 264
外部キーカラムの記述, 264
外部キーの記述, 263
カラム, 262
制限, 9
テーブルのカラムの記述, 265
テーブルの記述, 266
パブリケーションの格納, 266
パブリケーションの記述, 262
プロパティの格納, 267
ユーティリティ, 251
Ultra Light Java Edition データベースアンロードユーティリティ (uljunload)
構文, 254
Ultra Light Java Edition データベース転送ユーティリティ (ULjDbTServ)
起動, 257
構文, 257
転送, 258
データベース情報の表示, 261
データベースの削除, 260
データベースの受信, 259
ログファイルの表示, 261
Ultra Light Java Edition データベース情報ユーティリティ (uljinfo)
構文, 251
Ultra Light Java Edition データベースのオプション
説明, 245
Ultra Light Java Edition データベースのプロパティ
説明, 243
Ultra Light Java Edition データベースロードユーティリティ (uljload)
構文, 252
Ultra Light J SQL 文
DELETE 文の構文, 439
DROP INDEX 文の構文, 440
INSERT 文, 445
ROLLBACK 文の構文, 450
SELECT 文の構文, 451
START SYNCHRONIZATION DELETE 文の構文, 454
STOP SYNCHRONIZATION DELETE 文の構文, 454
SYNCHRONIZE 文の構文, 455
TRUNCATE TABLE 構文, 457
Ultra Light DROP TABLE 文の構文, 442
Ultra Light SQL
(*参照* Ultra Light Embedded SQL)
NULL 値, 271
SQL 関数、数値, 318
SQL 関数、その他, 317
SQL 関数、日付と時刻, 314
SQL 関数、文字列, 319
インデックスベースの最適化, 463
演算子, 290
カンマ区切りのリスト, 363
キーワード, 269
クエリのトラブルシューティング, 489
コメント, 270
式, 275
識別子, 269
時刻, 274
実行プラン, 469
数値, 271
データ型, 293
特別値, 271
日付, 274

文のパフォーマンスのチューニング, 480
変数, 293
ページベースの最適化, 471
文字列, 270
Ultra Light SQL 文
ALTER DATABASE SCHEMA FROM FILE 文の構文, 418
ALTER PUBLICATION 文の構文, 419
ALTER SYNCHRONIZATION PROFILE 文の構文, 420
ALTER TABLE 文の構文, 422
ALTER USER 文の構文, 425
CHECKPOINT 文の構文, 426
COMMIT 文の構文, 427
CREATE INDEX 文の構文, 428
CREATE PUBLICATION 文の構文, 430
CREATE SYNCHRONIZATION PROFILE 文の構文, 431
CREATE TABLE 文の構文, 432
CREATE USER 文の構文, 439
DELETE 文の構文, 439
DROP INDEX 文の構文, 440
FROM 句, 443
INSERT 文, 445
LOAD TABLE 文の構文, 446
ROLLBACK 文の構文, 450
SELECT 文の構文, 451
SET OPTION 文の構文, 453
START SYNCHRONIZATION DELETE 文の構文, 454
STOP SYNCHRONIZATION DELETE 文の構文, 454
SYNCHRONIZE 文の構文, 455
TRUNCATE TABLE 構文, 457
Ultra Light DROP TABLE 文の構文, 442
Ultra Light DROP USER 文の構文, 443
UNION 操作の構文, 458
UNIQUE パラメータ, 428
UPDATE 文の構文, 459
カテゴリ, 417
説明, 416
Ultra Light XML へのデータベースのアンロードユーティリティ (ulunload)
構文, 232
Ultra Light アプリケーション
Windows Mobile API, 17
Ultra Light アプリケーション
(*参照* Ultra Light C/C++ API)
(*参照* Ultra Light for AppForge API)
(*参照* Ultra Light.NET API)
ActiveSync プロバイダファイルの配備, 125
CustDB アプリケーションと readme ファイル, 16
FIPS 対応の配備, 147
Mobile Link によるファイルの転送, 82
SQLE_UNABLE_TO_CONNECT_OR_START のトラブルシューティング, 485
TLS が有効化された同期, 113
エラー -764, 489
エンジンロケーションの接続時の定義, 184
デバイスへの配備, 18
テンポラリファイルロケーションの接続時の定義, 185
同期, 69
同時実行性, 475
パブリック証明書へのアクセス, 226
複数の要求の管理, 475
ベンチマーク, 479
Ultra Light インデックスの追加
説明, 59
Ultra Light エンジン
Windows Mobile への配備, 18
エラー -764 のトラブルシューティング, 489
エラーコード, 197
起動ユーティリティ, 207
実行プログラムロケーションの接続時の定義, 184
説明, 18
停止ユーティリティ, 208
テンポラリファイルロケーションの接続時の定義, 185
データベース消去ユーティリティ, 209
デーモンとして実行, 207
同時実行性, 475
トラブルシューティング, 485
Ultra Light エンジン起動ユーティリティ
構文, 207
Ultra Light エンジン停止ユーティリティ (ulstop)
構文, 208
Ultra Light オプティマイザ
実行プランのアクセスオプション, 469
Ultra Light 管理ツール
トラブルシューティング, 488
Ultra Light クライアント
Mobile Link の説明, 69
Ultra Light クライアント同期

パラメータとオプション, 89
Ultra Light 作成パラメータ
説明, 137
Ultra Light システムテーブル
説明, 237
Ultra Light 情報ユーティリティ (ulinfo)
構文, 210
Ultra Light スキーマ
デバイスでのアップグレード, 122
Ultra Light 接続
トラブルシューティング, 485
Ultra Light 接続パラメータ
説明, 35
リスト, 164
Ultra Light 接続パラメータの指定
説明, 35
Ultra Light での同期の設計
説明, 74
Ultra Light テンポラリテーブル
管理, 470
Ultra Light テンポラリファイル
説明, 185
Ultra Light データベース
fips 作成パラメータを使用した暗号化, 146
Mobile Link からのモデリング, 23
Oracle のリファレンスデータベース, 69
page_size 作成パラメータ, 151
SQL Anywhere とのデータベース比較, 3
SQL Anywhere リファレンスデータベースからの作成, 33
Sybase Central からの初期化, 21
ulinit 実行後のデータ移植, 218
Ultra Light file プロパティ, 187
Ultra Light name プロパティ, 187
Ultra Light サイズの制限事項, 9
Ultra Light スキーマのアップグレードのリファレンス, 418
Windows Mobile でバックアップ, 38
Windows Mobile のファイルパス, 38
XML からの作成, 23
インデックスの格納, 238
インデックスの作成, 428
インデックスの操作, 56
インデックスのタイプ, 59
インデックスのハッシュ, 147
インデックスを作成する場合, 59
オブジェクトのコピー方法, 55
オプション設定の表示, 39
カラムの追加, 52
カラムの変更, 52
管理の基礎, 475
キャッシュサイズの調整, 461
検証, 45
コマンドプロンプトでの作成, 22
コマンドプロンプトでの初期化, 33
サイズ縮小, 487
作成, 21
作成パラメータ, 25
サポートされるインデックスのタイプ, 57
システム障害、バックアップとリカバリ, 46
消去, 209
照合順, 26
使用ステータスバイト, 476
スキーマ, 237
スキーマのアップグレード中のオブジェクト名変更, 122
スキーマの概要, 49
スキーマの変更, 49
スレッドの概要, 475
セキュリティの概要, 29
接続の概要, 35, 475
接続のトラブルシューティング, 485
接続パラメータの概要, 35
接続パラメータのリスト, 164
チェックポイントの使用, 478
デッドロック, 477
デバイスでのスキーマのアップグレード, 122
デバイスへの配備, 113
テンポラリテーブルの管理, 470
テンポラリファイル, 185
データとステータスの管理, 475
データの暗号化の配備, 18
データベース作成ウィザードで作成, 21
データベースの整合性, 476
テーブルのカラム, 239
テーブルのコピー, 54
テーブルの削除, 53
テーブルの作成, 51
テーブルのパブリッシュ, 85
テーブルのブラウズ, 54
同期の暗号化の配備, 113
同期の概要, 475
同期のカウント, 3
同期プロファイルオプション, 227
同期ユーティリティ (ulsync) の構文, 224
独立性レベル, 44

トランザクションの概要, 475
内のアーティクル, 237
パブリケーション, 240
パブリケーションの削除, 86
パブリケーションの説明, 84
ファイルパスの定義, 37
複数の管理, 475
プライマリキーのインデックス, 33
プロパティのブラウズ, 39
変換, 33
ベンチマークテスト, 479
マルチテーブルジョイン, 3
メモリの使用, 476
ユニークキー, 59
ユーザ ID とパスワード, 61
ユーザの管理, 62, 64, 65, 66
ユーティリティリファレンス, 196
要求の概要, 475
リカバリ, 476
ローサイズの制限, 50
ローステータスの保持, 476
ローの削除, 476
ローのパック, 151
ローのパックの概要, 50
ローのパブリッシュ, 86
ローのフェッチ, 44
ローのロック, 477
ロールバック, 476
Ultra Light データベース管理システム
説明, 1
Ultra Light データベース検証ユーティリティ (ulvalid)
説明, 235
Ultra Light データベース消去ユーティリティ
構文, 209
Ultra Light データベース初期化ユーティリティ (ulinit)
構文, 211
Ultra Light データベース設定の表示
説明, 39
Ultra Light データベース同期ユーティリティ (ulsync)
同期プロファイルオプション, 227
Ultra Light データベースの作成
説明, 21
Ultra Light データベースプロパティ
説明, 187
Ultra Light データベースへの XML のロードユーティリティ
構文, 219
Ultra Light データベースへの接続
説明, 35
Ultra Light テーブル
作成プロセスからの除外, 211
スキーマのアップグレード中の名前変更, 122
Ultra Light テーブルの削除
説明, 53
Ultra Light テーブルの作成
説明, 51
Ultra Light と SQL Anywhere との比較
説明, 3
Ultra Light 同期
説明, 69
リモート ID とユーザ ID, 195
Ultra Light 同期ユーティリティ (ulsync)
構文, 224
Ultra Light のアクセスプランの解釈
説明, 473
Ultra Light のアップグレード
接続のトラブルシューティング, 485
Ultra Light のカラム
スキーマのアップグレード中の名前変更, 122
Ultra Light のカラム定義の変更
説明, 52
Ultra Light の最適化
クエリパフォーマンス, 461
Ultra Light の式
説明, 275
Ultra Light システムテーブルのリファレンス
説明, 237
Ultra Light のデータ型
説明, 293
Ultra Light パスワード
説明, 61
Ultra Light パブリケーション
スキーマのアップグレード中の名前変更, 122
Ultra Light プラグイン
トラブルシューティング, 485
Ultra Light ベンチマークテスト
説明, 479
Ultra Light ユーザ ID
Mobile Link の一意性, 195
説明, 61
Ultra Light ユーザの認証
説明, 61

Ultra Light ユーティリティ
Ultra Light Interactive SQL (dbisql) 構文, 197
Ultra Light SQL プリプロセッサ (sqlpp) の構文, 203
Ultra Light エンジン (uleng16) の構文, 207
Ultra Light エンジン起動 (uleng16) の構文, 207
Ultra Light エンジン停止 (ulstop) の構文, 208
Ultra Light 情報 (ulinfo) の構文, 210
Ultra Light データベース検証 (ulvalid) の構文, 235
Ultra Light データベース消去 (ulerase) の構文, 209
Ultra Light データベース初期化 (ulinit) の構文, 211
Ultra Light データベースのアンロード (ulunload) の構文, 232
Ultra Light データベースへの XML のロード (ulload) の構文, 219
Ultra Light 同期 (ulsync) の構文, 224
説明, 196
Ultra Light ランタイム
説明, 18
同時実行性, 475
Ultra Light ランタイムライブラリ
Linux, 113
Windows, 113
Windows Mobile, 113
変更
Ultra Lite, 334
Ultra Light Java Edition データベース
(*参照* Ultra Light Java Edition 管理システム)
ulunload ユーティリティ
構文, 232
ulvalid ユーティリティ
構文, 235
Unicode 文字
Ultra Light 照合, 27
union-all 操作
Ultra Light 実行プラン, 473
UNION 操作
Ultra Light 構文, 458
UNION 文
Ultra Light 構文, 458
UNIQUE
Ultra Light CREATE INDEX パラメータ, 428
UNIQUEIDENTIFIER データ型
Ultra Light, 309
UPDATE 文
Ultra Light 構文, 459
UpgradeSchemaFromFile メソッド
スキーマのアップグレードの Ultra Light 代替, 122
upload_ok 同期パラメータ
Ultra Light リファレンス, 108
upload_only 同期パラメータ
Ultra Light リファレンス, 108
Upload OK
Ultra Light 同期パラメータ, 108
Upload Only
Ultra Light 同期パラメータ, 108
UPPER 関数
Ultra Light の構文, 411
user_data
Ultra Light 同期パラメータ, 109
user_name
Ultra Light 同期パラメータ, 110
User Data
Ultra Light 同期パラメータ, 109
User Name
Ultra Light 同期パラメータ, 110
utf8_encoding 作成パラメータ
Ultra Light 説明, 163
utf8_encoding データベースプロパティ
Ultra Light 使用法, 27
UTF8BIN 照合
Ultra Light 考慮事項, 27
UUID
Ultra Light NEWID 関数の SQL 構文, 374
Ultra Light STRTOUUID 関数の SQL 構文, 402
Ultra Light UUIDTOSTR 関数の SQL 構文, 411
UUIDTOSTR 関数
Ultra Light の構文, 411

V

VARBINARY データ型
Ultra Light, 310
VARCHAR データ型
Ultra Light, 295
Version
Ultra Light 同期パラメータ, 111
Version 同期パラメータ
Ultra Light リファレンス, 111
Visual Studio
Ultra Light CustDB アプリケーションの構築, 16

W

WEEKS 関数
Ultra Light の構文, 412
WHEN
Ultra Light CASE 式, 277
WHERE 句
CREATE PUBLICATION 文 [Mobile Link] [SQL Remote], 430
Ultra Light CREATE PUBLICATION 文, 430
Ultra Light DELETE 文, 440
Ultra Light SELECT 文, 452
Ultra Light UPDATE 文, 459
Ultra Light 探索条件, 281
Ultra Light 同期の制限事項, 78
Ultra Light パブリケーションの使用法, 86
Windows
(*参照* Windows ME)
(*参照* Windows Mobile 5)
(*参照* Windows Vista)
(*参照* Windows XP/200x)
Ultra Light 文字セット, 27
Ultra Light ランタイムライブラリ, 113
Windows Mobile
Ultra Light.NET を使用した CustDB アプリケーションの構築, 17
Ultra Light ActiveSync の配備, 125
Ultra Light FIPS の有効化, 147
Ultra Light Mobile Link クライアント, 87
Ultra Light uleng16 の配備, 18
Ultra Light エンジンとランタイムサポート, 18
Ultra Light エンジンの配備, 18
Ultra Light の考慮事項, 17
Ultra Light ファイルパスプレフィクス, 38
Ultra Light 文字セット, 27
Ultra Light ランタイムライブラリ, 113
エラー -764 のトラブルシューティング, 489
Windows Mobile
API の選択, 17
Windows デスクトップ
Ultra Light データベース, 38
WITH CHECKPOINT 句
Ultra Light LOAD TABLE 文, 449

X

XML
Ultra Light データベースのアンロード, 224
Ultra Light データベースのソース, 23
データベースへのロード, 219
XML データ型
Ultra Light で相当する型, 312

Y

YEARS 関数
Ultra Light の構文, 414
YEAR 関数
Ultra Light の構文, 414
YMD 関数
Ultra Light の構文, 415

あ

あいまいな文字列から日付への変換
Ultra Light, 149
値
Ultra Light インデックスのハッシュ, 147
圧縮済みカラム
Ultra Light ALTER TABLE 文, 422
アップグレード
Ultra Light SQL ALTER DATABASE SCHEMA FROM FILE 構文, 418
Ultra Light スキーマ処理, 122
Ultra Light スキーマのエラーコールバック, 124
アップロードのみの同期
Ultra Light upload_only 同期パラメータ, 108
Ultra Light データベース, 108
アプリケーション
(*参照* Ultra Light アプリケーション)
アプリケーションプロファイリング
Ultra Light ベンチマークテスト, 479
暗号化
Ultra Light Certicom モジュール, 29
Ultra Light encryption プロパティ, 187
Ultra Light fips 作成パラメータ, 146
Ultra Light fips の使用法, 29
Ultra Light fips プロパティの使用法, 29
Ultra Light J 開発, 29
Ultra Light obfuscate 作成パラメータ, 150
Ultra Light obfuscate プロパティの使用法, 29
Ultra Light TLS 同期の設定, 113
Ultra Light 暗号化キー, 175
Ultra Light キーの変更, 146
Ultra Light データの配備に関する考慮事項, 18
Ultra Light 同期の配備に関する考慮事項, 113
Ultra Light 配備手順, 18

暗号化キー
Ultra Light 変更, 146
アンパックされたロー
Ultra Light 説明, 50
アンロード
Ultra Light ulunload を使用してデータベースをアンロード, 232
Ultra Light データベース, 232
アーキテクチャの図
Ultra Light, 1
アークコサイン関数
Ultra Light ACOS 関数, 321
アークサイン関数
Ultra Light ASIN 関数, 323
アークタンジェント関数
Ultra Light の ATAN 関数, 324
アーティクル
Ultra Light コピー方法, 55
Ultra Light データベース, 78
Ultra Light の制限, 78

い

一意性制約
Ultra Light CREATE TABLE 文, 433
Ultra Light J CREATE TABLE 文, 433
Ultra Light コピー方法, 55
Ultra Light 特性, 57
イベント通知
Ultra Light 操作, 40
インジケータ
Ultra Light SQL ブロック内のコメント, 270
インストール
Ultra Light トラブルシューティング, 485
Ultra Light のデバイスへの, 113
インデックス
sysindexcolumn、Ultra Light Java Edition システムテーブル, 265
sysindex、Ultra Light Java Edition システムテーブル, 264
Ultra Light page_size の使用法, 151
Ultra Light sysindex システムテーブル, 238
Ultra Light sysixcol システムテーブル, 239
Ultra Light UNIQUE SQL パラメータ, 428
Ultra Light 概要, 463
Ultra Light コピー方法, 55
Ultra Light 削除, 60
Ultra Light 作成, 59
Ultra Light 使用の省略, 471
Ultra Light スキャンされるインデックスの決定, 469
Ultra Light 制限事項, 9
Ultra Light 操作, 56
Ultra Light タイプ, 59
Ultra Light で作成する場合, 59
Ultra Light で使用する場合, 58
Ultra Light ハッシュ値, 147
Ultra Light ハッシュの考慮事項, 147
Ultra Light パフォーマンスの向上, 464
Ultra Light プライマリキー, 33
Ultra Light ユニークキーの特性, 59
Ultra Light ユニークでないインデックスの特性, 59
Ultra Light ユニークなインデックスの特性, 59
インデックス作成ウィザード
Ultra Light 使用, 59
インデックススキャン
Ultra Light, 463
インデックスタイプの選択
Ultra Light 説明, 59
インデックスの削除
Ultra Light 説明, 60
インデックスの操作
Ultra Light 説明, 56
インデックスパフォーマンスの考慮事項
Ultra Light 説明, 147
インデックスベースの Ultra Light 最適化
Ultra Light 説明, 463
インプロセスランタイム
Ultra Light 説明, 18

え

永続的なメモリ
Ultra Light データベースの記憶領域, 37
エイリアス
Ultra Light カラム, 451
Ultra Light で相当する型, 312
エクスポート
Ultra Light ulunload を使用してデータベースをエクスポート, 232
エクスポートツール
Ultra Light ulunload ユーティリティ, 232
エスケープシーケンス
Ultra Light エンジンのパス, 489
エラー

Ultra Light SQLE_DATABASE_ERROR, 486
Ultra Light SQLE_DEVICE_ERROR, 486
Ultra Light SQLE_MEMORY_ERROR, 486
Ultra Light クライアントエラー -764, 489
エラーコード
Ultra Light データの XML へのアンロードユーティリティ (ulunload), 197
Ultra Light データベース同期ユーティリティ (ulsync), 197
Ultra Light データベースへの XML のロードユーティリティ (ulload), 197
Ultra Light ユーティリティ, 197
エラーコールバック
Ultra Light スキーマのアップグレードエラー, 124
エンコード
Ultra Light utf8_encoding 作成パラメータ, 163
Ultra Light utf8_encoding の使用法, 27
演算子
Ultra Light SQL 構文, 290
Ultra Light 演算子の優先度, 293
Ultra Light 算術演算子, 291
Ultra Light 比較演算子, 282
Ultra Light ビット処理演算子, 292
Ultra Light 文字列演算子, 292
Ultra Light 論理演算子, 283
演算子の優先度
Ultra Light SQL 構文, 293
演算の順序
Ultra Light SQL 演算子の優先度, 293
エンジン
(*参照* Ultra Light エンジン)

お

大文字と小文字の区別
LIKE 探索条件, 290
Ultra Light case 作成パラメータ, 139
Ultra Light case プロパティ, 187
Ultra Light 比較演算子, 282
Ultra Light 文字列, 270
大文字と小文字の区別の考慮事項
Ultra Light 説明, 139
大文字の文字
Ultra Light UPPER 関数, 411
大文字の文字列
Ultra Light UCASE 関数, 410
Ultra Light UPPER 関数, 411
オプション
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
Ultra Light Java Edition, 245
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light 情報ユーティリティ (ulinfo), 210
Ultra Light データベース検証ユーティリティ (ulvalid), 235
Ultra Light データベース消去ユーティリティ (ulerase), 209
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
Ultra Light 同期プロファイル, 227
Ultra Light 同期ユーティリティ (ulsync), 224
Ultra Light ブラウズ, 39
オプション (Ultra Light)
cache_allocation, 191
isolation_levelcache_allocation, 195
オプション (Ultra Light)
commit_flush_count, 192
commit_flush_timeout, 193
DB_PROPERTY 関数, 348
global_database_id, 194
ml_remote_id, 195
SET OPTION 文, 453
オプティマイザ
(*参照* クエリオプティマイザ)
Ultra Light 上書き, 469
Ultra Light 影響, 469
Ultra Light 実行プランのアクセスオプション, 469
Ultra Light 使用, 469
Ultra Light プランの解釈, 473
Ultra Light プランの操作, 473
オートコミット
Ultra Light トランザクションの概要, 477
オーバーフローエラー
AVG 関数, 326
SUM 関数, 405
オーバヘッド
Ultra Light 予約サイズの検討, 183

か

外部キー
- Ultra Light Java Edition システムテーブル, 263
- Ultra Light J 外部キー, 433
- Ultra Light J 名前のない, 433
- Ultra Light 外部キー, 433
- Ultra Light コピー方法, 55
- Ultra Light 特性, 57
- Ultra Light 名前のない, 433

外部キーカラム
- Ultra Light Java Edition システムテーブル, 264

外部キー循環
- Ultra Light 説明, 79
- Ultra Light の問題, 79

外部参照
- Ultra Light サブクエリ, 279

カスケード更新
- Ultra Light 制限事項, 3

カスケード削除
- Ultra Light 制限事項, 3

仮想ファイルシステム (*参照* VFS)

カタログ
- Ultra Light システムテーブル, 237

空のデータベース
- Ultra Light ulinit 実行後のデータ移植, 218

カラム
- Ultra Light ALTER TABLE 文, 422
- syscolumn、Ultra Light Java Edition システムテーブル, 262
- Ultra Light エイリアス, 451
- Ultra Light コピー方法, 55
- Ultra Light 制限事項, 9
- Ultra Light 追加方法, 52
- Ultra Light 変更の使用, 52

カラム式
- Ultra Light SQL ALTER TABLE 文, 422

カラム名
- Ultra Light SQL の構文, 276

環境変数
- Ultra Light ERRORLEVEL, 202
- Ultra Light ULSQLCONNECT, 37
- Ultra Light の使用法, 35

関数
- NULL 引数を指定した場合は NULL を返す, 313
- Ultra Light 概要, 313
- Ultra Light システム SQL, 320
- Ultra Light 集合, 313
- Ultra Light 数値, 318
- Ultra Light その他, 317
- Ultra Light データ型変換 SQL, 314
- Ultra Light の関数のタイプ, 313
- Ultra Light 日付と時刻, 314
- Ultra Light 文字列, 319

関数、システム
- ML_GET_SERVER_NOTIFICATION, 370
- SYNC_PROFILE_OPTION_VALUE, 406
- Ultra Light DATALENGTH, 339
- UltraLite DB_PROPERTY, 348

関数、集合
- Ultra Light AVG, 325
- Ultra Light COUNT, 338
- Ultra Light LIST, 363
- Ultra Light MAX, 367
- Ultra Light MIN, 368
- Ultra Light SUM, 405
- 説明, 313

関数、数値
- COUNT_UPLOAD_ROWS, 338
- DEGREES, 349
- RAND, 381
- Ultra Light ABS, 321
- Ultra Light ACOS, 321
- Ultra Light ASIN, 323
- Ultra Light ATAN, 324
- Ultra Light ATAN2, 325
- Ultra Light CEILING, 330
- Ultra Light COS, 336
- Ultra Light COT, 337
- Ultra Light EXP, 351
- Ultra Light FLOOR, 352
- Ultra Light J RAND, 381
- Ultra Light LOG, 364
- Ultra Light LOG10, 365
- Ultra Light MOD, 371
- Ultra Light PI, 379
- Ultra Light POWER, 379
- Ultra Light RADIANS, 381
- Ultra Light REMAINDER, 382
- Ultra Light ROUND, 386
- Ultra Light SIGN, 390
- Ultra Light SIN, 391
- Ultra Light SQRT, 393
- Ultra Light TAN, 407
- Ultra Light TRUNCATE, 409

Ultra Light TRUNCNUM, 409
Ultra Light 説明, 318
関数、その他
Ultra Light ARGN, 322
Ultra Light COALESCE, 333
Ultra Light EXPLANATION, 352
Ultra Light GREATER, 354
Ultra Light IFNULL, 357
Ultra Light LESSER, 362
Ultra Light NEWID, 374
Ultra Light NULLIF, 376
Ultra Light SHORT_PLAN, 389
Ultra Light 説明, 317
関数、データ型変換
Ultra Light CAST, 328
Ultra Light CONVERT, 334
Ultra Light HEXTOINT, 354
Ultra Light INTTOHEX, 358
Ultra Light ISDATE, 359
Ultra Light ISNULL, 359
Ultra Light 説明, 314
関数、日付と時刻
SWITCHOFFSET, 406
TODATETIMEOFFSET, 408
Ultra Light DATE, 340
Ultra Light DATEADD, 341
Ultra Light DATEDIFF, 342
Ultra Light DATEFORMAT, 343
Ultra Light DATENAME, 344
Ultra Light DATEPART, 345
Ultra Light DATETIME, 345
Ultra Light DAY, 346
Ultra Light DAYNAME, 346
Ultra Light DAYS, 347
Ultra Light DOW, 350
Ultra Light GETDATE, 353
Ultra Light HOUR, 355
Ultra Light HOURS, 356
Ultra Light MINUTE, 368
Ultra Light MINUTES, 369
Ultra Light MONTH, 372
Ultra Light MONTHNAME, 372
Ultra Light MONTHS, 373
Ultra Light NOW, 375
Ultra Light QUARTER, 380
Ultra Light SECOND, 387
Ultra Light SECONDS, 388
Ultra Light TODAY, 408
Ultra Light WEEKS, 412
Ultra Light YEARS, 414
Ultra Light YMD, 415
Ultra Light 説明, 314
関数、文字列
Ultra Light ASCII, 323
Ultra Light BYTE_LENGTH, 326
Ultra Light BYTE_SUBSTR, 327
Ultra Light CHAR, 331
Ultra Light CHAR_LENGTH, 332
Ultra Light CHARINDEX, 332
Ultra Light DIFFERENCE, 350
Ultra Light INSERTSTR, 358
Ultra Light LCASE, 360
Ultra Light LEFT, 361
Ultra Light LENGTH, 361
Ultra Light LOCATE, 363
Ultra Light LOWER, 366
Ultra Light LTRIM, 366
Ultra Light PATINDEX, 377
Ultra Light REPEAT, 383
Ultra Light REPLACE, 384
Ultra Light REPLICATE, 385
Ultra Light RIGHT, 385
Ultra Light RTRIM, 387
Ultra Light SIMILAR, 391
Ultra Light SOUNDEX, 392
Ultra Light SPACE, 393
Ultra Light STR, 400
Ultra Light STRING, 401
Ultra Light STRTOUUID, 402
Ultra Light STUFF, 403
Ultra Light SUBSTRING, 403
Ultra Light TRIM, 409
Ultra Light UCASE, 410
Ultra Light UPPER, 411
Ultra Light UUIDTOSTR, 411
Ultra Light 説明, 319
カンマで区切ったリスト
Ultra Light LIST 関数の構文, 363
管理
デバイスの Ultra Light, 113
管理ツール
Ultra Light トラブルシューティング, 485
Ultra Light ユーティリティリファレンス, 196
管理ユーティリティ
Ultra Light ユーティリティリファレンス, 196
カーソル

Ultra Light 現在のロー, 475
Ultra Light ダーティリード, 44

き

記憶領域
Ultra Light 制限事項, 9
Ultra Light 予約サイズ, 183
基準年の変換の考慮事項
Ultra Light 説明, 149
基礎
Ultra Light のデータベース管理, 475
機能
Ultra Light 比較リスト, 3
キャスト
Ultra Light データ型のリスト, 328
キャッシュ
Ultra Light cache_allocation プロパティ, 187
Ultra Light の調整, 461
キャッシュサイズ
Ultra Light 使用法, 151
Ultra Light のパフォーマンス, 461
キャッシュ割り当て
Ultra Light 設定, 191
協定世界時タイムスタンプ
Ultra Light CURRENT UTC TIMESTAMP, 273
行の長さ
Ultra Light sqlpp ユーティリティの出力, 203
共用体
Ultra Light 複数の select 文, 458
強力な暗号化
Ultra Light fips 作成パラメータ, 146
Ultra Light 使用法, 29
Ultra Light 配備手順, 18
キー
Ultra Light インデックスの作成, 59
Ultra Light インデックスのハッシュ, 147
Ultra Light プライマリ, 51
キーワード
Ultra Light SQL, 269

く

空間
サポートされている関数のリスト (Ultra Light), 311
空間 SQL API
Ultra Light の ST_Geometry 型, 311
空間関数
ST_AsBinary、Ultra Light, 394
ST_AsExtText、Ultra Light, 394
ST_AsText、Ultra Light, 395
ST_Distance、Ultra Light, 395
ST_Equals、Ultra Light, 396
ST_IntersectsRect、Ultra Light, 396
ST_PointFromExtText、Ultra Light, 397
ST_PointFromText、Ultra Light, 398
ST_PointFromWKB、Ultra Light, 399
ST_Point、Ultra Light, 397
ST_SRID、Ultra Light, 399
ST_X、Ultra Light, 400
ST_Y、Ultra Light, 400
空間データ
概要 (Ultra Light), 310
説明 (Ultra Light), 310
空間データ型
ST_GEOMETRY (Ultra Light), 311
クエリ
Ultra Light 最適化, 452
Ultra Light 予測できない結果セットのトラブルシューティング, 489
クエリオプティマイザ
(*参照* オプティマイザ)
Ultra Light, 469
クエリの最適化
(*参照* オプティマイザ)
Ultra Light SQL, 469
区別
Ultra Light case 作成パラメータ, 139
クライアント
Ultra Light Mobile Link クライアント, 69
クライアントデータベース
Ultra Light オプション, 89
グラフィカルなプラン
Ultra Light でサポートしていない, 472
繰り返し不可能読み出し
Ultra Light 説明, 44
グローバルオートインクリメント
Mobile Link システム内の Ultra Light クライアント, 70
Ultra Light global_database_id, 194
Ultra Light 設定, 71
Ultra Light デフォルトの設定, 73
Ultra Light の不足範囲, 71
グローバルデータベース ID の考慮事項
Ultra Light 説明, 194
グローバルデータベース識別子

Ultra Light global_database_id, 194
Ultra Light 設定, 71
グローバルデータベース識別子の設定
Mobile Link システム内の Ultra Light クライアント, 71
グローバルユニーク識別子
Mobile Link システム内の Ultra Light クライアント, 70
Ultra Light NEWID 関数の SQL 構文, 374

け

計画
Ultra Light 同期の設計の概要, 74
計算カラム
Ultra Light 制限事項, 3
桁
Ultra Light 最大数, 153
結果セット
Ultra Light 予測できない変更のトラブルシューティング, 489
結果セットの順序指定
Ultra Light トラブルシューティング, 489
現在のロー
Ultra Light 同時実行性, 475
検査制約
Ultra Light 制限事項, 3
検証
(*参照* データベースの検証)
Ultra Light checksum_level 作成パラメータ, 140
Ultra Light ulvalid を使用したデータベースの検証, 235
Ultra Light データベース, 45

こ

語
Ultra Light キーワード, 269
Ultra Light 予約語, 269
更新
Ultra Light データベース, 476
Ultra Light ローの更新, 459
構築
Ultra Light CustDB アプリケーション, 129
構文
Ultra Light CASE 式, 277
Ultra Light CURRENT DATE 特別値, 271
Ultra Light CURRENT TIMESTAMP 特別値, 273
Ultra Light CURRENT UTC TIMESTAMP 特別値, 273
Ultra Light IF 式, 277
Ultra Light SQLCODE 特別値, 274
Ultra Light SQL CURRENT TIME 特別値, 272
Ultra Light SQL 演算子, 290
Ultra Light SQL 演算子の優先度, 293
Ultra Light SQL 関数, 313
Ultra Light SQL コメント, 270
Ultra Light SQL 入力パラメータ, 280
Ultra Light カラム名, 276
Ultra Light 算術演算子, 291
Ultra Light 述部, 281
Ultra Light 探索条件, 281
Ultra Light 定数, 276
Ultra Light 特別値, 271
Ultra Light 比較演算子, 282
Ultra Light ビット処理演算子, 292
Ultra Light 文字列演算子, 292
Ultra Light 論理演算子, 283
互換性
Ultra Light SQL, 282
コサイン関数
Ultra Light COS 関数, 336
コタンジェント関数
Ultra Light COT 関数, 337
コピー
Ultra Light テーブルのコピー方法, 54
コマンドプロンプト
Interactive SQL モード, 197
コマンドラインユーティリティ
Ultra Light Interactive SQL (dbisql) 構文, 197
Ultra Light SQL プリプロセッサ (sqlpp) の構文, 203
Ultra Light エンジン起動 (uleng16) の構文, 207
Ultra Light エンジン停止 (ulstop) の構文, 208
Ultra Light 情報 (ulinfo) の構文, 210
Ultra Light データベース検証 (ulvalid) の構文, 235
Ultra Light データベース消去 (ulerase) の構文, 209
Ultra Light データベース初期化 (ulinit) の構文, 211
Ultra Light データベースのアンロード (ulunload) の構文, 232
Ultra Light データベースへの XML のロード (ulload) の構文, 219
Ultra Light 同期 (ulsync) の構文, 224

コミット
Ultra Light COMMIT 構文, 427
Ultra Light データベースのロー, 476
Ultra Light トランザクションの概要, 477
コミットされた読み出し
Ultra Light 独立性レベル, 43
コミットされていないトランザクション
Ultra Light 概要, 477
Ultra Light 独立性レベル, 44
コミットされていない読み出し
Ultra Light 独立性レベル, 43
コミットの遅延
パフォーマンスの強化, 478
コミットフラッシュ
Ultra Light 設定, 192
コメント
Ultra Light 構文, 270
小文字の文字列
Ultra Light LCASE 関数, 360
Ultra Light LOWER 関数, 366
コードポイント
Ultra Light, 26
コールバック
Ultra Light スキーマのアップグレードエラー, 124

さ

再起動可能なダウンロード
Ultra Light Keep Partial Download, 96
Ultra Light Partial Download Retained, 98
Ultra Light Resume Partial Download, 102
最小値
Ultra Light データ範囲, 293
最大値
Ultra Light データ範囲, 293
最大長
Ultra Light ユーザ ID, 186
最適化
Ultra Light SQL, 469
Ultra Light インデックス, 466
Ultra Light クエリ, 452
Ultra Light 実行プランのアクセスオプション, 469
Ultra Light チェックポイント, 478
Ultra Light ベンチマークテスト, 479
削除
SQL 文を使用した Ultra Light ユーザ, 64
Sybase Central を使用した Ultra Light ユーザ, 66
Ultra Light DROP SYNCHRONIZATION PROFILE 文, 441
Ultra Light DROP USER 文, 443
Ultra Light J DROP SYNCHRONIZATION PROFILE 文, 441
Ultra Light J SQL DROP INDEX 文, 440
Ultra Light J SQL DROP TABLE 文, 442
Ultra Light J START SYNCHRONIZATION DELETE 文, 454
Ultra Light J TRUNCATE TABLE 文, 457
Ultra Light SQL CREATE INDEX 文, 428
Ultra Light SQL DROP INDEX 文, 440
Ultra Light SQL DROP PUBLICATION 文, 441
Ultra Light SQL DROP TABLE 文, 442
Ultra Light START SYNCHRONIZATION DELETE 文, 454
Ultra Light TRUNCATE TABLE 文, 457
Ultra Light インデックス, 60
Ultra Light カラム, 422
Ultra Light データベース, 476
Ultra Light テーブルの削除方法, 53
Ultra Light パブリケーション, 86
接続パラメータを使用した Ultra Light ユーザ, 62
作成
Mobile Link 同期モデルからの Ultra Light データベース, 23
SQL 文を使用した Ultra Light ユーザ, 64
Sybase Central を使用した Ultra Light ユーザ, 65
Ultra Light CREATE PUBLICATION 文, 430
Ultra Light CREATE TABLE 文, 432
Ultra Light ulinit を使用してデータベースを作成, 211
Ultra Light インデックス, 59
Ultra Light データベース, 21
Ultra Light テーブルの作成方法, 51
Ultra Light テーブルのパブリケーション, 85
Ultra Light パブリケーション, 78
Ultra Light リファレンスデータベース, 33
Ultra Light リモートデータベース, 70
XML からの Ultra Light データベース, 23
コマンドプロンプトでの Ultra Light データベース, 22
接続パラメータを使用した Ultra Light ユーザ, 62

リモートデータベースの中央管理を使用して、Ultra Light データベースを, 23
作成パラメータ (Ultra Light)
case, 139
checksum_level, 140
collation, 141
date_format, 142
date_order, 145
fips 作成パラメータ, 146
max_hash_size, 147
nearest_century, 149
obfuscate, 150
page_size, 151
precision, 153
scale, 154
time_format, 155
timestamp_format, 157
timestamp_increment, 160
utf8_encoding, 163
説明, 137
サブクエリ
Ultra Light SQL, 279
算術演算子
Ultra Light SQL 構文, 291
Ultra Light 演算子, 291
参照整合性
Ultra Light インデックス, 58
Ultra Light データベース, 3
Ultra Light テーブル順序, 79
サンプル
(*参照* チュートリアル)
(*参照* 例)
サンプルアプリケーション
Ultra Light の CustDB, 14

し

式
Ultra Light CASE 式, 277
Ultra Light IF 式, 277
Ultra Light SQL, 275
Ultra Light SQL 演算子の優先度, 293
Ultra Light カラム名, 276
Ultra Light サブクエリ, 279
Ultra Light 集合, 278
Ultra Light 定数, 276
Ultra Light 入力パラメータ, 280
式内のカラム名
Ultra Light 説明, 276
式内の定数
Ultra Light 説明, 276
識別子
Ultra Light SQL, 269
時刻
Ultra Light の変換関数, 314
時刻関数
Ultra Light アルファベット順リスト, 314
時刻の考慮事項
Ultra Light 説明, 155
指数
Ultra Light, 271
指数関数
Ultra Light EXP 関数, 351
システム関数
Ultra Light 制限事項, 3
Ultra Light 説明, 320
システム障害
Ultra Light トランザクションの概要, 477
Ultra Light リカバリ, 46
システム障害からのリカバリ
Ultra Light 内部メカニズム, 46
システムテーブル
Ultra Light Java Edition sysarticles, 262
Ultra Light Java Edition syscolumn, 262
Ultra Light Java Edition sysfkcol, 264
Ultra Light Java Edition sysforeignkey, 263
Ultra Light Java Edition sysindex, 264
Ultra Light Java Edition sysindexcolumn, 265
Ultra Light Java Edition syspublications, 266
Ultra Light Java Edition sysuldata, 267
Ultra Light Java Edition システム, 266
Ultra Light sysarticle, 237
Ultra Light syscolumn, 237
Ultra Light sysindex, 238
Ultra Light sysixcol, 239
Ultra Light syspublication, 240
Ultra Light syssyncresult, 241
Ultra Light systable, 242
Ultra Light 説明, 237
Ultra Light ブラウズ方法, 54
実行
Ultra Light CustDB アプリケーション, 129
実行プラン
Ultra Light インデックスの使用のチェック, 469
Ultra Light 上書き, 469

Ultra Light 解釈の方法, 473
Ultra Light 操作, 469, 473
Ultra Light テキスト, 472
Ultra Light 表示, 472
集合関数
Ultra Light アルファベット順リスト, 313
集合式
Ultra Light SQL の構文, 278
終了コード
Ultra Light Interactive SQL ユーティリティ (dbisql), 202
Ultra Light データの XML へのアンロードユーティリティ (ulunload), 197
Ultra Light データベース同期ユーティリティ (ulsync), 197
Ultra Light データベースへの XML のロードユーティリティ (ulload), 197
述部
Ultra Light ALL, 284
Ultra Light ANY, 285
Ultra Light BETWEEN, 286
Ultra Light EXISTS, 286
Ultra Light IN, 287
Ultra Light LIKE 探索条件, 287
Ultra Light SQL, 281
Ultra Light 比較演算子, 282
循環
Ultra Light 外部キー, 79
Ultra Light 外部キーの問題, 79
準備文
Ultra Light 入力パラメータ, 280
ジョイン
Ultra Light FROM 句の構文, 443
障害
Ultra Light スキーマのアップグレードエラー, 124
Ultra Light メモリ不足エラーの回避, 183
条件
Ultra Light ALL 条件, 284
Ultra Light ANY, 285
Ultra Light BETWEEN, 286
Ultra Light EXISTS, 286
Ultra Light IN, 287
Ultra Light 探索, 281
上限
Ultra Light 物理的制限, 9
照合
Ultra Light CollationName プロパティ, 187
Ultra Light collation 作成パラメータ, 141
Ultra Light サポート対象外, 34
照合順
Ultra Light LIKE 探索条件, 288
Ultra Light 説明, 26
Ultra Light 変更, 26
小数
Ultra Light, 271
小数点の位置の考慮事項
Ultra Light precision, 153
Ultra Light scale 作成パラメータ, 154
証明書
Ultra Light アプリケーションから暗号化情報へのアクセス, 226
初期化
Ultra Light ulinit を使用してデータベースを初期化, 211
進行状況のカウンタ
Ultra Light オフセットの不一致, 3
診断情報
Ultra Light ベンチマークテスト, 479
信頼できる証明書
Ultra Light アプリケーションから暗号化情報へのアクセス, 226

す

スイッチ
Ultra Light Interactive SQL ユーティリティ (dbisql), 198
Ultra Light SQL プリプロセッサユーティリティ (sqlpp), 203
Ultra Light 情報ユーティリティ (ulinfo), 210
Ultra Light データベース検証ユーティリティ (ulvalid), 235
Ultra Light データベース消去ユーティリティ (ulerase), 209
Ultra Light データベース初期化ユーティリティ (ulinit), 211
Ultra Light データベースのアンロードユーティリティ (ulunload), 232
Ultra Light データベースへの XML のロードユーティリティ (ulload), 220
Ultra Light 同期ユーティリティ (ulsync), 224
数値
Ultra Light SQL, 271
数値関数
Ultra Light アルファベット順リスト, 318

数値式
Ultra Light 算術演算子, 291
数値の精度
Ultra Light precision, 153
スキャン
Ultra Light クエリ内のインデックス, 463
Ultra Light クエリ内のデータベースページ, 471
スキーマ
Ultra Light SQL ALTER DATABASE SCHEMA FROM FILE 構文, 418
Ultra Light システムテーブル, 237
Ultra Light 操作, 49
Ultra Light テーブルのカタログ, 49
Ultra Light 変更時の注意事項, 49
スクリプトバージョン
Ultra Light getScriptVersion, 111
Ultra Light setScriptVersion メソッド, 111
Ultra Light Version 同期パラメータ, 111
スクリプトを使用したアップロード
Ultra Light CREATE PUBLICATION 構文, 430
ステータス管理
Ultra Light 概要, 475
ステータスバイト
Ultra Light データベース, 476
ストアドプロシージャ
Ultra Light 制限事項, 3
ストリーム同期パラメータ
Ultra Light 同期パラメータ, 106
スペース
Ultra Light ファイルパスの定義, 485
スラッシュ ー アスタリスク
Ultra Light コメントインジケータ, 270
スレッド
Ultra Light 同時実行性, 475

せ

制限事項
Ultra Light, 9
Ultra Light データ型, 293
整合性
Ultra Light CREATE TABLE 文, 433
Ultra Light J CREATE TABLE 文, 433
整合性制約
Ultra Light J 使用法, 433
Ultra Light 使用法, 433
制約
Ultra Light ALTER TABLE 文, 422
Ultra Light 参照整合性, 79
Ultra Light 名前の変更, 422
セキュリティ
Ultra Light, 29
Ultra Light AES_FIPS データベースの配備, 18
Ultra Light FIPS の暗号化, 146
Ultra Light TLS が有効化された同期, 113
Ultra Light 概要, 29
接続
Mobile Link Ultra Light Stream Type 同期パラメータ, 105
Ultra Light connCount プロパティ, 187
Ultra Light 概要, 35
Ultra Light データベースのトラブルシューティング, 36
Ultra Light 同時実行性, 475
Ultra Light トラブルシューティング, 485
接続パラメータ
CE, 178
iPhone, 178
Mac, 177
NT, 177
Ultra Light, 35
Ultra Light CACHE_MAX_SIZE, 166
Ultra Light CACHE_MIN_SIZE, 167
Ultra Light CACHE_SIZE, 168
Ultra Light CE_FILE, 170
Ultra Light COMMIT_FLUSH, 171
Ultra Light CON, 172
Ultra Light DBF, 173
Ultra Light DBKEY, 175
Ultra Light DBN, 176
Ultra Light device, 178
Ultra Light file_name, 173
Ultra Light key, 175
Ultra Light MIRROR_FILE, 179
Ultra Light NT_FILE, 181
Ultra Light password, 182
Ultra Light PWD, 182
Ultra Light RESERVE_SIZE, 183
Ultra Light START, 184
Ultra Light TEMP_DIR, 185
Ultra Light UID, 186
Ultra Light userid, 186
Ultra Light 概要, 35
Ultra Light 接続のまとめ, 35
Ultra Light 選択, 37

Ultra Light, 177
Ultra Light 優先度, 36
アルファベット順リスト (Ultra Light), 164
接続方法
Ultra Light 説明, 35
接続文字列
Ultra Light 接続パラメータ, 164
Ultra Light の設定, 35
Ultra Light パラメータの概要, 35
選択
Ultra Light J SELECT 文, 451
Ultra Light SELECT 文, 451
占有容量
Ultra Light データベース, 3

そ

挿入
LOAD TABLE 文を使用した Ultra Light データ, 446
Ultra Light INSERT 文, 445
Ultra Light J INSERT 文, 445
Ultra Light ローのバルク, 446
ソート順
Ultra Light 照合, 27

た

タイムスタンプ
Ultra Light timestamp_format 作成パラメータ, 157
Ultra Light timestamp_increment 作成パラメータ, 160
ダイレクトページスキャン
Ultra Light 説明, 471
ダウンロード確認
Ultra Light send_download_ack 同期パラメータ, 103
ダウンロード専用テーブル
Ultra Light 非同期テーブル, 76
ダウンロードのみの同期
Ultra Light download_only 同期パラメータ, 94
Ultra Light 定義の概要, 80
Ultra Light 同期パラメータ, 94
探索条件
Ultra Light ALL, 284
Ultra Light ANY, 285
Ultra Light BETWEEN, 286
Ultra Light EXISTS, 286
Ultra Light IN, 287
Ultra Light LIKE, 287
Ultra Light SQL, 281
ダーティリード
Ultra Light 独立性レベル, 44

ち

チェックサム
Ultra Light checksum_level 作成パラメータ, 140
Ultra Light checksum_level プロパティ, 187
チェックポイント
Ultra Light CHECKPOINT 構文, 426
Ultra Light のパフォーマンスの最適化, 478
チュートリアル
Ultra Light CustDB ファイル, 129
直列化されたトランザクション
Ultra Light での処理, 477

つ

追加
SQL 文を使用した Ultra Light ユーザ, 64
Sybase Central を使用した Ultra Light ユーザ, 65
Ultra Light インデックス, 59
Ultra Light カラム, 422
Ultra Light カラムの追加方法, 52
接続パラメータを使用した Ultra Light ユーザ, 62
追跡
Ultra Light 同期, 3

て

底が 10 の対数
Ultra Light LOG10 関数, 365
定数
Ultra Light SQL 構文, 276
テキストプラン
Ultra Light 表示, 472
デスクトップ接続パラメータ
Ultra Light 構文, 177
デスクトップでの作成
Ultra Light 説明, 21
デッドロック
Ultra Light での予防策, 477
デバイス
Ultra Light の配備, 113
Ultra Light の配備方法, 113

Ultra Light 複数の接続パラメータ, 37
デバイス上での作成
説明, 24
デフォルト
Ultra Light J オートインクリメント, 433
Ultra Light オートインクリメント, 433
デフォルト値
Ultra Light CURRENT DATE, 271
Ultra Light CURRENT TIME, 272
Ultra Light CURRENT TIMESTAMP, 273
Ultra Light CURRENT UTC TIMESTAMP, 273
Ultra Light SQLCODE, 274
デフォルトのグローバルオートインクリメントカラムの宣言
Mobile Link システム内の Ultra Light クライアント, 73
デモ
(*参照* チュートリアル)
テンポラリテーブル
Ultra Light クエリ, 472
Ultra Light 制限事項, 9
Ultra Light 説明, 470
Ultra Light 同期, 77
テンポラリテーブルの管理
Ultra Light 説明, 470
テンポラリファイル
Ultra Light 制限事項, 185
データ
Ultra Light J ローの選択, 451
Ultra Light アンロード, 232
Ultra Light 表示方法, 54
Ultra Light ロード, 219
Ultra Light ローの選択, 451
データ移植
Ultra Light ulinit を使用して作成したデータベースへのデータ移植, 218
データ型
Ultra Light 値のキャスト, 328
Ultra Light エイリアスに相当, 312
Ultra Light 制限事項, 9
Ultra Light 説明, 293
Ultra Light 特別値, 271
Ultra Light の SQL 変換関数, 314
空間 (Ultra Light), 311
データ型変換
Ultra Light CAST, 328
データ型変換関数
Ultra Light 説明, 314
データ管理
Ultra Light コンポーネント, 18
Ultra Light 説明, 1
Ultra Light のステータス情報, 475
データソース
Ultra Light CustDB チュートリアルの例, 129
データの一貫性
Ultra Light ダーティリード, 44
データのインポート
Ultra Light トラブルシューティング, 488
データのコピー
Ultra Light データベース, 54
データの削除
Ultra Light ファイルサイズへの影響, 487
データのデータベースへのインポート
Ultra Light ulload ユーティリティ, 219
データの同期
Ultra Light ファイルサイズへの影響, 487
データのロード
Ultra Light LOAD TABLE 文, 446
データベース
(*参照* Ultra Light データベース)
Ultra Light と SQL Anywhere の比較, 3
Ultra Light バルクデータのロード, 446
Ultra Light プラグインで作成, 21
データベースオブジェクト
Ultra Light コピー方法, 55
データベースオプション
(*参照* データベースオプション (Ultra Light))
Ultra Light, 191
データベースオプション (Ultra Light)
cache_allocation, 191
isolation_levelcache_allocation, 195
データベースオプション (Ultra Light)
commit_flush_count, 192
commit_flush_timeout, 193
DB_PROPERTY 関数, 348
global_database_id, 194
ml_remote_id, 195
SET OPTION 文, 453
ブラウズ, 39
[データベースオプション] ウィンドウ
アクセス, 39
データベース管理
Ultra Light レイヤ, 1
データベース検証
Ultra Light, 45
データベース検証ユーティリティ (ulvalid)

Ultra Light 説明, 235
データベースサイズ
Ultra Light 制限事項, 9
データベース作成ウィザード
Ultra Light 使用法, 21
データベース消去ユーティリティ
Ultra Light 構文, 209
データベーススキーマ
Ultra Light システムテーブル, 237
データベースのオプション
Ultra Light Java Edition, 245
データベースの管理
Ultra Light のデータとステータス, 475
データベースの検証
Ultra Light データベース検証ユーティリティ (ulvalid), 235
データベースの作成
Sybase Central Ultra Light プラグイン, 21
Ultra Light 作成パラメータ, 137
Ultra Light リモートデータベース, 70
データベースの初期化
Sybase Central Ultra Light プラグイン, 21
データベースのプロパティ
Ultra Light Java Edition, 243
データベースのページサイズの考慮事項
Ultra Light 説明, 151
データベースのロード
Ultra Light ulload を使用してデータベースをロード, 219
データベースファイル
(*参照* Ultra Light データベース)
Ultra Light 暗号化, 175
Ultra Light 最大サイズ, 9
Ultra Light 接続パラメータ, 37
データベースプロパティ
(*参照* データベースプロパティ (Ultra Light))
Ultra Light, 187
データベースプロパティ (Ultra Light)
fips の使用法, 29
obfuscate の使用法, 29
utf8_encoding の使用法, 27
作成パラメータ, 25
ブラウズ, 39
データベースユーティリティ
Ultra Light データベース接続, 35
テーブル
Ultra Light allsync テーブル, 77
Ultra Light ALTER TABLE 文, 422
Ultra Light CREATE TABLE 文, 432
Ultra Light INSERT 文, 445
Ultra Light J CREATE TABLE 文, 432
Ultra Light J INSERT 文, 445
Ultra Light J TRUNCATE TABLE 文, 457
Ultra Light nosync テーブル, 76
Ultra Light TRUNCATE TABLE 文, 457
Ultra Light コピー方法, 54, 55
Ultra Light 削除方法, 53
Ultra Light 作成方法, 51
Ultra Light 順序, 79
Ultra Light 制限事項, 9
Ultra Light 操作, 50
Ultra Light ダウンロード専用, 76
Ultra Light でのサイズ制限, 50
Ultra Light テンポラリテーブルの使用法, 472
Ultra Light パブリケーション, 78
Ultra Light パブリケーションによる同期の制御, 78
Ultra Light バルクロード, 446
Ultra Light ブラウズ方法, 54
Ultra Light 編集方法, 54
Ultra Light ローのパブリッシュ, 86
テーブルサイズ
Ultra Light 制限事項, 9
ローの数, 9
テーブル作成ウィザード
Ultra Light 使用法, 51
テーブル式
Ultra Light サブクエリ, 279
テーブル制約
Ultra Light CREATE TABLE 文, 433
Ultra Light J CREATE TABLE 文, 433
テーブル全体
Ultra Light パブリッシュ, 78
テーブル全体のパブリッシュ
Ultra Light, 85
テーブルの所有者
Ultra Light, 269
テーブルの制約
Ultra Light 追加、削除、変更, 422
デーモン
Ultra Light エンジンを実行, 207

と

同期

timestamp_increment 作成パラメータの設定, 160
Ultra Light ALTER TABLE 文, 422
Ultra Light CREATE TABLE 文, 433
Ultra Light CustDB チュートリアル, 129
Ultra Light download_only パラメータ, 94
Ultra Light J CREATE TABLE 文, 433
Ultra Light SQLE_DOWNLOAD_CONFLICT エラー, 80
Ultra Light syssyncresult システムテーブル, 241
Ultra Light Upload Only パラメータ, 108
Ultra Light アプリケーションの実装, 80
Ultra Light 外部キー, 79
Ultra Light 概要, 69
Ultra Light クライアント, 69
Ultra Light クライアント固有のデータ, 77
Ultra Light 作業の概要, 69
Ultra Light 参照整合性, 79
Ultra Light システムテーブル, 242
Ultra Light 進行状況のカウントの仕組み, 3
Ultra Light スキーマの変更, 49
Ultra Light 設計の概要, 74
Ultra Light 停止, 98
Ultra Light データベースの ulsync ユーティリティ, 224
Ultra Light パブリケーションによるテーブルの除外, 78
Ultra Light 無視されるロー, 95
Ultra Light モニタ, 98
Ultra Light 読み込み専用テーブル, 80
Windows Mobile での Ultra Light データベース, 87
同期スクリプト
Ultra Light サンプルのブラウズ, 129
同期ストリーム
Ultra Light getStream メソッド, 105
Ultra Light setStreamParms メソッド, 106
Ultra Light stream_error 同期パラメータ, 104
Ultra Light stream_parms 同期パラメータ, 106
Ultra Light stream 同期パラメータ, 105
Ultra Light ULHTTPSStream, 106
Ultra Light ULHTTPStream, 106
Ultra Light 設定, 105
同期ストリームパラメータ
Ultra Light ストリームタイプ, 105
同期の制御
Ultra Light パブリケーション, 78
同期の追加
Ultra Light アプリケーション, 80
同期のモニタ
Ultra Light Observer 同期パラメータ, 98
Ultra Light setObserver メソッド, 98
同期パラメータ
Ultra Light, 89
Ultra Light additionalparms, 90
Ultra Light Authentication Value, 94
Ultra Light Disable Concurrency の概要, 475
Ultra Light download_only, 94
Ultra Light getScriptVersion, 111
Ultra Light getStrea+m メソッド, 105
Ultra Light getUploadOK メソッド, 108
Ultra Light Keep Partial Download, 96
Ultra Light newmobilinkpwd, 96
Ultra Light Observer, 98
Ultra Light Partial Download Retained, 98
Ultra Light Password, 99
Ultra Light Ping, 100
Ultra Light Publication, 101
Ultra Light Resume Partial Download, 102
Ultra Light send_download_ack, 103
Ultra Light setObserver メソッド, 98
Ultra Light setScriptVersion メソッド, 111
Ultra Light setStreamParms メソッド, 106
Ultra Light setStream メソッド, 105
Ultra Light setUserData メソッド, 109
Ultra Light stream_parms, 106
Ultra Light Sync Result, 107
Ultra Ligh upload_ok, 108
Ultra Light upload_only, 108
Ultra Light user_data, 109
Ultra Light user_name, 110
Ultra Light Version, 111
Ultra Light ストリームタイプ, 105
Ultra Light 必須, 89
同期プロファイル
Ultra Light ALTER SYNCHRONIZATION PROFILE 文, 420
Ultra Light CREATE SYNCHRONIZATION PROFILE 文, 431
Ultra Light DROP SYNCHRONIZATION PROFILE 文, 441
Ultra Light J DROP SYNCHRONIZATION PROFILE 文, 441
Ultra Light J SYNCHRONIZE 文, 455
Ultra Light SYNCHRONIZE 文, 455
同期プロファイルオプション

概要、Ultra Light クライアント, 227
同期モデル
Ultra Light データベース, 23
同期論理
Ultra Light Sybase Central のブラウズ, 129
統合データベース
Ultra Light サンプル, 129
Ultra Light の互換性, 69
Ultra Light の選択, 3
同時実行性
Ultra Light 同期, 475
Ultra Light の問題, 475
動的 SQL
Ultra Light 演算子の優先度, 293
Ultra Light 算術演算子, 291
Ultra Light ビット処理演算子, 292
Ultra Light 文字列演算子, 292
Ultra Light 論理演算子, 283
登録
ActiveSync に対する Mobile Link Ultra Light アプリケーション, 127
特別値
Ultra Light CURRENT DATE, 271
Ultra Light CURRENT TIME, 272
Ultra Light CURRENT TIMESTAMP, 273
Ultra Light CURRENT UTC TIMESTAMP, 273
Ultra Light SQL, 271
Ultra Light SQLCODE, 274
独立性レベル
Ultra Light, 43
Ultra Light ダーティリード, 44
Ultra Light の独立性レベル, 43
トラブルシューティング
Ultra Light getUploadOK メソッド, 108
Ultra Light Ping 同期パラメータ, 100
Ultra Light Stream Error 同期パラメータ, 104
Ultra Light Sync Result 同期パラメータ, 107
Ultra Light upload_ok 同期パラメータ, 108
Ultra Light アプリケーションのバックアップ, 69
Ultra Light グローバル ID 番号, 71
Ultra Light 再開可能なダウンロードの実装, 80
Ultra Light タイムスタンプとその増分の管理, 158
Ultra Light チェックサムエラー, 140
Ultra Light での接続パラメータの優先度, 36
Ultra Light の GLOBAL AUTOINCREMENT の値の取り出し, 72
外部キー循環に関連する同期の問題の防止, 79
トランケート
Ultra Light J テーブル, 457
Ultra Light テーブル, 457
トランザクション
Ultra Light COMMIT 文, 427
Ultra Light J ロールバック, 450
Ultra Light スキーマ変更の影響, 49
Ultra Light チェックポイント, 426
Ultra Light データベース, 476
Ultra Light 同時実行性, 475
Ultra Light 独立性レベル, 44
Ultra Light フラッシュ, 192
Ultra Light ロールバック, 450
トランザクション管理
Ultra Light, 475
Ultra Light のスループット向上, 478
トランザクション処理
Ultra Light commit_flush_count オプション, 192
Ultra Light commit_flush_timeout オプション, 193
Ultra Light COMMIT_FLUSH 接続パラメータ, 171
チェックポイントとコミット, 478
トランザクションログ
Ultra Light 内部メカニズム, 46
トランスポートレイヤセキュリティ
(*参照* TLS)
トリガ
Ultra Light 制限事項, 3
取り消し
Ultra Light J トランザクション, 450
Ultra Light トランザクション, 450
ドロップ
Ultra Light インデックス, 60

な

内部参照
Ultra Light サブクエリ, 279
名前
Ultra Light カラム名, 276
名前のない外部キー
Ultra Light J 使用法, 433
Ultra Light 使用法, 433
名前の変更
Ultra Light テーブル, 422
名前変更

アップグレード中の Ultra Light データベースオブジェクト, 122
難読化
Ultra Light J 開発, 29
Ultra Light 使用法, 29

に

二重スラッシュ
Ultra Light コメントインジケータ, 270
二重ハイフン
Ultra Light コメントインジケータ, 270
入力パラメータ
Ultra Light 説明, 280
認証
Ultra Light ユーザ, 61
認証ステータス
Ultra Light 同期パラメータ, 92
認証値
Ultra Light 同期パラメータ, 94
認証パラメータ
Ultra Light 同期パラメータ, 92

ね

ネットワークプロトコル
HTTPS による Ultra Light 同期, 106
HTTP による Ultra Light 同期, 106
TCP/IP による Ultra Light 同期, 106
Ultra Light Sync Result 同期パラメータ, 107
Ultra Light のサポート対象, 3

は

排他 OR
Ultra Light ビット処理演算子, 292
バイナリ
Ultra Light ソート, 26
バイナリデータ型
Ultra Light 最大サイズ, 9
配備
Ultra Light ActiveSync プロバイダファイル, 125
Ultra Light FIPS 対応のアプリケーション, 147
Ultra Light アップグレードのデバイスへの, 113
Ultra Light エンジンのトラブルシューティング, 489
Ultra Light クライアントで ActiveSync を使用するアプリケーション, 87
Ultra Light スキーマアップグレード, 122
Ultra Light データベース, 121
Ultra Light データベースファイルへの変更, 113
Ultra Light のデバイスへの, 113
パス
Ultra Light エンジン, 485
Ultra Light 接続パラメータ, 37
パスワード
PWD Ultra Light 接続パラメータ, 182
Ultra Light Password 同期パラメータ, 99
Ultra Light 新規 Mobile Link パスワードパラメータ, 96
Ultra Light 接続パラメータのソルト値, 182
Ultra Light データベース, 61
最大長, 439
文字セット変換, 439
パスワードのソルト値
Ultra Light 構文, 182
派生テーブル
Ultra Light FROM 句, 443
Ultra Light SQL, 279
パターン一致
Ultra Light LIKE 探索条件, 287
Ultra Light PATINDEX 関数, 377
Ultra Light 大文字と小文字の区別, 288
Ultra Light 照合, 288
Ultra Light パターン長, 288
パターン長
Ultra Light LIKE 探索条件, 288
バックアップ (*参照* バックアップ)
Ultra Light 説明, 46
Ultra Light トランザクションの概要, 477
Ultra Light 内部メカニズム, 46
Windows Mobile での Ultra Light データベース, 38
パックされたロー
Ultra Light 説明, 50
ハッシュ
Ultra Light max_hash_size プロパティ, 187
Ultra Light インデックス, 147
Ultra Light サイズの考慮事項, 147
Ultra Light サイズの設定, 468
Ultra Light 最適なサイズ, 466
ハッシュサイズの設定
説明, 468
パフォーマンス
Ultra Light CACHE_MAX_SIZE 接続パラメータ, 166

Ultra Light CACHE_MIN_SIZE 接続パラメータ, 167
Ultra Light CACHE_SIZE 接続パラメータ, 168
Ultra Light FOR READ ONLY 句の指定, 469
Ultra Light アップロードのみの同期, 108
Ultra Light アプリケーションでのインデックスの使用, 33
Ultra Light インデックスの使用, 58
Ultra Light インデックスのハッシュ, 147
Ultra Light インデックスのハッシュによるクエリのチューニング, 464
Ultra Light クエリの最適化, 452
Ultra Light クエリの最適化の方法, 461
Ultra Light クエリパフォーマンス向上のためのインデックスの使用, 428
Ultra Light 最適なハッシュサイズの選択, 466
Ultra Light 実行プランの表示, 472
Ultra Light 大規模なテーブルでのインデックスの使用, 57
Ultra Light ダウンロードのみの同期, 94
Ultra Light ページサイズ, 151
Ultra Light メモリ障害の回避, 183
コミットのチェックポイントからの分離, 478
パフォーマンスのチューニング
Ultra Light max_hash_size, 147
Ultra Light、インデックスのハッシュ, 464
パブリケーション
sysarticles、Ultra Light Java Edition システムテーブル, 262
syspublications、Ultra Light Java Edition システムテーブル, 266
Ultra Light ALTER PUBLICATION 文, 419
Ultra Light CREATE PUBLICATION 文, 430
Ultra Light Java Edition データベーススキーマの説明, 266
Ultra Light J SQL DROP INDEX 文, 440
Ultra Light J SQL DROP TABLE 文, 442
Ultra Light publication 同期パラメータ, 101
Ultra Light SQL CREATE INDEX 文, 428
Ultra Light SQL DROP INDEX 文, 440
Ultra Light SQL DROP PUBLICATION 文, 441
Ultra Light SQL DROP TABLE 文, 442
Ultra Light sysarticle システムテーブル, 237
Ultra Light syspublication システムテーブル, 240
Ultra Light WHERE 句の使用法, 86
Ultra Light コピー方法, 55
Ultra Light 削除, 86
Ultra Light 制限, 211
Ultra Light 操作, 84
Ultra Light テーブルのパブリッシュ, 85
Ultra Light 同期, 101
Ultra Light 同期パラメータ, 101
Ultra Light の設計の計画, 74
Ultra Light パブリッシュの概要, 78
Ultra Light ローのパブリッシュ, 86
スキーマ内の Ultra Light Java Edition テーブルリスト, 262
パブリケーション作成ウィザード
Ultra Light 使用法, 85
Ultra Light パブリケーションの作成, 78
Ultra Light ローのパブリッシュ, 86
パブリケーションの削除
Ultra Light クライアント, 86
パブリケーションの作成
Ultra Light データベースと Mobile Link アプリケーション, 78
パブリック証明書
Ultra Light アプリケーションからのアクセス, 226
パブリッシュ
Ultra Light テーブル, 85
Ultra Light テーブル全体, 78
Ultra Light ロー, 86
パラメータ
Ultra Light SQL 入力, 280
Ultra Light 接続の概要, 35
Ultra Light 接続リスト, 164
Ultra Light データベース作成, 137
パラメータの接続文字列へのアセンブル
Ultra Light 説明, 35
バルクロード
Ultra Light LOAD TABLE 文, 446
範囲
Ultra Light データ型, 293
バージョン
Ultra Light ユーティリティのトラブルシューティング, 488

ひ

比較
Ultra Light 探索条件, 281
Ultra Light と SQL Anywhere のデータベース, 3
比較演算子
Ultra Light SQL, 282

Ultra Light 動的 SQL 構文, 282
日付
Ultra Light date_format プロパティ, 187
Ultra Light date_order プロパティ, 187
Ultra Light Java Edition の date_format プロパティ, 243
Ultra Light Java Edition の date_order プロパティ, 243
Ultra Light あいまいな文字列の変換, 149
Ultra Light 順序, 145
Ultra Light の表示形式, 142
Ultra Light の変換関数, 314
Ultra Light ロールオーバーポイント, 149
現在のシステム日付のクエリ, 354
日付関数
Ultra Light アルファベット順リスト, 314
日付の考慮事項
Ultra Light 説明, 142
日付の単位
Ultra Light で使用可能, 142
説明, 314
ビット処理演算子
Ultra Light SQL 構文, 292
表示
Ultra Light テーブルの表示方法, 54
Ultra Light の実行プラン, 472

ふ

ファイル
MLFileTransfer による転送, 82
Ultra Light ActiveSync プロバイダ, 125
Ultra Light CustDB サンプルのロケーション, 16
ファイルサイズ
Ultra Light データベースのトラブルシューティング, 487
ファイルシステム
(*参照* VFS)
ファイルの転送
MLFleTransfer による Ultra Light ファイル, 82
ファイル名
Ultra Light 接続パラメータ, 37
フェッチ
Ultra Light, 44
フォーマットオプション (Ultra Light)
utf8_encoding の使用法, 27
複数のデバイス
Ultra Light 接続パラメータ, 37
複数のデータベース
Ultra Light 最大数, 475
物理的制限
Ultra Light, 9
部分文字列
Ultra Light 説明, 403
Ultra Light での置換, 384
プライマリキー
Ultra Light J カラムの順序, 433
Ultra Light J 整合性制約, 433
Ultra Light UUID と GUID, 374
Ultra Light UUID を使用したユニークな値の生成, 374
Ultra Light インデックス, 33
Ultra Light カラムの順序, 433
Ultra Light 整合性制約, 433
Ultra Light データインポートのトラブルシューティング, 488
Ultra Light テーブル, 51
Ultra Light テーブル順序, 79
Ultra Light 特性, 57
Ultra Light ユニークな値の生成, 374
プライマリキーインデックス
Ultra Light 使用の省略, 471
プライマリキーの一意性の管理
Mobile Link システム内の Ultra Light クライアント, 70
ブラウズ
Ultra Light テーブル情報, 54
Ultra Light テーブルのブラウズ方法, 54
プラグイン
Ultra Light トラブルシューティング, 485
フラッシュ
Ultra Light データベース, 192
フラッシュカウント
Ultra Light フォーマット, 192
フラッシュタイムアウト
Ultra Light フォーマット, 193
プラットフォーム
Ultra Light ファイルの記憶領域, 37
Ultra Light 複数の接続パラメータ, 37
プラン
Ultra Light クエリプランの解釈, 473
Ultra Light クエリプランの操作, 469
Ultra Light クエリプランの変更, 469
Ultra Light 操作, 473
Ultra Light テキストプラン, 472
Ultra Light のカーソル, 352

Ultra Light の構文, 352
Ultra Light プランの解釈, 473
Ultra Light プランの操作, 473
プレフィクス
Windows Mobile データベース用 Ultra Light, 38
プレースホルダ
Ultra Light SQL 入力パラメータ, 280
プロシージャ
Ultra Light 制限事項, 3
プロセスの終了
Ultra Light アップグレードのトラブルシューティング, 485
プロバイダ
Ultra Light ActiveSync ファイル, 125
プロバイダファイル
Ultra Light ActiveSync の配備, 125
プロパティ
DB_PROPERTY 関数, 348
Ultra Light Java Edition システムテーブル, 267
Ultra Light Java Edition データベース, 243
Ultra Light データベース, 187
Ultra Light データベース作成パラメータ, 25
Ultra Light ブラウズ, 39
プロパティ (Ultra Light)
DB_PROPERTY 関数, 348
文
Ultra Light, 416
Ultra Light 準備文の入力パラメータ, 280
Ultra Light タイプ, 417
Ultra Light 文のパフォーマンスのチューニング, 480
分割
Ultra Light プライマリキー, 71
Ultra Light ローのパブリッシュ, 86
分割サイズ
Ultra Light デフォルトの不足分割サイズ, 71
Ultra Light デフォルト分割サイズの上書き, 73
Ultra Light デフォルト分割サイズの選択, 71
プール
Ultra Light 未使用のグローバル ID, 71

へ

平均関数
Ultra Light AVG 関数, 325
平方根関数
Ultra Light SQRT 関数, 393
変換
SQL Anywhere データベースから Ultra Light データベースへ, 33
Ultra Light あいまいな日付, 149
UltraLite CAST, 328
変換関数
Ultra Light アルファベット順リスト, 314
変換文字列
Ultra Light 説明, 319
変更
Ultra Light ALTER PUBLICATION 文, 419
Ultra Light ALTER TABLE 文, 422
Ultra Light ALTER USER 文, 425
Ultra Light カラム, 422
Ultra Light カラムの変更方法, 52
Ultra Light テーブル, 422
編集
Ultra Light テーブルの編集方法, 54
変数
Ultra Light SQL, 293
ベンチマークテスト
Ultra Light SQL のテスト, 480
Ultra Light 手法の概要, 481
Ultra Light 種類, 479
Ultra Light 信頼できるテストの特性, 482
Ultra Light テストの作成, 482
Ultra Light テストの実行および分析, 483
Ultra Light テストの準備, 481
Ultra Light データベース, 479
ページ
Ultra Light でのサイズの考慮事項, 151
ページサイズ
Ultra Light Java Edition の page_size プロパティ, 243
Ultra Light page_size プロパティ, 187

ほ

ホスト
Ultra Light ULSynchronize 引数, 106
ポート番号
Ultra Light ULSynchronize 引数, 106

ま

幻ロー
Ultra Light, 44
マルチスレッド
Ultra Light アプリケーション, 475
マルチテーブルジョイン

Ultra Light データベース, 3
丸め
Ultra Light scale, 154
Ultra Light scale 作成パラメータ, 154

め

メカニズム
Ultra Light アプリケーションおよびデータベースの配備, 113
メタデータ
Ultra Light 予約サイズの検討, 183
メディア障害
Ultra Light トランザクションの概要, 477
メモリ
Ultra Light 制限事項, 9
メモリ障害
Ultra Light 回避, 183
メモリの使用
Ultra Light SQL のパフォーマンス, 480
Ultra Light インデックス, 463
Ultra Light データベースの記憶領域, 37
Ultra Light ローのステータス, 476

も

文字関数
Ultra Light アルファベット順リスト, 319
文字セット
Ultra Light char_set プロパティ, 187
Ultra Light collation 作成パラメータ, 141
Ultra Light データベース, 27
Ultra Light 文字セット, 26
Ultra Light 文字列, 270
Windows Mobile 上の Ultra Light, 27
Windows 上の Ultra Light, 27
文字セット変換
パスワード, 439
文字列
Ultra Light Embedded SQL, 203
Ultra Light SQL, 270
Ultra Light 大文字と小文字の区別, 270
Ultra Light 最大サイズ, 9
Ultra Light での後続ブランクの削除, 387
Ultra Light での置換, 384
Ultra Light の SQL 関数, 319
Ultra Light 日付への nearest_century 変換, 149
文字列演算子
Ultra Light 動的 SQL 構文, 292
文字列関数
Ultra Light アルファベット順リスト, 319
文字列の位置
Ultra Light LOCATION 関数, 363
文字列の長さ
Ultra Light LENGTH 関数, 361
文字列リテラル
Ultra Light 定数, 276
モデリング
Mobile Link からの Ultra Light データベース, 23
役割名
Ultra Light J 外部キー, 433
Ultra Light J 役割名, 433
Ultra Light 外部キー, 433
Ultra Light 役割名, 433

ゆ

優先度
Ultra Light SQL 演算子の優先度, 293
ユニバーサルユニーク識別子
(*参照* UUID)
Ultra Light NEWID 関数の SQL 構文, 374
ユニークインデックス
Ultra Light UNIQUE SQL パラメータ, 428
Ultra Light インデックスの作成, 59
Ultra Light 特性, 57
ユニークキー
Ultra Light インデックスの作成, 59
Ultra Light 特性, 57
ユニークでないインデックス
Ultra Light インデックスの作成, 59
Ultra Light 特性, 57
ユーザ
Ultra Light ALTER USER 文, 425
Ultra Light DROP USER 文, 443
Ultra Light データベースでの管理, 61
Ultra Light データベース用の作成, 65
ユーザ ID
Ultra Light データベース, 61
Ultra Light の最大長, 186
ユーザ作成ウィザード
Ultra Light 使用法, 65
ユーザ定義データ型
Ultra Light サポート対象外, 293
Ultra Light で相当する型, 312
ユーザ認証

PWD Ultra Light 接続パラメータ, 182
Ultra Light, 61
ユーザの認証
Ultra Light Authentication Value 同期パラメータ, 94
Ultra Light getUserName メソッド, 110
Ultra Light Password 同期パラメータ, 99
Ultra Light user_name 同期, 110
Ultra Light 同期時のカスタム, 97
Ultra Light 同期ステータスレポート, 92
カスタム Mobile Link, 92
ユーティリティ
Ultra Light, 196
Ultra Light Interactive SQL (dbisql) 構文, 197
Ultra Light Java Edition データベースアンロード (uljunload) 構文, 254
Ultra Light Java Edition データベース情報 (uljinfo) 構文, 251
Ultra Light Java Edition データベース転送ユーティリティ (ULjDbTServ) 構文, 257
Ultra Light Java Edition データベースロード (uljload) 構文, 252
Ultra Light SQL プリプロセッサ (sqlpp) の構文, 203
Ultra Light エラーコード, 197
Ultra Light エンジン起動 (uleng16) の構文, 207
Ultra Light エンジン停止 (ulstop) の構文, 208
Ultra Light 情報 (ulinfo) の構文, 210
Ultra Light データベース検証 (ulvalid) の構文, 235
Ultra Light データベース消去 (ulerase) の構文, 209
Ultra Light データベース初期化 (ulinit) の構文, 211
Ultra Light データベースのアンロード (ulunload) の構文, 232
Ultra Light データベースへの XML のロード (ulload) の構文, 219
Ultra Light 同期 (ulsync) の構文, 224
Ultra Light トラブルシューティング, 488
デバイスでの Windows Mobile データベース管理, 38

よ

要求
Ultra Light 同時実行性, 475
Ultra Light による管理, 475
曜日
Ultra Light DOW 関数, 350
読み込み
Ultra Light テーブルロー, 44
読み込み専用テーブル
Ultra Light データベース, 80
Ultra Light 同期, 80
予約語
Ultra Light SQL, 269

ら

ライブラリ
Ultra Light FIPS 対応のアプリケーション, 147
Ultra Light uleng を Windows Mobile に配備, 18
Ultra Light での選択, 18
乱数
RAND 関数, 381
Ultra Light J RAND, 381
ランタイム
(*参照* Ultra Light ランタイム)
ランタイムライブラリ
Ultra Light, 18

り

リカバリ
Ultra Light, 46
Ultra Light 説明, 46
Ultra Light トランザクションの概要, 477
Ultra Light の概要, 476
リスト
Ultra Light LIST 関数の構文, 363
リストア
Ultra Light トランザクションの概要, 477
リターンコード
Ultra Light Interactive SQL ユーティリティ (dbisql), 202
リテラル
Ultra Light 定数, 276
リファレンスデータベース
Ultra Light オプション, 33
Ultra Light 作成, 33
説明, 33
リモート ID
Ultra Light データベースで設定, 195
リモートサーバ
Ultra Light CREATE TABLE 文, 432
リモートデータベース

Ultra Light クライアントの作成, 70
Ultra Light 同期のカウント, 3
リモートデータベースの作成
Ultra Light クライアント, 70
リモートデータベースの中央管理
Ultra Light データベース, 23
リンク
Ultra Light ランタイムライブラリ, 18

れ

例
(*参照* サンプル)
(*参照* チュートリアル)
連結文字列
Ultra Light 文字列演算子, 292
連邦情報処理規格 (*参照* FIPS)

ろ

ログ
Ultra Light 内部メカニズム, 46
ロック
Ultra Light 同時実行性, 477
論理
Ultra Light 同期の設計のための取り込み, 74
論理演算子
Ultra Light SQL 構文, 283
ロー
Ultra Light INSERT 文, 445
Ultra Light J INSERT 文, 445
Ultra Light J テーブルからすべてのローを削除, 457
Ultra Light でのフェッチ, 44
Ultra Light でのロック, 477
Ultra Light テーブルからすべてのローを削除, 457
Ultra Light パブリッシュ, 86
Ultra Light バルク挿入, 446
ロード
Ultra Light LOAD TABLE 文, 446
Ultra Light バルク挿入, 446
ローのパック
Ultra Light 影響, 151
Ultra Light 確認, 183
Ultra Light 説明, 50
ローのフェッチ
Ultra Light 同時実行性, 44
ロールバック
Ultra Light J トランザクション, 450
Ultra Light データベース, 476
Ultra Light トランザクション, 450
Ultra Light トランザクションの概要, 477

わ

ワイルドカード
Ultra Light LIKE 探索条件, 287
Ultra Light PATINDEX 関数, 377