SQL Anywhere - Ultra Light
文書バージョン: 17 – 2016-05-11

Ultra Light - C++ API リファレンス

目次

1 Ultra Light C++ API リファレンス

Ultra Light C++ は豊富な API オブジェクトを提供します。

次に、よく使用される API オブジェクトの一部を示します。

ULDatabaseManager

データベースと接続を管理する方法を提供します。

ULConnection

Ultra Light データベースへの接続を表します。ULConnection オブジェクトは 1 つまたは複数作成できます。

ULTable

データベースのテーブルへの直接アクセスを提供します。

ULPreparedStatement、ULResultSet、ULResultSetSchema

動的 SQL 文の作成、クエリの記述、INSERT、UPDATE、DELETE 文の実行、プログラムによるデータベースの結果セットの制御を行います。

ヘッダファイル

- `ulcpp.h`

> **注記**
>
> 主な **SQL Anywhere** マニュアルをお探しですか。マニュアルをローカルにインストールした場合は、Windows のスタートメニューを使用してアクセスするか (Microsoft Windows)、`C:¥Program Files¥SQL Anywhere 17¥Documentation` にナビゲートします。
>
> また、DocCommentXchange の Web で、主な SQL Anywhere API リファレンスマニュアルにアクセスすることもできます。http://dcx.sap.com

このセクションの内容:

Ultra Light テーブルのインデックスのスキーマを表します。

ULPreparedStatement クラス [78 ページ]
準備された SQL 文を表します。

ULResultSet クラス [93 ページ]
Ultra Light データベースの結果セットを表します。

ULResultSetSchema クラス [152 ページ]
Ultra Light の結果セットのスキーマを表します。

ULTable クラス [159 ページ]
Ultra Light データベース内のテーブルを表します。

ULTableSchema クラス [172 ページ]
Ultra Light のテーブルのスキーマを表します。

ul_column_default_type 列挙 [182 ページ]
カラムのデフォルト型を識別します。

ul_column_name_type 列挙 [183 ページ]
結果セットの記述時にカラムの名前を取得する方法を制御する値を指定します。

ul_index_flag 列挙 [184 ページ]
インデックスのプロパティを識別するフラグ (ビットフィールド)。

ul_table_sync_type 列挙 [185 ページ]
テーブルの同期タイプを識別します。

UL_BLOB_CONTINUE 変数 [186 ページ]
ULResultSet.GetStringChunk メソッドまたは ULResultSet.GetByteChunk メソッドを使用してデータを読み込む場合に使用します。

ul_index_iter_start 変数 [187 ページ]
テーブル内のインデックス反復を初期化する場合に GetNextIndex メソッドで使用されます。

ul_publication_iter_start 変数 [187 ページ]
データベースでのパブリケーション反復を初期化する場合に GetNextPublication メソッドで使用されます。

ul_table_iter_start 変数 [187 ページ]
データベースでのテーブル反復を初期化する場合に GetNextTable メソッドで使用されます。

1.1 ULConnection クラス

Ultra Light データベースへの接続を表します。

構文

```
public class ULConnection
```

メンバー

ULConnection のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public virtual ul_u_long	CancelGetNotification(const char *) [15 ページ]	指定された名前に一致する、すべてのキューに登録されている保留中の取得通知コールをキャンセルします。
public virtual bool	ChangeEncryptionKey(const char *) [15 ページ]	Ultra Light データベースのデータベース暗号化キーを変更します。
public virtual bool	Checkpoint() [16 ページ]	チェックポイント操作を実行し、保留中になっているコミット済みトランザクションをデータベースにフラッシュします。
public virtual void	Close(ULError *) [16 ページ]	この接続と、残りの関連オブジェクトを破棄します。
public virtual bool	Commit() [17 ページ]	現在のトランザクションをコミットします。
public virtual ul_u_long	CountUploadRows(const char *, ul_u_long) [17 ページ]	同期するためにアップロードする必要があるローの数を数えます。
public virtual bool	CreateNotificationQueue(const char *, const char *) [18 ページ]	この接続のイベント通知キューを作成します。
public virtual bool	DeclareEvent(const char *) [19 ページ]	登録およびトリガされるイベントを宣言します。
public virtual bool	DestroyNotificationQueue(const char *) [20 ページ]	指定されたイベント通知キューを破棄します。
public virtual bool	ExecuteScalar(void *, size_t, ul_column_storage_type, const char *, ...) [20 ページ]	SQL SELECT 文を直接実行し、単一の結果を返します。
public virtual bool	ExecuteScalarV(void *, size_t, ul_column_storage_type, const char *, va_list) [22 ページ]	代入値のリストを指定して、SQL SELECT 文の文字列を実行します。
public virtual bool	ExecuteStatement(const char *) [24 ページ]	SQL 文の文字列を直接実行します。
public virtual ul_u_long	GetChildObjectCount() [25 ページ]	接続で現在開いている子オブジェクトの数を取得します。
public virtual const char *	GetDatabaseProperty(const char *) [25 ページ]	データベースプロパティの値を取得します。
public virtual ul_u_long	GetDatabasePropertyInt(const char *) [26 ページ]	データベースプロパティの整数値を取得します。
public virtual ULDatabaseSchema *	GetDatabaseSchema() [27 ページ]	データベースのスキーマを問い合わせるために使用されるオブジェクトポインタを返します。
public virtual bool	GetLastDownloadTime(const char *, DECL_DATETIME *) [27 ページ]	指定したパブリケーションが最後にダウンロードされた時刻を取得します。

変更子とタイプ	メソッド	説明
public virtual const ULError *	GetLastError() [28 ページ]	最後の呼び出しに関連付けられているエラー情報を返します。
public virtual ul_u_big	GetLastIdentity() [28 ページ]	@@identity の値を取得します。
public virtual const char *	GetNotification(const char *, ul_u_long) [29 ページ]	イベント通知を読み込みます。
public virtual const char *	GetNotificationParameter(const char *, const char *) [30 ページ]	GetNotification メソッドによって読み込まれたイベント通知のパラメータを取得します。
public virtual SQLCA *	GetSqlca() [31 ページ]	この接続に関連付けられている SQLCA を取得します。
public virtual bool	GetSyncResult(ul_sync_result *) [31 ページ]	前回の同期結果を取得します。
public virtual void *	GetUserPointer() [32 ページ]	SetUserPointer メソッドによって最後に設定されたポインタ値を取得します。
public virtual ul_u_short	GlobalAutoincUsage() [32 ページ]	グローバルオートインクリメントのデフォルト値を持つすべてのカラムで、デフォルト値が使用されている比率 (%) を取得します。
public virtual bool	GrantConnectTo(const char *, const char *) [33 ページ]	指定されたパスワードを持つ新しいまたは既存のユーザ ID に、Ultra Light データベースへのアクセスを許可します。
public virtual void	InitSyncInfo(ul_sync_info *) [33 ページ]	同期情報の構造体を初期化します。
public virtual ULTable *	OpenTable(const char *, const char *) [34 ページ]	テーブルを開きます。
public virtual ULPreparedStatement *	PrepareStatement(const char *) [35 ページ]	SQL 文の準備を行います。
public virtual bool	RegisterForEvent(const char *, const char *, const char *, bool) [35 ページ]	イベントの通知を受け取るためのキューを登録または登録解除します。
public virtual bool	ResetLastDownloadTime(const char *) [36 ページ]	アプリケーションが以前にダウンロードされたデータを再同期するように、パブリケーションの最終ダウンロード時間をリセットします。
public virtual bool	RevokeConnectFrom(const char *) [37 ページ]	Ultra Light データベースからユーザ ID のアクセス権を取り消します。
public virtual bool	Rollback() [37 ページ]	現在のトランザクションをロールバックします。
public virtual bool	RollbackPartialDownload() [38 ページ]	失敗した同期からの変更をロールバックします。
public virtual ul_u_long	SendNotification(const char *, const char *, const char *) [38 ページ]	指定された名前と一致するすべてのキューに通知を送信します。
public virtual bool	SetDatabaseOption(const char *, const char *) [39 ページ]	指定されたデータベースオプションを設定します。
public virtual bool	SetDatabaseOptionInt(const char *, ul_u_long) [40 ページ]	データベースオプションを設定します。

変更子とタイプ	メソッド	説明
public virtual void	SetSynchronizationCallback(ul_sync_observer_fn, void *) [41 ページ]	同期の実行中に呼び出されるようコールバックを設定します。
public virtual bool	SetSyncInfo(char const *, ul_sync_info *) [41 ページ]	指定された ul_sync_info 構造体に基づいて、指定された名前を使用して同期プロファイルを作成します。
public virtual void *	SetUserPointer(void *) [42 ページ]	呼び出すアプリケーションによって接続に使用される任意のポインタ値を設定します。
public virtual bool	StartSynchronizationDelete() [42 ページ]	この接続の START SYNCHRONIZATION DELETE を設定します。
public virtual bool	StopSynchronizationDelete() [43 ページ]	この接続の STOP SYNCHRONIZATION DELETE を設定します。
public virtual bool	Synchronize(ul_sync_info *) [43 ページ]	Ultra Light アプリケーションで同期を開始します。
public virtual bool	SynchronizeFromProfile(const char *, const char *, ul_sync_observer_fn, void *) [44 ページ]	指定されたプロファイルとマージパラメータを使用して、データベースを同期します。
public virtual ul_u_long	TriggerEvent(const char *, const char *) [45 ページ]	ユーザ定義のイベントをトリガして、登録されたすべてのキューに通知を送信します。
public virtual bool	ValidateDatabase(ul_u_short, ul_validate_callback_fn, void *, const char *) [46 ページ]	この接続でのデータベースを検証します。

このセクションの内容:

CancelGetNotification(const char *) メソッド [15 ページ]
指定された名前に一致する、すべてのキューに登録されている保留中の取得通知コールをキャンセルします。

ChangeEncryptionKey(const char *) メソッド [15 ページ]
Ultra Light データベースのデータベース暗号化キーを変更します。

Checkpoint() メソッド [16 ページ]
チェックポイント操作を実行し、保留中になっているコミット済みトランザクションをデータベースにフラッシュします。

Close(ULError *) メソッド [16 ページ]
この接続と、残りの関連オブジェクトを破棄します。

Commit() メソッド [17 ページ]
現在のトランザクションをコミットします。

CountUploadRows(const char *, ul_u_long) メソッド [17 ページ]
同期するためにアップロードする必要があるローの数を数えます。

CreateNotificationQueue(const char *, const char *) メソッド [18 ページ]
この接続のイベント通知キューを作成します。

DeclareEvent(const char *) メソッド [19 ページ]
登録およびトリガされるイベントを宣言します。

DestroyNotificationQueue(const char *) メソッド [20 ページ]

指定されたイベント通知キューを破棄します。

ExecuteScalar(void *, size_t, ul_column_storage_type, const char *, ...) メソッド [20 ページ]
SQL SELECT 文を直接実行し、単一の結果を返します。

ExecuteScalarV(void *, size_t, ul_column_storage_type, const char *, va_list) メソッド [22 ページ]
代入値のリストを指定して、SQL SELECT 文の文字列を実行します。

ExecuteStatement(const char *) メソッド [24 ページ]
SQL 文を直接実行します。

GetChildObjectCount() メソッド [25 ページ]
接続で現在開いている子オブジェクトの数を取得します。

GetDatabaseProperty(const char *) メソッド [25 ページ]
データベースプロパティの値を取得します。

GetDatabasePropertyInt(const char *) メソッド [26 ページ]
データベースプロパティの整数値を取得します。

GetDatabaseSchema() メソッド [27 ページ]
データベースのスキーマを問い合わせるために使用されるオブジェクトポインタを返します。

GetLastDownloadTime(const char *, DECL_DATETIME *) メソッド [27 ページ]
指定したパブリケーションが最後にダウンロードされた時刻を取得します。

GetLastError() メソッド [28 ページ]
最後の呼び出しに関連付けられているエラー情報を返します。

GetLastIdentity() メソッド [28 ページ]
@@identity の値を取得します。

GetNotification(const char *, ul_u_long) メソッド [29 ページ]
イベント通知を読み込みます。

GetNotificationParameter(const char *, const char *) メソッド [30 ページ]
GetNotification メソッドによって読み込まれたイベント通知のパラメータを取得します。

GetSqlca() メソッド [31 ページ]
この接続に関連付けられている SQLCA を取得します。

GetSyncResult(ul_sync_result *) メソッド [31 ページ]
前回の同期結果を取得します。

GetUserPointer() メソッド [32 ページ]
SetUserPointer メソッドによって最後に設定されたポインタ値を取得します。

GlobalAutoincUsage() メソッド [32 ページ]
グローバルオートインクリメントのデフォルト値を持つすべてのカラムで、デフォルト値が使用されている比率 (%) を取得します。

GrantConnectTo(const char *, const char *) メソッド [33 ページ]
指定されたパスワードを持つ新しいまたは既存のユーザ ID に、Ultra Light データベースへのアクセスを許可します。

InitSyncInfo(ul_sync_info *) メソッド [33 ページ]
同期情報の構造体を初期化します。

OpenTable(const char *, const char *) メソッド [34 ページ]

テーブルを開きます。

PrepareStatement(const char *) メソッド [35 ページ]
SQL 文の準備を行います。

RegisterForEvent(const char *, const char *, const char *, bool) メソッド [35 ページ]
イベントの通知を受け取るためのキューを登録または登録解除します。

ResetLastDownloadTime(const char *) メソッド [36 ページ]
アプリケーションが以前にダウンロードされたデータを再同期するように、パブリケーションの最終ダウンロード時間をリセットします。

RevokeConnectFrom(const char *) メソッド [37 ページ]
Ultra Light データベースからユーザ ID のアクセス権を取り消します。

Rollback() メソッド [37 ページ]
現在のトランザクションをロールバックします。

RollbackPartialDownload() メソッド [38 ページ]
失敗した同期からの変更をロールバックします。

SendNotification(const char *, const char *, const char *) メソッド [38 ページ]
指定された名前と一致するすべてのキューに通知を送信します。

SetDatabaseOption(const char *, const char *) メソッド [39 ページ]
指定されたデータベースオプションを設定します。

SetDatabaseOptionInt(const char *, ul_u_long) メソッド [40 ページ]
データベースオプションを設定します。

SetSynchronizationCallback(ul_sync_observer_fn, void *) メソッド [41 ページ]
同期の実行中に呼び出されるようコールバックを設定します。

SetSyncInfo(char const *, ul_sync_info *) メソッド [41 ページ]
指定された ul_sync_info 構造体に基づいて、指定された名前を使用して同期プロファイルを作成します。

SetUserPointer(void *) メソッド [42 ページ]
呼び出すアプリケーションによって接続に使用される任意のポインタ値を設定します。

StartSynchronizationDelete() メソッド [42 ページ]
この接続の START SYNCHRONIZATION DELETE を設定します。

StopSynchronizationDelete() メソッド [43 ページ]
この接続の STOP SYNCHRONIZATION DELETE を設定します。

Synchronize(ul_sync_info *) メソッド [43 ページ]
Ultra Light アプリケーションで同期を開始します。

SynchronizeFromProfile(const char *, const char *, ul_sync_observer_fn, void *) メソッド [44 ページ]
指定されたプロファイルとマージパラメータを使用して、データベースを同期します。

TriggerEvent(const char *, const char *) メソッド [45 ページ]
ユーザ定義のイベントをトリガして、登録されたすべてのキューに通知を送信します。

ValidateDatabase(ul_u_short, ul_validate_callback_fn, void *, const char *) メソッド [46 ページ]
この接続でのデータベースを検証します。

1.1.1 CancelGetNotification(const char *) メソッド

指定された名前に一致する、すべてのキューに登録されている保留中の取得通知コールをキャンセルします。

構文

```
public virtual ul_u_long CancelGetNotification (const char * queueName)
```

パラメータ

queueName キューの名前。

戻り値

影響を受けるキューの数。(必ずしも、ブロックされた読み込みの数ではありません)

1.1.2 ChangeEncryptionKey(const char *) メソッド

Ultra Light データベースのデータベース暗号化キーを変更します。

構文

```
public virtual bool ChangeEncryptionKey (const char * newKey)
```

パラメータ

newKey データベースの新しい暗号化キー。

戻り値

成功した場合は true、失敗した場合は false。

備考

このメソッドを呼び出すアプリケーションでは、データベースが同期されていること、または信頼できるバックアップコピーが作成されていることを、先に確認しておく必要があります。ChangeEncryptionKey メソッドは、完了まで実行する必要のある操作であるため、信頼できるバックアップがあることが重要です。データベース暗号化キーを変更すると、まずデータベースのすべてのローは古いキーを使用して復号され、次に新しいキーを使用して再度暗号化されて、書き込まれます。この操作は元に戻せません。暗号化変更処理が完了しなかった場合、データベースは無効な状態のままになり、再度アクセスすることはできなくなります。

1.1.3 Checkpoint() メソッド

チェックポイント操作を実行し、保留中になっているコミット済みトランザクションをデータベースにフラッシュします。

構文

```
public virtual bool Checkpoint ()
```

戻り値

成功した場合は true、失敗した場合は false。

備考

Checkpoint メソッドを呼び出しても、現在のトランザクションすべてがコミットされるわけではありません。このメソッドは、パフォーマンスを向上させるために後回しにされた自動トランザクションチェックポイントとともに (*commit_flush* 接続パラメータを使用して) 使用されます。

Checkpoint メソッドを使用すると、保留中のコミット済みトランザクションがすべてデータベースの記憶領域に書き込まれることが保証されます。

1.1.4 Close(ULError *) メソッド

この接続と、残りの関連オブジェクトを破棄します。

構文

```
public virtual void Close (ULError * error)
```

パラメータ

error エラー情報を受信するためのオプションの ULError オブジェクト。

1.1.5 Commit() メソッド

現在のトランザクションをコミットします。

構文

```
public virtual bool Commit ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.1.6 CountUploadRows(const char *, ul_u_long) メソッド

同期するためにアップロードする必要があるローの数を数えます。

構文

```
public virtual ul_u_long CountUploadRows (
    const char * pubList,
    ul_u_long threshold
)
```

パラメータ

pubList チェック対象となるパブリケーションのカンマ区切りのリストを含む文字列。空の文字列 (UL_SYNC_ALL マクロ) は、"非同期" とマーク付けされたものを除くすべてのテーブルを表します。アスタリスクのみの文字列 (UL_SYNC_ALL_PUBS マクロ) は、いずれかのパブリケーションで参照されているすべてのテーブルを表します。一部のテーブルは、どのパブリケーションの一部でもないため、この値が "*" の場合は含まれません。

threshold カウントするローの最大数を判断します。呼び出しの所要時間を制限します。threshold が 0 の場合、制限はありません (つまり、同期する必要のあるすべてのローをカウントします)。また、threshold が 1 の場合、同期の必要なローがあるかどうかを簡単に判別するために使用できます。

戻り値

指定されたパブリケーションのセットまたはデータベース全体のいずれかで、同期を必要とするローの数。

備考

このメソッドを使用すると、ユーザは同期、または自動バックグラウンド同期が行われるタイミングの決定を求められます。

次の呼び出しでは、データベース全体をチェックして、同期させるローの総数を確認します。

```
count = conn->CountUploadRows( UL_SYNC_ALL, 0 );
```

次の呼び出しでは、最大 1000 のローに対してパブリケーション PUB1 と PUB2 がチェックされます。

```
count = conn->CountUploadRows( "PUB1,PUB2", 1000 );
```

次の呼び出しでは、パブリケーション PUB1 と PUB2 で同期させる必要のあるローがあるかどうかがチェックされます。

```
anyToSync = conn->CountUploadRows( "PUB1,PUB2", 1 ) != 0;
```

1.1.7 CreateNotificationQueue(const char *, const char *) メソッド

この接続のイベント通知キューを作成します。

構文

```
public virtual bool CreateNotificationQueue (
    const char * name,
    const char * parameters
)
```

パラメータ

name 新しいキューの名前。
parameters 予約済み。NULL に設定されます。

戻り値

成功した場合は true、失敗した場合は false。

備考

キュー名は、接続ごとにスコープされるため、別々の接続で同じ名前を持つキューを作成できます。イベント通知が送信されると、データベース内で一致する名前を持つすべてのキューが、個別のインスタンスの通知を受け取ります。名前では、大文字と小文字が区別されません。RegisterForEvent メソッドを呼び出したときに、キューが指定されていない場合は、接続ごとにデフォルトのキューが作成されます。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。

関連情報

RegisterForEvent(const char *, const char *, const char *, bool) メソッド [35 ページ]

1.1.8 DeclareEvent(const char *) メソッド

登録およびトリガされるイベントを宣言します。

構文

```
public virtual bool DeclareEvent (const char * eventName)
```

パラメータ

eventName 新しいユーザ定義イベントの名前。

戻り値

イベントが正常に宣言された場合は true。正常に宣言されず、名前がすでに使用されているか無効な場合は false。

備考

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。このメソッドは、ユーザ定義イベントを宣言します。ユーザ定義イベントは、TriggerEvent メソッドでトリガされます。イベント名は、ユニークにする必要があります。名前では、大文字と小文字が区別されません。

関連情報

TriggerEvent(const char *, const char *) メソッド [45 ページ]

1.1.9 DestroyNotificationQueue(const char *) メソッド

指定されたイベント通知キューを破棄します。

構文

```
public virtual bool DestroyNotificationQueue (const char * name)
```

パラメータ

name 破棄するキューの名前。

戻り値

成功した場合は true、失敗した場合は false。

備考

キュー内に未読の通知が残っている場合は、警告が通知されます。未読の通知は破棄されます。接続のデフォルトのイベントキューが作成されている場合、接続が閉じると破棄されます。

1.1.10 ExecuteScalar(void *, size_t, ul_column_storage_type, const char *, ...) メソッド

SQL SELECT 文を直接実行し、単一の結果を返します。

構文

```
public virtual bool ExecuteScalar (
    void * dstPtr,
    size_t dstSize,
    ul_column_storage_type dstType,
```

```
    const char * sql,
    ...
)
```

パラメータ

dstPtr 値を受信するための必要な型の変数へのポインタ。

dstSize 値を受信するための変数のサイズ (適用できる場合)。

dstType 取得する値の型。この値は、変数の型と一致する必要があります。

sql SELECT 文。オプションで '?' パラメータが含まれます。各 '?' 代入パラメータに、対応する文字列 (char *) パラメータを指定します。

戻り値

クエリが正常に実行され、値が正常に取得される場合は true。それ以外の場合で値がフェッチされないときは false。SQLCODE エラーコードをチェックして、false が返される理由を特定します。警告またはエラー (SQLE_NOERROR) が示されない場合、選択した値は NULL になります。

備考

dstPtr 値は、dstType 値に一致する正しい型の変数を指す必要があります。dstSize パラメータは、文字列やバイナリなどの可変サイズの値にのみ必要とされ、それ以外の場合は無視されます。パラメータ値の変数リストは、文のパラメータに対応している必要があり、すべての値は文字列であると想定されます。(内部的には、Ultra Light は、文で必要とされるときにパラメータ値をキャストします)

次の型がサポートされます。

UL_TYPE_BIT/UL_TYPE_TINY

変数の型 ul_byte (8 ビット、符号なし) を使用します。

UL_TYPE_U_SHORT/UL_TYPE_S_SHORT

変数の型 ul_u_short/ul_s_short (16 ビット) を使用します。

UL_TYPE_U_LONG/UL_TYPE_S_LONG

変数の型 ul_u_long/ul_s_long (32 ビット) を使用します。

UL_TYPE_U_BIG/UL_TYPE_S_BIG

変数の型 ul_u_big/ul_s_big (64 ビット) を使用します。

UL_TYPE_DOUBLE

変数の型 ul_double (double) を使用します。

UL_TYPE_REAL

変数の型 ul_real (float) を使用します。

UL_TYPE_BINARY

変数の型 ul_binary を使用し、*dstSize* を指定します (GetBinary() の場合と同様)。

UL_TYPE_TIMESTAMP_STRUCT

変数の型 DECL_DATETIME を使用します。

UL_TYPE_CHAR

変数の型 char [] (文字バッファ) を使用し、*dstSize* にバッファのサイズを設定します (GetString() の場合と同様)。

UL_TYPE_WCHAR

変数の型 ul_wchar [] (ワイド文字バッファ) を使用し、*dstSize* にバッファのサイズを設定します (GetString() の場合と同様)。

UL_TYPE_TCHAR

どちらのバージョンのメソッドが呼び出されているかに応じて、UL_TYPE_CHAR または UL_TYPE_WCHAR と同様。

次の例は、整数のフェッチを示しています。

```
ul_u_long       val;
ok = conn->ExecuteScalar( &val, 0, UL_TYPE_U_LONG,
   "SELECT count(*) FROM t WHERE col LIKE ?", "ABC%" );
```

次の例は、文字列のフェッチを示しています。

```
char    val[40];
ok = conn->ExecuteScalar( &val, sizeof(val), UL_TYPE_CHAR,
   "SELECT uuidtostr( newid() )" );
```

1.1.11 ExecuteScalarV(void *, size_t, ul_column_storage_type, const char *, va_list) メソッド

代入値のリストを指定して、SQL SELECT 文の文字列を実行します。

構文

```
public virtual bool ExecuteScalarV (
    void * dstPtr,
    size_t dstSize,
    ul_column_storage_type dstType,
    const char * sql,
    va_list args
)
```

パラメータ

dstPtr 値を受信するための必要な型の変数へのポインタ。

dstSize 値を受信するための変数のサイズ (適用できる場合)。

dstType 取得する値の型。この値は、変数の型と一致する必要があります。

sql SELECT 文。オプションで '?' パラメータが含まれます。
args 代入する文字列 (char *) 値のリスト。

戻り値

クエリが正常に実行され、値が正常に取得される場合は true。それ以外の場合で値がフェッチされないときは false。SQLCODE エラーコードをチェックして、false が返される理由を特定します。警告またはエラー (SQLE_NOERROR) が示されない場合、選択した値は NULL になります。

備考

dstPtr 値は、dstType 値に一致する正しい型の変数を指す必要があります。dstSize パラメータは、文字列やバイナリなどの可変サイズの値にのみ必要とされ、それ以外の場合は無視されます。パラメータ値の変数リストは、文のパラメータに対応している必要があり、すべての値は文字列であると想定されます。(内部的には、Ultra Light は、文で必要とされるときにパラメータ値をキャストします)

次の型がサポートされます。

UL_TYPE_BIT/UL_TYPE_TINY

変数の型 ul_byte (8 ビット、符号なし) を使用します。
UL_TYPE_U_SHORT/UL_TYPE_S_SHORT

変数の型 ul_u_short/ul_s_short (16 ビット) を使用します。
UL_TYPE_U_LONG/UL_TYPE_S_LONG

変数の型 ul_u_long/ul_s_long (32 ビット) を使用します。
UL_TYPE_U_BIG/UL_TYPE_S_BIG

変数の型 ul_u_big/ul_s_big (64 ビット) を使用します。
UL_TYPE_DOUBLE

変数の型 ul_double (double) を使用します。
UL_TYPE_REAL

変数の型 ul_real (float) を使用します。
UL_TYPE_BINARY

変数の型 ul_binary を使用し、*dstSize* を指定します (GetBinary() の場合と同様)。
UL_TYPE_TIMESTAMP_STRUCT

変数の型 DECL_DATETIME を使用します。
UL_TYPE_CHAR

変数の型 char [] (文字バッファ) を使用し、*dstSize* にバッファのサイズを設定します (GetString() の場合と同様)。
UL_TYPE_WCHAR

変数の型 ul_wchar [] (ワイド文字バッファ) を使用し、*dstSize* にバッファのサイズを設定します (GetString() の場合と同様)。

UL_TYPE_TCHAR

どちらのバージョンのメソッドが呼び出されているかに応じて、UL_TYPE_CHAR または UL_TYPE_WCHAR と同様。

1.1.12 ExecuteStatement(const char *) メソッド

SQL 文を直接実行します。

構文

```
public virtual bool ExecuteStatement (const char * sql)
```

パラメータ

sql 実行する SQL 文

戻り値

成功した場合は true、失敗した場合は false。

備考

このメソッドを使用して、non-SELECT 文を直接実行します。

PrepareStatement メソッドを使用して、変数パラメータと共に文を繰り返し実行するか、複数の結果をフェッチします。

関連情報

PrepareStatement(const char *) メソッド [35 ページ]

1.1.13 GetChildObjectCount() メソッド

接続で現在開いている子オブジェクトの数を取得します。

構文

```
public virtual ul_u_long GetChildObjectCount ()
```

戻り値

現在開いている子オブジェクトの数。

備考

このメソッドを使用してオブジェクトのリークを検出できます。

1.1.14 GetDatabaseProperty(const char *) メソッド

データベースプロパティの値を取得します。

構文

```
public virtual const char * GetDatabaseProperty (const char * propName)
```

パラメータ

propName 要求されているプロパティの名前。

戻り値

正常に実行された場合は、データベースプロパティの値が格納された文字列バッファへのポインタ。失敗した場合は NULL。

備考

戻り値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保存しておきたい場合はその値のコピーを作成してください。

例

次の例では、CharSet データベースプロパティの値を取得する方法を示します。

```
const char * charset = GetDatabaseProperty( "CharSet" );
```

1.1.15 GetDatabasePropertyInt(const char *) メソッド

データベースプロパティの整数値を取得します。

構文

```
public virtual ul_u_long GetDatabasePropertyInt (const char * propName)
```

パラメータ

propName 要求されているプロパティの名前。

戻り値

成功した場合は、プロパティの整数値。それ以外の場合は 0。

例

次の例では、ConnCount データベースプロパティの値を取得する方法を示します。

```
unsigned connectionCount = GetDatabasePropertyInt( "ConnCount" );
```

1.1.16 GetDatabaseSchema() メソッド

データベースのスキーマを問い合わせるために使用されるオブジェクトポインタを返します。

構文

```
public virtual ULDatabaseSchema * GetDatabaseSchema ()
```

戻り値

データベースのスキーマを問い合わせるために使用される ULDatabaseSchema オブジェクト。

1.1.17 GetLastDownloadTime(const char *, DECL_DATETIME *) メソッド

指定したパブリケーションが最後にダウンロードされた時刻を取得します。

構文

```
public virtual bool GetLastDownloadTime (
    const char * publication,
    DECL_DATETIME * value
)
```

パラメータ

publication パブリケーション名。

value 投入する DECL_DATETIME 構造体へのポインタ。値 January 1, 1900 は、パブリケーションがまだ同期されていないか、時間がリセットされたことを示します。

戻り値

指定されたパブリケーションの最終ダウンロード時間までに value が正常に投入された場合は true。それ以外の場合は false。

備考

次の呼び出しでは、'pub1' パブリケーションがダウンロードされた日付と時刻とともに dt 構造体が投入されます。

```
DECL_DATETIME dt;
ok = conn->GetLastDownloadTime( "pub1", &dt );
```

1.1.18 GetLastError() メソッド

最後の呼び出しに関連付けられているエラー情報を返します。

構文

```
public virtual const ULError * GetLastError ()
```

戻り値

最後の呼び出しに関連付けられている情報を含む ULError オブジェクトへのポインタ。

備考

アドレスが返されるエラーオブジェクトは、接続が開いている間は有効ですが、以降の呼び出しで自動的に更新されるわけではありません。GetLastError を呼び出して、更新されたステータス情報を取得する必要があります。

関連情報

ULError クラス [65 ページ]

1.1.19 GetLastIdentity() メソッド

@@identity の値を取得します。

構文

```
public virtual ul_u_big GetLastIdentity ()
```

戻り値

オートインクリメントカラムまたはグローバルオートインクリメントカラムに最後に挿入された値。

備考

この値は、データベースのオートインクリメントカラムまたはグローバルオートインクリメントカラムに最後に挿入された値です。データベースが停止している場合、この値は記録されません。このため、オートインクリメントの値が挿入される前にこのメソッドを呼び出すと、0 が返されます。

> **注記**
>
> 最後に挿入された値が別の接続の値である可能性があります。

1.1.20 GetNotification(const char *, ul_u_long) メソッド

イベント通知を読み込みます。

構文

```
public virtual const char * GetNotification (
    const char * queueName,
    ul_u_long waitms
)
```

パラメータ

queueName 読み取るキュー。または、デフォルト接続キューの場合は NULL。

waitms 返す前に、待機 (ブロック) する時間 (ミリ秒単位)。

戻り値

読み込まれたイベントの名前。エラーが発生した場合は NULL。

備考

この呼び出しは、通知が受信されるまで、または指定された待機時間が経過するまでブロックします。無期限に待機するには、waitms パラメータを UL_READ_WAIT_INFINITE に設定します。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 メソッドを使用します。通知を読み込んだ後に GetNotificationParameter メソッドを使用して、追加のパラメータを名前で取得します。

関連情報

CancelGetNotification(const char *) メソッド [15 ページ]
GetNotificationParameter(const char *, const char *) メソッド [30 ページ]

1.1.21 GetNotificationParameter(const char *, const char *) メソッド

GetNotification メソッドによって読み込まれたイベント通知のパラメータを取得します。

構文

```
public virtual const char * GetNotificationParameter (
    const char * queueName,
    const char * parameterName
)
```

パラメータ

queueName 読み取るキュー。デフォルト接続キューの場合は NULL。
parameterName 読み込むパラメータの名前 (または "*")。

戻り値

パラメータ値。エラーが発生した場合は NULL。

備考

指定されたキューで最近読み込まれた通知のパラメータのみが使用可能です。パラメータは名前によって取得されます。パラメータ名を "*" と指定すると、パラメータ文字列全体が取得されます。

関連情報

GetNotification(const char *, ul_u_long) メソッド [29 ページ]

1.1.22 GetSqlca() メソッド

この接続に関連付けられている SQLCA を取得します。

構文

```
public virtual SQLCA * GetSqlca ()
```

戻り値

この接続の SQLCA オブジェクトへのポインタ。

1.1.23 GetSyncResult(ul_sync_result *) メソッド

前回の同期結果を取得します。

構文

```
public virtual bool GetSyncResult (ul_sync_result * syncResult)
```

パラメータ

syncResult 投入する ul_sync_result 構造体へのポインタ。

戻り値

成功した場合は true、失敗した場合は false。

1.1.24 GetUserPointer() メソッド

SetUserPointer メソッドによって最後に設定されたポインタ値を取得します。

構文

```
public virtual void * GetUserPointer ()
```

関連情報

SetUserPointer(void *) メソッド [42 ページ]

1.1.25 GlobalAutoincUsage() メソッド

グローバルオートインクリメントのデフォルト値を持つすべてのカラムで、デフォルト値が使用されている比率 (%) を取得します。

構文

```
public virtual ul_u_short GlobalAutoincUsage ()
```

戻り値

グローバルオートインクリメントの値のカウンタによる使用済み比率 (%)。

備考

このデフォルト値を使用するカラムがデータベース内に複数含まれている場合は、すべてのカラムに対してこの値が計算され、最大値が返されます。たとえば、戻り値 99 は、少なくとも 1 つのカラムではデフォルト値が残されているが、きわめて少ないことを示します。

1.1.26 GrantConnectTo(const char *, const char *) メソッド

指定されたパスワードを持つ新しいまたは既存のユーザ ID に、Ultra Light データベースへのアクセスを許可します。

構文

```
public virtual bool GrantConnectTo (
    const char * uid,
    const char * pwd
)
```

パラメータ

uid ユーザ ID を保持する文字配列。最大長は 31 文字です。
pwd ユーザ ID のパスワードを保持する文字配列。

戻り値

成功した場合は true、失敗した場合は false。

備考

このメソッドは、既存のユーザ ID を指定したときに、既存のユーザのパスワードを更新します。

関連情報

RevokeConnectFrom(const char *) メソッド [37 ページ]

1.1.27 InitSyncInfo(ul_sync_info *) メソッド

同期情報の構造体を初期化します。

構文

```
public virtual void InitSyncInfo (ul_sync_info * info)
```

パラメータ

info 同期パラメータを保持する ul_sync_info 構造体へのポインタ。

備考

このメソッドを呼び出してから、ul_sync_info structure のフィールドの値を設定するようにしてください。

1.1.28 OpenTable(const char *, const char *) メソッド

テーブルを開きます。

構文

```
public virtual ULTable * OpenTable (
    const char * tableName,
    const char * indexName
)
```

パラメータ

tableName 開くテーブルの名前。
indexName テーブルを開く場合に使用するインデックスの名前。プライマリキーを使用してテーブルを開く場合は NULL、順序付けなしでテーブルを開く場合は空の文字列を渡します。

戻り値

呼び出しが成功した場合は ULTable オブジェクト。失敗した場合は NULL。

備考

アプリケーションがテーブルを初めて開いたときは、カーソルの位置が最初のローの前に設定されます。

1.1.29 PrepareStatement(const char *) メソッド

SQL 文の準備を行います。

構文

```
public virtual ULPreparedStatement * PrepareStatement (const char * sql)
```

パラメータ

sql 準備する SQL 文。

戻り値

成功した場合は ULPreparedStatement オブジェクト、それ以外の場合は NULL。

1.1.30 RegisterForEvent(const char *, const char *, const char *, bool) メソッド

イベントの通知を受け取るためのキューを登録または登録解除します。

構文

```
public virtual bool RegisterForEvent (
    const char * eventName,
    const char * objectName,
    const char * queueName,
    bool register_not_unreg
)
```

パラメータ

eventName 登録するシステム定義またはユーザ定義のイベント。
objectName イベントを適用するオブジェクト。(テーブル名など)。
queueName NULL は、デフォルトの接続キューの使用を表します。
register_not_unreg 登録する場合は true、登録解除する場合は false を設定します。

戻り値

正常に登録できた場合は true。正常に登録できず、キューまたはイベントが存在しない場合は false。

備考

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 メソッドとは異なり、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。別の接続に、同じ名前の他のキューがある場合、それらは、同様に明示的に登録されていないかぎり、通知を受信しません。

事前に定義されたシステムイベントは次のとおりです。

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。要求の影響を受けるローの数にかかわらず、要求ごとに 1 つの通知が送信されます。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。このイベントには、table_name というパラメータがあり、このパラメータの値は変更されたテーブルの名前です。

Commit

コミットが完了した後にトリガされます。このイベントにはパラメータはありません。

SyncComplete

同期が完了した後にトリガされます。このイベントにはパラメータはありません。

1.1.31 ResetLastDownloadTime(const char *) メソッド

アプリケーションが以前にダウンロードされたデータを再同期するように、パブリケーションの最終ダウンロード時間をリセットします。

構文

```
public virtual bool ResetLastDownloadTime (const char * pubList)
```

パラメータ

pubList リセットするパブリケーションのカンマ区切りのリストを含む文字列。空の文字列は、"非同期" とマーク付けされたものを除くすべてのテーブルを意味します。アスタリスクのみの文字列 ("*") は、すべてのパブリケーションを表します。一部のテーブルは、どのパブリケーションの一部でもないため、この値が "*" の場合は含まれません。

戻り値

成功した場合は true、失敗した場合は false。

備考

次のメソッド呼び出しは、すべてのテーブルの最終ダウンロード時間をリセットします。

```
conn->ResetLastDownloadTime( "" );
```

1.1.32 RevokeConnectFrom(const char *) メソッド

Ultra Light データベースからユーザ ID のアクセス権を取り消します。

構文

```
public virtual bool RevokeConnectFrom (const char * uid)
```

パラメータ

uid データベースアクセスから除外するユーザ ID を保持する文字配列。

戻り値

成功した場合は true、失敗した場合は false。

1.1.33 Rollback() メソッド

現在のトランザクションをロールバックします。

構文

```
public virtual bool Rollback ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.1.34 RollbackPartialDownload() メソッド

失敗した同期からの変更をロールバックします。

構文

```
public virtual bool RollbackPartialDownload ()
```

戻り値

成功した場合は true、失敗した場合は false。

備考

再開可能なダウンロードを使用 (Keep Partial Download オプションを有効にして同期) しているときに、同期のダウンロードフェーズ中に通信エラーが発生すると、Ultra Light は、ダウンロードされた変更を保持します (このため、同期は、中断した時点から再開可能です)。ダウンロードを再開しない場合は、このメソッドを使用して、この部分的なダウンロードを破棄します。

このメソッドは、再開可能なダウンロードの使用時にのみ効果があります。

1.1.35 SendNotification(const char *, const char *, const char *) メソッド

指定された名前と一致するすべてのキューに通知を送信します。

構文

```
public virtual ul_u_long SendNotification (
    const char * queueName,
    const char * eventName,
    const char * parameters
)
```

パラメータ

queueName 対象となるキューの名前 (または "*")。
eventName 通知の ID。
parameters オプションのパラメータオプションのリスト。

戻り値

送信済みの通知の数。(一致するキューの数)

備考

これに含まれるのは、現在の接続におけるキューです。この呼び出しはブロックしません。特別なキュー名の "*" を使用すると、すべてのキューに送信します。指定されたイベント名は、システム定義またはユーザ定義のイベントと対応する必要はありません。読み込まれたイベント名は、通知を識別するためにそのまま渡され、送信者と受信者に対してしか意味を持たないからです。

パラメータの値には、"名前=値" のペアをセミコロンで区切ったオプションリストを指定します。通知が読み込まれた後、パラメータの値が GetNotificationParameter メソッドによって読み込まれます。

関連情報

GetNotificationParameter(const char *, const char *) メソッド [30 ページ]

1.1.36 SetDatabaseOption(const char *, const char *) メソッド

指定されたデータベースオプションを設定します。

構文

```
public virtual bool SetDatabaseOption (
    const char * optName,
    const char * value
)
```

パラメータ

optName 設定されるオプションの名前。
value オプションの新しい値。

戻り値

成功した場合は true、失敗した場合は false。

1.1.37 SetDatabaseOptionInt(const char *, ul_u_long) メソッド

データベースオプションを設定します。

構文

```
public virtual bool SetDatabaseOptionInt (
    const char * optName,
    ul_u_long value
)
```

パラメータ

optName 設定されるオプションの名前。
value オプションの新しい値。

戻り値

成功した場合は true、失敗した場合は false。

1.1.38 SetSynchronizationCallback(ul_sync_observer_fn, void *) メソッド

同期の実行中に呼び出されるようコールバックを設定します。

構文

```
public virtual void SetSynchronizationCallback (
    ul_sync_observer_fn callback,
    void * userData
)
```

パラメータ

callback ul_sync_observer_fn コールバック。
userData コールバックに渡されるユーザコンテキスト情報。

1.1.39 SetSyncInfo(char const *, ul_sync_info *) メソッド

指定された ul_sync_info 構造体に基づいて、指定された名前を使用して同期プロファイルを作成します。

構文

```
public virtual bool SetSyncInfo (
    char const * profileName,
    ul_sync_info * info
)
```

パラメータ

profileName 同期プロファイルの名前。
info 同期パラメータを保持する ul_sync_info 構造体へのポインタ。

戻り値

成功した場合は true、失敗した場合は false。

備考

同じ名前の同期プロファイルがすでにある場合は、この同期プロファイルで置き換えられます。構造体に NULL ポインタを指定することによって、指定されたプロファイルが削除されます。

1.1.40 SetUserPointer(void *) メソッド

呼び出すアプリケーションによって接続に使用される任意のポインタ値を設定します。

構文

```
public virtual void * SetUserPointer (void * ptr)
```

戻り値

前に設定されていたポインタ値。

備考

これを使用して、アプリケーションデータを接続に関連付けることができます。

1.1.41 StartSynchronizationDelete() メソッド

この接続の START SYNCHRONIZATION DELETE を設定します。

構文

```
public virtual bool StartSynchronizationDelete ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.1.42 StopSynchronizationDelete() メソッド

この接続の STOP SYNCHRONIZATION DELETE を設定します。

構文

```
public virtual bool StopSynchronizationDelete ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.1.43 Synchronize(ul_sync_info *) メソッド

Ultra Light アプリケーションで同期を開始します。

構文

```
public virtual bool Synchronize (ul_sync_info * info)
```

パラメータ

info 同期パラメータを保持する ul_sync_info 構造体へのポインタ。

戻り値

成功した場合は true、失敗した場合は false。

備考

このメソッドで、Mobile Link サーバとの同期を開始します。このメソッドは、同期が完了するまで戻りませんが、同期中に、別の接続を使用する追加スレッドがデータベースにアクセスし続ける場合があります。

このメソッドを呼び出す前に、ULDatabaseManager クラスの各メソッドで使用しているプロトコルと暗号化を有効にしてください。たとえば、"HTTP" を使用している場合は、ULDatabaseManager.EnableHttpSynchronization メソッドを呼び出します。

次の例は、データベースの同期を示しています。

```
ul_sync_info info;
conn->InitSyncInfo( &info );
info.user_name = "my_user";
info.version = "myapp_1_2";
info.stream = "HTTP";
info.stream_parms = "host=myserver.com";
conn->Synchronize( &info );
```

関連情報

EnableHttpSynchronization() メソッド [53 ページ]

1.1.44 SynchronizeFromProfile(const char *, const char *, ul_sync_observer_fn, void *) メソッド

指定されたプロファイルとマージパラメータを使用して、データベースを同期します。

構文

```
public virtual bool SynchronizeFromProfile (
    const char * profileName,
    const char * mergeParms,
    ul_sync_observer_fn observer,
    void * userData
)
```

パラメータ

profileName 同期するプロファイルの名前。
mergeParms 同期で使用するマージパラメータ。
observer ステータス更新の送信先となる observer コールバック。
userData コールバックに渡されるユーザコンテキストデータ。

戻り値

成功した場合は true、失敗した場合は false。

備考

このメソッドは、SYNCHRONIZE 文を実行するのと同じです。

関連情報

Synchronize(ul_sync_info *) メソッド [43 ページ]

1.1.45 TriggerEvent(const char *, const char *) メソッド

ユーザ定義のイベントをトリガして、登録されたすべてのキューに通知を送信します。

構文

```
public virtual ul_u_long TriggerEvent (
    const char * eventName,
    const char * parameters
)
```

パラメータ

eventName トリガするシステム定義またはユーザ定義のイベントの名前。
parameters オプションのパラメータオプションのリスト。

戻り値

送信済みのイベント通知の数。

備考

パラメータの値には、"名前=値" のペアをセミコロンで区切ったオプションリストを指定します。通知が読み込まれた後、パラメータの値が GetNotificationParameter() によって読み込まれます。

関連情報

GetNotificationParameter(const char *, const char *) メソッド [30 ページ]

1.1.46 ValidateDatabase(ul_u_short, ul_validate_callback_fn, void *, const char *) メソッド

この接続でのデータベースを検証します。

構文

```
public virtual bool ValidateDatabase (
    ul_u_short flags,
    ul_validate_callback_fn fn,
    void * user_data,
    const char * tableName
)
```

パラメータ

flags 検証のタイプを制御するフラグ。後述の例を参照してください。

fn 検証の進行状況の情報を受け取る関数。

user_data コールバックにより呼び出し元に送り返すユーザデータ。

tableName 省略可能です。検証する特定のテーブル。

戻り値

成功した場合は true、失敗した場合は false。

備考

このルーチンに渡されるフラグに応じて、テーブル、インデックス、およびデータベースページを検証できます。検証中に情報を受け取るには、コールバック関数を実装し、アドレスをこのルーチンに渡します。検証対象を特定のテーブルに限定するには、テーブルの名前または ID を最後のパラメータとして渡します。

flags パラメータは、次のいずれかの値の組み合わせです。

- ULVF_TABLE
- ULVF_INDEX

- ULVF_DATABASE
- ULVF_EXPRESS
- ULVF_FULL_VALIDATE

例

次の例は、エクスプレスモードでのテーブルとインデックスの検証を示します。

```
flags = ULVF_TABLE | ULVF_INDEX | ULVF_EXPRESS;
```

1.2 ULDatabaseManager クラス

接続とデータベースを管理します。

構文

```
public class ULDatabaseManager
```

メンバー

ULDatabaseManager のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public static ULConnection *	CreateDatabase(const char *, const char *, ULError *) [50 ページ]	新しいデータベースを作成します。
public static bool	DropDatabase(const char *, ULError *) [51 ページ]	現在実行していない既存のデータベースを消去します。
public static void	EnableAesDBEncryption() [51 ページ]	AES データベース暗号化を有効にします。
public static void	EnableAesFipsDBEncryption() [52 ページ]	FIPS 140-2 認定 AES データベース暗号化を有効にします。
public static void	EnableAllSynchronization() [52 ページ]	4 種類の同期、TCPIP、TLS、HTTP、HTTPS をすべて有効にします。
public static void	EnableHttpsSynchronization() [53 ページ]	HTTPS 同期を有効にします。
public static void	EnableHttpSynchronization() [53 ページ]	HTTP 同期を有効にします。
public static void	EnableRsaE2ee() [53 ページ]	RSA エンドツーエンド暗号化を有効にします。

変更子とタイプ	メソッド	説明
public static void	EnableRsaFipsE2ee() [54 ページ]	FIPS 140-2 認定 RSA エンドツーエンド暗号化を有効にします。
public static void	EnableRsaFipsSyncEncryption() [54 ページ]	SSL ストリームまたは TLS ストリームの FIPS 140-2 認定 RSA 同期暗号化を有効にします。
public static void	EnableRsaSyncEncryption() [55 ページ]	RSA 同期暗号化を有効にします。
public static void	EnableTcpipSynchronization() [55 ページ]	TCP/IP 同期を有効にします。
public static void	EnableTlsSynchronization() [56 ページ]	TLS 同期を有効にします。
public static void	EnableZlibSyncCompression() [56 ページ]	同期ストリームの ZLIB 圧縮を有効にします。
public static void	Fini() [56 ページ]	Ultra Light ランタイムをファイナライズします。
public static bool	Init() [57 ページ]	Ultra Light ランタイムを初期化します。
public static ULConnection *	OpenConnection(const char *, ULError *, void *) [57 ページ]	既存のデータベースへの新しい接続を開きます。
public static void	SetErrorCallback(ul_cpp_error_callback_fn, void *) [58 ページ]	エラーの発生時に呼び出されるようコールバックを設定します。
public static bool	ValidateDatabase(const char *, ul_u_short, ul_validate_callback_fn, void *, ULError *) [59 ページ]	データベースで低レベルのインデックス検証を実行します。

備考

スレッド対応環境で Init メソッドを呼び出してから、他の呼び出しを行う必要があります。終了したら、同様にスレッド対応環境で Fini メソッドを呼び出してください。

> **注記**
>
> このクラスは静的です。このクラスのインスタンスを作成しないでください。

このセクションの内容:

CreateDatabase(const char *, const char *, ULError *) メソッド [50 ページ]
: 新しいデータベースを作成します。

DropDatabase(const char *, ULError *) メソッド [51 ページ]
: 現在実行していない既存のデータベースを消去します。

EnableAesDBEncryption() メソッド [51 ページ]
: AES データベース暗号化を有効にします。

1.2.1 CreateDatabase(const char *, const char *, ULError *) メソッド

新しいデータベースを作成します。

構文

```
public static ULConnection * CreateDatabase (
    const char * connParms,
    const char * createParms,
    ULError * error
)
```

パラメータ

connParms セミコロンで区切った接続パラメータ文字列で、キーワード=値のペアで設定されます。接続文字列には、データベースの名前を含める必要があります。ここに含まれるパラメータは、データベースの接続時に指定されるパラメータセットと同じです。

createParms データベース作成パラメータをセミコロンで区切った文字列。キーワードと値のペアとして設定されます。例: page_size=2048;obfuscate=yes。

error エラー情報を受信するためのオプションの ULError オブジェクト。

戻り値

データベースが正常に作成された場合は、新しいデータベースへの ULConnection オブジェクトが返されます。メソッドが失敗した場合は、NULL が返されます。通常、失敗の原因は、無効なファイル名やアクセスの拒否です。

備考

2 セットのパラメータで指定される情報を使用してデータベースが作成されます。

connParms パラメータは、ファイル名や暗号化キーなど、データベースへのアクセス時に必ず適用される一連の標準接続パラメータです。

createParms パラメータは、チェックサムレベル、ページサイズ、照合、時刻と日付の形式など、データベースの作成時にのみ意味を持つ一連のパラメータです。

次のコードは、CreateDatabase メソッドを使用して、ファイル `mydb.udb` に Ultra Light データベースを作成する方法を示します。

```
ULConnection * conn;
conn = ULDatabaseManager::CreateDatabase( "DBF=mydb.udb", "checksum_level=2" );
if( conn != NULL ) {
```

```
   // success
} else {
   // unable to create
}
```

1.2.2 DropDatabase(const char *, ULError *) メソッド

現在実行していない既存のデータベースを消去します。

構文

```
public static bool DropDatabase (
    const char * parms,
    ULError * error
)
```

パラメータ

parms データベース識別パラメータ。(接続文字列)
error エラー情報を受信するためのオプションの ULError オブジェクト。

戻り値

データベースが正常に削除された場合は true、正常に削除されなかった場合は false。

1.2.3 EnableAesDBEncryption() メソッド

AES データベース暗号化を有効にします。

構文

```
public static void EnableAesDBEncryption ()
```

備考

このメソッドを呼び出して、AES データベース暗号化を使用します。DBKEY 接続パラメータを使用して、暗号化パスフレーズを指定します。このメソッドを呼び出してからデータベース接続を開くようにしてください。

1.2.4 EnableAesFipsDBEncryption() メソッド

FIPS 140-2 認定 AES データベース暗号化を有効にします。

構文

```
public static void EnableAesFipsDBEncryption ()
```

備考

このメソッドを呼び出して、FIPS AES データベース暗号化を使用します。DBKEY 接続パラメータを使用して、暗号化パスフレーズを指定します。

データベース作成パラメータ文字列には、'fips=yes' を指定する必要があります。このメソッドを呼び出してからデータベース接続を開くようにしてください。

関連情報

EnableAesDBEncryption() メソッド [51 ページ]

1.2.5 EnableAllSynchronization() メソッド

4 種類の同期、TCPIP、TLS、HTTP、HTTPS をすべて有効にします。

構文

```
public static void EnableAllSynchronization ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

同期を開始するときは、ストリームパラメータを "TCPIP"、"HTTP"、"TLS"、または "HTTPS" に設定します。TLS または HTTPS を使用する場合は、ネットワークプロトコル証明書のオプションも設定します。

1.2.6 EnableHttpsSynchronization() メソッド

HTTPS 同期を有効にします。

構文

```
public static void EnableHttpsSynchronization ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

同期を開始するときは、ストリームパラメータを "HTTPS" に設定します。また、ネットワークプロトコル証明書オプションも設定します。

1.2.7 EnableHttpSynchronization() メソッド

HTTP 同期を有効にします。

構文

```
public static void EnableHttpSynchronization ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

同期を開始するときは、ストリームパラメータを "HTTP" に設定します。

1.2.8 EnableRsaE2ee() メソッド

RSA エンドツーエンド暗号化を有効にします。

構文

```
public static void EnableRsaE2ee ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

エンドツーエンド暗号化を使用するには、*e2ee_public_key* ネットワークプロトコルオプションを設定します。

1.2.9 EnableRsaFipsE2ee() メソッド

FIPS 140-2 認定 RSA エンドツーエンド暗号化を有効にします。

構文

```
public static void EnableRsaFipsE2ee ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

エンドツーエンド暗号化を使用するには、*e2ee_public_key* ネットワークプロトコルオプションを設定します。この場合は、*FIPS* オプションを "yes" に設定する必要があります。

1.2.10 EnableRsaFipsSyncEncryption() メソッド

SSL ストリームまたは TLS ストリームの FIPS 140-2 認定 RSA 同期暗号化を有効にします。

構文

```
public static void EnableRsaFipsSyncEncryption ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

これは、FIPS RSA 暗号化に対して ストリーム パラメータを "TLS" または "HTTPS" に設定するときに必要です。この場合は、*FIPS* オプションを "yes" に設定する必要があります。

関連情報

EnableRsaSyncEncryption() メソッド [55 ページ]

1.2.11 EnableRsaSyncEncryption() メソッド

RSA 同期暗号化を有効にします。

構文

```
public static void EnableRsaSyncEncryption ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

これは、RSA 暗号化に対して ストリームパラメータを "TLS" または "HTTPS" に設定するときに必要です。

1.2.12 EnableTcpipSynchronization() メソッド

TCP/IP 同期を有効にします。

構文

```
public static void EnableTcpipSynchronization ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

同期を開始するときは、ストリームパラメータを "TCPIP" に設定します。

1.2.13 EnableTlsSynchronization() メソッド

TLS 同期を有効にします。

構文

```
public static void EnableTlsSynchronization ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

同期を開始するときは、ストリームパラメータを "TLS" に設定します。また、ネットワークプロトコル証明書オプションも設定します。

1.2.14 EnableZlibSyncCompression() メソッド

同期ストリームの ZLIB 圧縮を有効にします。

構文

```
public static void EnableZlibSyncCompression ()
```

備考

このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。

圧縮を使用するには、圧縮ネットワークプロトコルオプションを "zlib" に設定します。

1.2.15 Fini() メソッド

Ultra Light ランタイムをファイナライズします。

構文

```
public static void Fini ()
```

備考

このメソッドは、アプリケーションの終了時に、単一のスレッドで 1 度だけ呼び出す必要があります。このメソッドはスレッド対応ではありません。

1.2.16 Init() メソッド

Ultra Light ランタイムを初期化します。

構文

```
public static bool Init ()
```

戻り値

成功した場合は true、失敗した場合は false。また、メソッドを複数回呼び出した場合にも、false が返されます。

備考

このメソッドは、その他の呼び出しを行う前に、単一のスレッドで 1 度だけ呼び出す必要があります。このメソッドはスレッド対応ではありません。

通常、メモリが使用可能であるかぎり、このメソッドは失敗しません。

1.2.17 OpenConnection(const char *, ULError *, void *) メソッド

既存のデータベースへの新しい接続を開きます。

構文

```
public static ULConnection * OpenConnection (
    const char * connParms,
    ULError * error,
    void * reserved
)
```

パラメータ

connParms 接続文字列。
error エラー情報を返すためのオプションの ULError オブジェクト。
reserved 内部用に予約されています。除外または NULL に設定されます。

戻り値

メソッドが成功した場合は、新しい ULConnection オブジェクト。成功しなかった場合は NULL。

備考

接続文字列は、どのデータベースに接続するかを示す option=value 接続パラメータ (セミコロンで区切られた) および接続に使用するオプションのセットです。たとえば、暗号化パスフレーズを安全に取得した後に得られる接続文字列は、"DBF=mydb.udb;DBKEY=iyntTZId9OEa#&G" のようになります。

エラー情報を取得するには、ULError オブジェクトへのポインタを渡します。次に、可能性のあるエラーのリストを示します。

SQLE_INVALID_PARSE_PARAMETER

connParms が正しくフォーマットされていません。
SQLE_UNRECOGNIZED_OPTION

接続オプション名のスペルを間違えた可能性があります。
SQLE_INVALID_OPTION_VALUE

接続オプション値が正しく指定されていません。
SQLE_ULTRALITE_DATABASE_NOT_FOUND

指定されたデータベースが見つかりませんでした。
SQLE_INVALID_LOGON

無効なユーザ ID または間違ったパスワードを入力しました。
SQLE_TOO_MANY_CONNECTIONS

同時データベース接続の最大数を超えました。

1.2.18 SetErrorCallback(ul_cpp_error_callback_fn, void *) メソッド

エラーの発生時に呼び出されるようコールバックを設定します。

構文

```
public static void SetErrorCallback (
```

```
    ul_cpp_error_callback_fn callback,
    void * userData
)
```

パラメータ

callback コールバック関数。
userData コールバックに渡されるユーザコンテキスト情報。

備考

このメソッドはスレッド対応ではありません。

1.2.19 ValidateDatabase(const char *, ul_u_short, ul_validate_callback_fn, void *, ULError *) メソッド

データベースで低レベルのインデックス検証を実行します。

構文

```
public static bool ValidateDatabase (
    const char * connParms,
    ul_u_short flags,
    ul_validate_callback_fn fn,
    void * userData,
    ULError * error
)
```

パラメータ

connParms データベースへの接続に使用されるパラメータ。
flags 検証のタイプを制御するフラグ。次の例を参照してください。
fn 検証の進行状況の情報を受け取る関数。
userData コールバックにより呼び出し元に送り返すユーザデータ。
error エラー情報を受信するためのオプションの ULError オブジェクト。

戻り値

検証が成功した場合は true、そうでない場合は false。

備考

flags パラメータは、次のいずれかの値の組み合わせです。

- ULVF_TABLE
- ULVF_INDEX
- ULVF_DATABASE
- ULVF_EXPRESS
- ULVF_FULL_VALIDATE

例

次の例は、エクスプレスモードでのテーブルとインデックスの検証を示します。

```
flags = ULVF_TABLE | ULVF_INDEX | ULVF_EXPRESS;
```

1.3 ULDatabaseSchema クラス

Ultra Light データベースのスキーマを表します。

構文

```
public class ULDatabaseSchema
```

メンバー

ULDatabaseSchema のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public virtual void	Close() [61 ページ]	このオブジェクトを破棄します。
public virtual ULConnection *	GetConnection() [62 ページ]	ULConnection オブジェクトを取得します。

変更子とタイプ	メソッド	説明
public virtual const char *	GetNextPublication(ul_publication_iter *) [62 ページ]	データベース内の次のパブリケーションの名前を取得します。
public virtual ULTableSchema *	GetNextTable(ul_table_iter *) [63 ページ]	データベース内の次のテーブル (スキーマ) を取得します。
public virtual ul_publication_count	GetPublicationCount() [64 ページ]	データベース内のパブリケーション数を取得します。
public virtual ul_table_num	GetTableCount() [64 ページ]	データベース内のテーブルの数を返します。
public virtual ULTableSchema *	GetTableSchema(const char *) [64 ページ]	指定したテーブルのスキーマを返します。

このセクションの内容:

Close() メソッド [61 ページ]
　このオブジェクトを破棄します。

GetConnection() メソッド [62 ページ]
　ULConnection オブジェクトを取得します。

GetNextPublication(ul_publication_iter *) メソッド [62 ページ]
　データベース内の次のパブリケーションの名前を取得します。

GetNextTable(ul_table_iter *) メソッド [63 ページ]
　データベース内の次のテーブル (スキーマ) を取得します。

GetPublicationCount() メソッド [64 ページ]
　データベース内のパブリケーション数を取得します。

GetTableCount() メソッド [64 ページ]
　データベース内のテーブルの数を返します。

GetTableSchema(const char *) メソッド [64 ページ]
　指定したテーブルのスキーマを返します。

1.3.1 Close() メソッド

このオブジェクトを破棄します。

構文

```
public virtual void Close ()
```

1.3.2 GetConnection() メソッド

ULConnection オブジェクトを取得します。

構文

```
public virtual ULConnection * GetConnection ()
```

戻り値

このオブジェクトに関連付けられている ULConnection。

1.3.3 GetNextPublication(ul_publication_iter *) メソッド

データベース内の次のパブリケーションの名前を取得します。

構文

```
public virtual const char * GetNextPublication (ul_publication_iter * iter)
```

パラメータ

iter 繰り返し変数へのポインタ。

戻り値

次のパブリケーションの名前。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。反復が完了すると NULL が返されます。

備考

最初の呼び出しの前に、iter 値を ul_publication_iter_start 定数に初期化します。

関連情報

ul_publication_iter_start 変数 [187 ページ]

1.3.4 GetNextTable(ul_table_iter *) メソッド

データベース内の次のテーブル (スキーマ) を取得します。

構文

```
public virtual ULTableSchema * GetNextTable (ul_table_iter * iter)
```

パラメータ

iter 繰り返し変数へのポインタ。

戻り値

反復が完了したときに ULTableSchema オブジェクトまたは NULL。

備考

最初の呼び出しの前に、iter 値を ul_table_iter_start 定数に初期化します。

関連情報

ul_table_iter_start 変数 [187 ページ]

1.3.5 GetPublicationCount() メソッド

データベース内のパブリケーション数を取得します。

構文

```
public virtual ul_publication_count GetPublicationCount ()
```

戻り値

データベース内のパブリケーションの数です。

備考

パブリケーション ID の範囲は 1 から、このメソッドが返す数字までです。

1.3.6 GetTableCount() メソッド

データベース内のテーブルの数を返します。

構文

```
public virtual ul_table_num GetTableCount ()
```

戻り値

テーブルの数を表す整数。

1.3.7 GetTableSchema(const char *) メソッド

指定したテーブルのスキーマを返します。

構文

```
public virtual ULTableSchema * GetTableSchema (const char * tableName)
```

パラメータ

tableName テーブル名。

戻り値

特定のテーブルの場合は ULTableSchema オブジェクト。それ以外で、テーブルが存在しない場合は UL_NULL。

1.4 ULError クラス

Ultra Light ランタイムから返されたエラーを管理します。

構文

```
public class ULError
```

メンバー

ULError のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

コンストラクタ

変更子とタイプ	コンストラクタ	説明
public	ULError() [66 ページ]	ULError オブジェクトを構築します。

メソッド

変更子とタイプ	メソッド	説明
public void	Clear() [67 ページ]	現在のエラーをクリアします。
public const ul_error_info *	GetErrorInfo [67 ページ]	基本となる ul_error_info オブジェクトへのポインタを返します。
public size_t	GetParameter(ul_u_short, char *, size_t) [68 ページ]	指定されたバッファに、指定されたエラーパラメータをコピーします。
public ul_u_short	GetParameterCount() [69 ページ]	エラーパラメータの数を返します。
public an_sql_code	GetSQLCode() [69 ページ]	最後の操作の SQLCODE エラーコードを返します。
public ul_s_long	GetSQLCount() [70 ページ]	最後の操作とその操作の結果によって異なる値を返します。

変更子とタイプ	メソッド	説明
public size_t	GetString(char *, size_t) [70 ページ]	現在のエラーの説明を返します。
public size_t	GetURL(char *, size_t, const char *) [71 ページ]	このエラーの資料ページの URL を返します。
public bool	IsOK() [72 ページ]	エラーコードをテストします。

このセクションの内容:

ULError() コンストラクタ [66 ページ]
ULError オブジェクトを構築します。

Clear() メソッド [67 ページ]
現在のエラーをクリアします。

GetErrorInfo メソッド [67 ページ]
基本となる ul_error_info オブジェクトへのポインタを返します。

GetParameter(ul_u_short, char *, size_t) メソッド [68 ページ]
指定されたバッファに、指定されたエラーパラメータをコピーします。

GetParameterCount() メソッド [69 ページ]
エラーパラメータの数を返します。

GetSQLCode() メソッド [69 ページ]
最後の操作の SQLCODE エラーコードを返します。

GetSQLCount() メソッド [70 ページ]
最後の操作とその操作の結果によって異なる値を返します。

GetString(char *, size_t) メソッド [70 ページ]
現在のエラーの説明を返します。

GetURL(char *, size_t, const char *) メソッド [71 ページ]
このエラーの資料ページの URL を返します。

IsOK() メソッド [72 ページ]
エラーコードをテストします。

1.4.1 ULError() コンストラクタ

ULError オブジェクトを構築します。

構文

```
public ULError ()
```

1.4.2 Clear() メソッド

現在のエラーをクリアします。

構文

```
public void Clear ()
```

備考

現在のエラーは、ほとんどの呼び出しで自動的にクリアされます。そのため、通常、このメソッドがアプリケーションによって呼び出されることはありません。

1.4.3 GetErrorInfo メソッド

基本となる ul_error_info オブジェクトへのポインタを返します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public const ul_error_info *	GetErrorInfo() [68 ページ]	基本となる ul_error_info オブジェクトへのポインタを返します。
public ul_error_info	GetErrorInfo() [68 ページ]	基本となる ul_error_info オブジェクトへのポインタを返します。

このセクションの内容:

GetErrorInfo() メソッド [68 ページ]
基本となる ul_error_info オブジェクトへのポインタを返します。

GetErrorInfo() メソッド [68 ページ]
基本となる ul_error_info オブジェクトへのポインタを返します。

1.4.3.1 GetErrorInfo() メソッド

基本となる ul_error_info オブジェクトへのポインタを返します。

構文

```
public const ul_error_info * GetErrorInfo ()
```

戻り値

基本となる ul_error_info オブジェクトへのポインタ。

1.4.3.2 GetErrorInfo() メソッド

基本となる ul_error_info オブジェクトへのポインタを返します。

構文

```
public ul_error_info * GetErrorInfo ()
```

戻り値

基本となる ul_error_info オブジェクトへのポインタ。

1.4.4 GetParameter(ul_u_short, char *, size_t) メソッド

指定されたバッファに、指定されたエラーパラメータをコピーします。

構文

```
public size_t GetParameter (
    ul_u_short parmNo,
    char * dst,
    size_t len
)
```

パラメータ

parmNo 1 から始まるパラメータ番号。
dst パラメータを受け取るバッファ。
len バッファのサイズ。

戻り値

パラメータの格納に必要なサイズ。または、序数が無効の場合は 0。戻り値が len 値より大きい場合、パラメータはトランケートされます。

備考

バッファが小さすぎて文字列がトランケートされる場合でも、出力文字列は常に NULL で終了します。

1.4.5 GetParameterCount() メソッド

エラーパラメータの数を返します。

構文

```
public ul_u_short GetParameterCount ()
```

戻り値

エラーパラメータ数。

1.4.6 GetSQLCode() メソッド

最後の操作の SQLCODE エラーコードを返します。

構文

```
public an_sql_code GetSQLCode ()
```

戻り値

sqlcode 値。

1.4.7 GetSQLCount() メソッド

最後の操作とその操作の結果によって異なる値を返します。

構文

```
public ul_s_long GetSQLCount ()
```

戻り値

適用される場合は、最後の操作の値。それ以外で、適用されない場合は -1。

備考

次のリストは、考えられる操作と、返される結果の概要を示します。

INSERT, UPDATE, or DELETE operation executed successfully

文によって影響を受けたローの数を返します。

SQL statement syntax error (SQLE_SYNTAX_ERROR)

文内のおおよそのエラー検出位置を返します。

1.4.8 GetString(char *, size_t) メソッド

現在のエラーの説明を返します。

構文

```
public size_t GetString (
    char * dst,
    size_t len
)
```

パラメータ

dst エラーの説明を受信するバッファ。
len バッファのサイズ (配列の要素数)。

戻り値

文字列の格納に必要なサイズ。戻り値が len 値より大きい場合、文字列はトランケートされます。

備考

文字列には、エラーコードとすべてのパラメータが含まれます。ULError.GetURL メソッドで返された URL をロードすると、エラーの詳細な説明が得られます。

バッファが小さすぎて文字列がトランケートされる場合であっても、出力文字列は常に NULL で終了します。

関連情報

GetURL(char *, size_t, const char *) メソッド [71 ページ]

1.4.9 GetURL(char *, size_t, const char *) メソッド

このエラーの資料ページの URL を返します。

構文

```
public size_t GetURL (
    char * buffer,
    size_t len,
    const char * reserved
)
```

パラメータ

buffer URL を受け取るバッファ。
len バッファのサイズ。
reserved 今後の使用のために予約されているため、デフォルトの NULL を渡す必要があります。

戻り値

URL の格納に必要なサイズ。戻り値が len より大きい場合、URL はトランケートされます。

1.4.10 IsOK() メソッド

エラーコードをテストします。

構文

```
public bool IsOK ()
```

戻り値

現在のコードが SQLE_NOERROR か、警告の場合は true。それ以外で、現在のコードがエラーを示している場合は、false。

1.5 ULIndexSchema クラス

Ultra Light テーブルのインデックスのスキーマを表します。

構文

```
public class ULIndexSchema
```

メンバー

ULIndexSchema のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public virtual void	Close() [74 ページ]	このオブジェクトを破棄します。
public virtual ul_column_num	GetColumnCount() [74 ページ]	インデックス内のカラム数を取得します。
public virtual const char *	GetColumnName(ul_column_num) [74 ページ]	インデックス内のカラムの位置を指定して、カラムの名前を取得します。

変更子とタイプ	メソッド	説明
public virtual ULConnection *	GetConnection() [75 ページ]	ULConnection オブジェクトを取得します。
public virtual ul_column_num	GetIndexColumnID(const char *) [75 ページ]	1 から始まるインデックスカラム ID を名前から取得します。
public virtual ul_index_flag	GetIndexFlags() [76 ページ]	インデックスプロパティフラグのビットフィールドを取得します。
public virtual const char *	GetName() [76 ページ]	インデックスの名前を取得します。
public virtual const char *	GetReferencedIndexName() [76 ページ]	関連付けられているプライマリインデックスの名前を取得します。
public virtual const char *	GetReferencedTableName() [77 ページ]	関連付けられているプライマリテーブルの名前を取得します。
public virtual const char *	GetTableName() [77 ページ]	このインデックスが含まれるテーブルの名前を取得します。
public virtual bool	IsColumnDescending(ul_column_num) [78 ページ]	カラムが降順かどうかを調べます。

このセクションの内容:

Close() メソッド [74 ページ]
このオブジェクトを破棄します。

GetColumnCount() メソッド [74 ページ]
インデックス内のカラム数を取得します。

GetColumnName(ul_column_num) メソッド [74 ページ]
インデックス内のカラムの位置を指定して、カラムの名前を取得します。

GetConnection() メソッド [75 ページ]
ULConnection オブジェクトを取得します。

GetIndexColumnID(const char *) メソッド [75 ページ]
1 から始まるインデックスカラム ID を名前から取得します。

GetIndexFlags() メソッド [76 ページ]
インデックスプロパティフラグのビットフィールドを取得します。

GetName() メソッド [76 ページ]
インデックスの名前を取得します。

GetReferencedIndexName() メソッド [76 ページ]
関連付けられているプライマリインデックスの名前を取得します。

GetReferencedTableName() メソッド [77 ページ]
関連付けられているプライマリテーブルの名前を取得します。

GetTableName() メソッド [77 ページ]
このインデックスが含まれるテーブルの名前を取得します。

IsColumnDescending(ul_column_num) メソッド [78 ページ]
カラムが降順かどうかを調べます。

1.5.1 Close() メソッド

このオブジェクトを破棄します。

構文

```
public virtual void Close ()
```

1.5.2 GetColumnCount() メソッド

インデックス内のカラム数を取得します。

構文

```
public virtual ul_column_num GetColumnCount ()
```

戻り値

インデックス内のカラム数。

1.5.3 GetColumnName(ul_column_num) メソッド

インデックス内のカラムの位置を指定して、カラムの名前を取得します。

構文

```
public virtual const char * GetColumnName (ul_column_num col_id_in_index)
```

パラメータ

col_id_in_index インデックス内のカラムの位置を示す 1 から始まる順序数。

戻り値

カラム名。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。

1.5.4 GetConnection() メソッド

ULConnection オブジェクトを取得します。

構文

```
public virtual ULConnection * GetConnection ()
```

戻り値

このオブジェクトに関連付けられている接続。

1.5.5 GetIndexColumnID(const char *) メソッド

1 から始まるインデックスカラム ID を名前から取得します。

構文

```
public virtual ul_column_num GetIndexColumnID (const char * columnName)
```

パラメータ

columnName カラムの名前。

戻り値

0。また、カラム名が存在しない場合は、SQLE_COLUMN_NOT_FOUND を設定します。

1.5.6 GetIndexFlags() メソッド

インデックスプロパティフラグのビットフィールドを取得します。

構文

```
public virtual ul_index_flag GetIndexFlags ()
```

関連情報

ul_index_flag 列挙 [184 ページ]

1.5.7 GetName() メソッド

インデックスの名前を取得します。

構文

```
public virtual const char * GetName ()
```

戻り値

インデックスの名前。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。

1.5.8 GetReferencedIndexName() メソッド

関連付けられているプライマリインデックスの名前を取得します。

構文

```
public virtual const char * GetReferencedIndexName ()
```

戻り値

参照先インデックスの名前。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。

備考

このメソッドは、外部キーにのみ適用されます。

1.5.9 GetReferencedTableName() メソッド

関連付けられているプライマリテーブルの名前を取得します。

構文

```
public virtual const char * GetReferencedTableName ()
```

戻り値

参照先テーブルの名前。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。

備考

このメソッドは、外部キーにのみ適用されます。

1.5.10 GetTableName() メソッド

このインデックスが含まれるテーブルの名前を取得します。

構文

```
public virtual const char * GetTableName ()
```

戻り値

このインデックスが含まれるテーブルの名前。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。

1.5.11 IsColumnDescending(ul_column_num) メソッド

カラムが降順かどうかを調べます。

構文

```
public virtual bool IsColumnDescending (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムが降順の場合は true、昇順の場合は false。

1.6 ULPreparedStatement クラス

準備された SQL 文を表します。

構文

```
public class ULPreparedStatement
```

メンバー

ULPreparedStatement のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public virtual bool	AppendParameterByteChunk(ul_column_num, const ul_byte *, size_t) [81 ページ]	複数のチャンクに分解される、サイズの大きい Binary パラメータを設定します。
public virtual bool	AppendParameterStringChunk(ul_column_num, const char *, size_t) [82 ページ]	複数のチャンクに分解される、サイズの大きい String パラメータを設定します。
public virtual void	Close() [82 ページ]	このオブジェクトを破棄します。
public virtual ULResultSet *	ExecuteQuery() [83 ページ]	SQL SELECT 文をクエリとして実行します。
public virtual bool	ExecuteStatement() [83 ページ]	SQL INSERT 文、DELETE 文、UPDATE 文のように、結果セットを返さない文を実行します。
public virtual ULConnection *	GetConnection() [83 ページ]	接続オブジェクトを取得します。
public virtual ul_u_short	GetParameterCount() [84 ページ]	この文の入力パラメータの数を取得します。
public virtual ul_column_num	GetParameterID(const char *) [84 ページ]	1 から始まるパラメータ名の順序数を取得します。
public virtual ul_column_storage_type	GetParameterType(ul_column_num) [85 ページ]	パラメータの記憶タイプまたはホスト変数の型を取得します。
public virtual size_t	GetPlan(char *, size_t) [85 ページ]	クエリ実行プランのテキストベースの記述を取得します。
public virtual const ULResultSetSchema &	GetResultSetSchema() [86 ページ]	結果セットのスキーマを取得します。
public virtual ul_s_long	GetRowsAffectedCount() [86 ページ]	最後の文の影響を受けるローの数を取得します。
public virtual bool	HasResultSet() [87 ページ]	SQL 文に結果セットがあるかどうかを調べます。
public virtual bool	SetParameterBinary(ul_column_num, const p_ul_binary) [87 ページ]	パラメータを ul_binary 値に設定します。
public virtual bool	SetParameterDateTime(ul_column_num, DECL_DATETIME *) [88 ページ]	パラメータを DECL_DATETIME 値に設定します。
public virtual bool	SetParameterDouble(ul_column_num, ul_double) [89 ページ]	パラメータを double 値に設定します。
public virtual bool	SetParameterFloat(ul_column_num, ul_real) [89 ページ]	パラメータを float 値に設定します。
public virtual bool	SetParameterGuid(ul_column_num, GUID *) [90 ページ]	パラメータを GUID 値に設定します。
public virtual bool	SetParameterInt(ul_column_num, ul_s_long) [90 ページ]	パラメータを整数値に設定します。
public virtual bool	SetParameterIntWithType(ul_column_num, ul_s_big, ul_column_storage_type) [91 ページ]	パラメータを、指定された整数型の整数値に設定します。

変更子とタイプ	メソッド	説明
public virtual bool	SetParameterNull(ul_column_num) [92 ページ]	パラメータを NULL に設定します。
public virtual bool	SetParameterString(ul_column_num, const char *, size_t) [92 ページ]	パラメータを文字列値に設定します。

このセクションの内容:

AppendParameterByteChunk(ul_column_num, const ul_byte *, size_t) メソッド [81 ページ]
複数のチャンクに分解される、サイズの大きい Binary パラメータを設定します。

AppendParameterStringChunk(ul_column_num, const char *, size_t) メソッド [82 ページ]
複数のチャンクに分解される、サイズの大きい String パラメータを設定します。

Close() メソッド [82 ページ]
このオブジェクトを破棄します。

ExecuteQuery() メソッド [83 ページ]
結果セットを返す準備された文を実行します。

ExecuteStatement() メソッド [83 ページ]
SQL INSERT 文、DELETE 文、UPDATE 文のように、結果セットを返さない文を実行します。

GetConnection() メソッド [83 ページ]
接続オブジェクトを取得します。

GetParameterCount() メソッド [84 ページ]
準備した文の入力パラメータの数を取得します。

GetParameterID(const char *) メソッド [84 ページ]
1 から始まるパラメータ名の順序数を取得します。

GetParameterType(ul_column_num) メソッド [85 ページ]
パラメータの記憶タイプまたはホスト変数の型を取得します。

GetPlan(char *, size_t) メソッド [85 ページ]
クエリ実行プランのテキストベースの記述を取得します。

GetResultSetSchema() メソッド [86 ページ]
結果セットのスキーマを取得します。

HasResultSet() メソッド [86 ページ]
最後の文の影響を受けるローの数を取得します。

HasResultSet() メソッド [87 ページ]
SQL 文に結果セットがあるかどうかを調べます。

SetParameterBinary(ul_column_num, const p_ul_binary) メソッド [87 ページ]
パラメータを ul_binary 値に設定します。

SetParameterBytes(ul_column_num pid, const ul_byte * value, size_t len) [88 ページ]
パラメータを byte 配列値に設定します。

SetParameterDateTime(ul_column_num, DECL_DATETIME *) メソッド [88 ページ]
パラメータを DECL_DATETIME 値に設定します。

SetParameterDouble(ul_column_num, ul_double) メソッド [89 ページ]
パラメータを double 値に設定します。

SetParameterFloat(ul_column_num, ul_real) メソッド [89 ページ]
パラメータを float 値に設定します。

SetParameterGuid(ul_column_num, GUID *) メソッド [90 ページ]
パラメータを GUID 値に設定します。

SetParameterInt(ul_column_num, ul_s_long) メソッド [90 ページ]
パラメータを整数値に設定します。

SetParameterIntWithType(ul_column_num, ul_s_big, ul_column_storage_type) メソッド [91 ページ]
パラメータを、指定された整数型の整数値に設定します。

SetParameterNull(ul_column_num) メソッド [92 ページ]
パラメータを NULL に設定します。

SetParameterString(ul_column_num, const char *, size_t) メソッド [92 ページ]
パラメータを文字列値に設定します。

1.6.1 AppendParameterByteChunk(ul_column_num, const ul_byte *, size_t) メソッド

複数のチャンクに分解される、サイズの大きい Binary パラメータを設定します。

構文

```
public virtual bool AppendParameterByteChunk (
    ul_column_num pid,
    const ul_byte * value,
    size_t valueSize
)
```

パラメータ

pid パラメータの、1 から始まる序数。
value 追加するバイトのチャンク。
valueSize バッファのサイズ。

戻り値

成功した場合は true、失敗した場合は false。

1.6.2 AppendParameterStringChunk(ul_column_num, const char *, size_t) メソッド

複数のチャンクに分解される、サイズの大きい String パラメータを設定します。

構文

```
public virtual bool AppendParameterStringChunk (
    ul_column_num pid,
    const char * value,
    size_t len
)
```

パラメータ

pid パラメータの、1 から始まる序数。

value 追加する文字列のチャンク。

len 省略可能です。文字列のチャンクの長さ (バイト単位)、または文字列のチャンクが NULL で終了する場合は UL_NULL_TERMINATED_STRING に設定されます。

戻り値

成功した場合は true、失敗した場合は false。

1.6.3 Close() メソッド

このオブジェクトを破棄します。

構文

```
public virtual void Close ()
```

1.6.4 ExecuteQuery() メソッド

結果セットを返す準備された文を実行します。

構文

```
public virtual ULResultSet * ExecuteQuery ()
```

戻り値

クエリの結果 (ローのセット) を含む ULResultSet オブジェクト。

1.6.5 ExecuteStatement() メソッド

SQL INSERT 文、DELETE 文、UPDATE 文のように、結果セットを返さない文を実行します。

構文

```
public virtual bool ExecuteStatement ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.6.6 GetConnection() メソッド

接続オブジェクトを取得します。

構文

```
public virtual ULConnection * GetConnection ()
```

戻り値

この準備文に関連付けられている ULConnection オブジェクト。

1.6.7 GetParameterCount() メソッド

準備した文の入力パラメータの数を取得します。

構文

```
public virtual ul_u_short GetParameterCount ()
```

戻り値

準備した文の入力パラメータの数。

1.6.8 GetParameterID(const char *) メソッド

1 から始まるパラメータ名の順序数を取得します。

構文

```
public virtual ul_column_num GetParameterID (const char * name)
```

パラメータ

name ホスト変数名。

戻り値

1 から始まるパラメータ名の順序数。

1.6.9 GetParameterType(ul_column_num) メソッド

パラメータの記憶タイプまたはホスト変数の型を取得します。

構文

```
public virtual ul_column_storage_type GetParameterType (ul_column_num pid)
```

パラメータ

pid パラメータの、1 から始まる序数。

戻り値

指定したパラメータのタイプ。

1.6.10 GetPlan(char *, size_t) メソッド

クエリ実行プランのテキストベースの記述を取得します。

構文

```
public virtual size_t GetPlan (
    char * dst,
    size_t dstSize
)
```

パラメータ

dst プランテキストの宛先のバッファ。NULL を渡し、プランの保持に必要なバッファのサイズを特定します。
dstSize 宛先のバッファのサイズ。

戻り値

バッファにコピーされるバイト数。それ以外で、dst 値が NULL の場合は、プランの格納に必要なバイト数 (NULL ターミネータを含まない)。

備考

このメソッドは、主に開発中の使用を目的とします。

プランがない場合は、空の文字列を返します。準備された文が SQL クエリの場合には、プランが存在します。

関連するクエリの実行前にプランが取得された場合は、クエリの実行に使用される操作がプランに表示されます。また、クエリの実行後にプランが取得された場合は、各操作で生成されるロー数も表示されます。このプランを使用して、クエリの実行に関する理解を深めることができます。

1.6.11 GetResultSetSchema() メソッド

結果セットのスキーマを取得します。

構文

```
public virtual const ULResultSetSchema & GetResultSetSchema ()
```

戻り値

結果セットのスキーマに関する情報を取得するために使用できる ULResultSetSchema オブジェクト。

1.6.12 HasResultSet() メソッド

最後の文の影響を受けるローの数を取得します。

構文

```
public virtual ul_s_long GetRowsAffectedCount ()
```

戻り値

最後の文の影響を受けるローの数。ローの数を使用できない場合 (たとえば、文によってデータではなくスキーマが変更される場合)、戻り値は -1 になります。

1.6.13 HasResultSet() メソッド

SQL 文に結果セットがあるかどうかを調べます。

構文

```
public virtual bool HasResultSet ()
```

戻り値

この文が実行されたときに結果セットが生成される場合は true。それ以外で、結果セットが生成されない場合は false。

1.6.14 SetParameterBinary(ul_column_num, const p_ul_binary) メソッド

パラメータを ul_binary 値に設定します。

構文

```
public virtual bool SetParameterBinary (
    ul_column_num pid,
    const p_ul_binary value
)
```

パラメータ

pid パラメータの、1 から始まる序数。
value ul_binary 値。

戻り値

成功した場合は true、失敗した場合は false。

1.6.15 SetParameterBytes(ul_column_num pid, const ul_byte * value, size_t len)

パラメータを byte 配列値に設定します。

構文

```
public virtual size_t SetParameterBytes (
    ul_column_num cid,
    ul_byte * value,
    size_t len
)
```

パラメータ

cid パラメータの、1 から始まる序数。

value byte 配列値。

len 配列値のサイズを返します。

戻り値

成功した場合は true、失敗した場合は false。

1.6.16 SetParameterDateTime(ul_column_num, DECL_DATETIME *) メソッド

パラメータを DECL_DATETIME 値に設定します。

構文

```
public virtual bool SetParameterDateTime (
    ul_column_num pid,
    DECL_DATETIME * value
)
```

パラメータ

pid パラメータの、1 から始まる序数。

value DECL_DATETIME 値。

戻り値

成功した場合は true、失敗した場合は false。

1.6.17 SetParameterDouble(ul_column_num, ul_double) メソッド

パラメータを double 値に設定します。

構文

```
public virtual bool SetParameterDouble (
    ul_column_num pid,
    ul_double value
)
```

パラメータ

pid パラメータの、1 から始まる序数。

value double 値。

戻り値

成功した場合は true、失敗した場合は false。

1.6.18 SetParameterFloat(ul_column_num, ul_real) メソッド

パラメータを float 値に設定します。

構文

```
public virtual bool SetParameterFloat (
    ul_column_num pid,
    ul_real value
)
```

パラメータ

pid パラメータの、1 から始まる序数。
value float 値。

戻り値

成功した場合は true、失敗した場合は false。

1.6.19 SetParameterGuid(ul_column_num, GUID *) メソッド

パラメータを GUID 値に設定します。

構文

```
public virtual bool SetParameterGuid (
    ul_column_num pid,
    GUID * value
)
```

パラメータ

pid パラメータの、1 から始まる序数。
value GUID 値。

戻り値

成功した場合は true、失敗した場合は false。

1.6.20 SetParameterInt(ul_column_num, ul_s_long) メソッド

パラメータを整数値に設定します。

構文

```
public virtual bool SetParameterInt (
```

```
    ul_column_num pid,
    ul_s_long value
)
```

パラメータ

pid パラメータの、1 から始まる序数。
value 整数値。

戻り値

成功した場合は true、失敗した場合は false。

1.6.21 SetParameterIntWithType(ul_column_num, ul_s_big, ul_column_storage_type) メソッド

パラメータを、指定された整数型の整数値に設定します。

構文

```
public virtual bool SetParameterIntWithType (
    ul_column_num pid,
    ul_s_big value,
    ul_column_storage_type type
)
```

パラメータ

pid パラメータの、1 から始まる序数。
value 整数値。
type この値を処理するときの整数型。

戻り値

成功した場合は true、失敗した場合は false。

備考

次に、value パラメータに使用できる整数値のリストを示します。

- UL_TYPE_BIT
- UL_TYPE_TINY
- UL_TYPE_S_SHORT
- UL_TYPE_U_SHORT
- UL_TYPE_S_LONG
- UL_TYPE_U_LONG
- UL_TYPE_S_BIG
- UL_TYPE_U_BIG

1.6.22 SetParameterNull(ul_column_num) メソッド

パラメータを NULL に設定します。

構文

```
public virtual bool SetParameterNull (ul_column_num pid)
```

パラメータ

pid パラメータの、1 から始まる序数。

戻り値

成功した場合は true、失敗した場合は false。

1.6.23 SetParameterString(ul_column_num, const char *, size_t) メソッド

パラメータを文字列値に設定します。

構文

```
public virtual bool SetParameterString (
    ul_column_num pid,
```

```
    const char * value,
    size_t len
)
```

パラメータ

pid パラメータの、1 から始まる序数。

value 文字列値。

len 省略可能です。文字列の長さ (バイト単位)、または文字列が NULL で終了する場合は UL_NULL_TERMINATED_STRING に設定されます。このパラメータが 32K を超える場合は、SQLE_INVALID_PARAMETER に設定されます。サイズの大きい文字列の場合は、代わりに AppendParameterStringChunk メソッドが呼び出されます。

戻り値

成功した場合は true、失敗した場合は false。

関連情報

AppendParameterStringChunk(ul_column_num, const char *, size_t) メソッド [82 ページ]

1.7 ULResultSet クラス

Ultra Light データベースの結果セットを表します。

構文

```
public class ULResultSet
```

メンバー

ULResultSet のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public virtual bool	AfterLast() [97 ページ]	カーソルを最後のローの後に移動します。
public virtual bool	AppendByteChunk [98 ページ]	バイトをカラムに追加します。
public virtual bool	AppendStringChunk [100 ページ]	文字列のチャンクをカラムに追加します。
public virtual bool	BeforeFirst() [102 ページ]	カーソルを最初のローの前に移動します。
public virtual void	Close() [102 ページ]	このオブジェクトを破棄します。
public virtual bool	Delete() [103 ページ]	現在のローを削除し、次の有効なローに移動します。
public virtual bool	DeleteNamed(const char *) [103 ページ]	現在のローを削除し、次の有効なローに移動します。
public virtual bool	First() [104 ページ]	カーソルを最初のローに移動します。
public virtual bool	GetBinary [104 ページ]	カラムから値を ul_binary 値としてフェッチします。
public virtual size_t	GetBinaryLength [106 ページ]	カラムの値のバイナリ長さを取得します。
public virtual size_t	GetByteChunk [108 ページ]	カラムからバイナリチャンクを取得します。
public virtual ULConnection *	GetConnection() [110 ページ]	接続オブジェクトを取得します。
public virtual bool	GetDateTime [111 ページ]	カラムから値を DECL_DATETIME としてフェッチします。
public virtual ul_double	GetDouble [112 ページ]	カラムから値を double としてフェッチします。
public virtual ul_real	GetFloat [114 ページ]	カラムから値を float としてフェッチします。
public virtual bool	GetGuid [116 ページ]	カラムから値を GUID としてフェッチします。
public virtual ul_s_long	GetInt [117 ページ]	カラムから値を integer としてフェッチします。
public virtual ul_s_big	GetIntWithType [119 ページ]	カラムから値を、指定された整数型としてフェッチします。
public virtual const ULResultSetSchema &	GetResultSetSchema() [121 ページ]	結果セットに関する情報を取得するために使用できるオブジェクトを返します。
public virtual ul_u_long	GetRowCount(ul_u_long) [121 ページ]	テーブルのローの数を取得します。
public virtual UL_RS_STATE	GetState() [122 ページ]	カーソルの内部ステータスを取得します。
public virtual bool	GetString [123 ページ]	カラムから値を NULL で終了する文字列としてフェッチします。
public virtual size_t	GetStringChunk [125 ページ]	カラムから文字列チャンクを取得します。
public virtual size_t	GetStringLength [127 ページ]	カラムの値の文字列長さを取得します。
public virtual bool	IsNull [130 ページ]	カラムが NULL であるかどうかをチェックします。
public virtual bool	Last() [131 ページ]	カーソルを最後のローに移動します。
public virtual bool	Next() [131 ページ]	カーソルをロー 1 つ分進めます。

変更子とタイプ	メソッド	説明
public virtual bool	Previous() [132 ページ]	カーソルをロー 1 つ分戻します。
public virtual bool	Relative(ul_fetch_offset) [132 ページ]	カーソルを、現在のカーソルの位置から、offset で指定したロー数分移動します。
public virtual bool	SetBinary [133 ページ]	カラムを ul_binary 値に設定します。
public virtual bool	SetDateTime [138 ページ]	カラムを DECL_DATETIME 値に設定します。
public virtual bool	SetDefault [139 ページ]	カラムをそのデフォルト値に設定します。
public virtual bool	SetDouble [141 ページ]	カラムを double 値に設定します。
public virtual bool	SetFloat [143 ページ]	カラムを float 値に設定します。
public virtual bool	SetGuid [144 ページ]	カラムを GUID 値に設定します。
public virtual bool	SetInt [146 ページ]	カラムを整数値に設定します。
public virtual bool	SetIntWithType [135 ページ]	カラムを、指定された整数型の整数値に設定します。
public virtual bool	SetNull [148 ページ]	カラムを NULL に設定します。
public virtual bool	SetString [149 ページ]	カラムを文字列値に設定します。
public virtual bool	Update() [151 ページ]	現在の行を更新します。
public virtual bool	UpdateBegin() [152 ページ]	カラムの設定に使用される更新モードを選択します。

このセクションの内容:

AfterLast() メソッド [97 ページ]
カーソルを最後のローの後に移動します。

AppendByteChunk メソッド [98 ページ]
バイトをカラムに追加します。

AppendStringChunk メソッド [100 ページ]
文字列のチャンクをカラムに追加します。

BeforeFirst() メソッド [102 ページ]
カーソルを最初のローの前に移動します。

Close() メソッド [102 ページ]
このオブジェクトを破棄します。

Delete() メソッド [103 ページ]
現在のローを削除し、次の有効なローに移動します。

DeleteNamed(const char *) メソッド [103 ページ]
現在のローを削除し、次の有効なローに移動します。

First() メソッド [104 ページ]
カーソルを最初のローに移動します。

GetBinary メソッド [104 ページ]

カラムから値を ul_binary 値としてフェッチします。

GetBinaryLength メソッド [106 ページ]
カラムの値のバイナリ長さを取得します。

GetBytes(ul_column_num cid, ul_byte * dst, size_t len) メソッド [107 ページ]
カラムから値をバイト配列としてフェッチします。

GetByteChunk メソッド [108 ページ]
カラムからバイナリチャンクを取得します。

GetConnection() メソッド [110 ページ]
接続オブジェクトを取得します。

GetDateTime メソッド [111 ページ]
カラムから値を DECL_DATETIME としてフェッチします。

GetDouble メソッド [112 ページ]
カラムから値を double としてフェッチします。

GetFloat メソッド [114 ページ]
カラムから値を float としてフェッチします。

GetGuid メソッド [116 ページ]
カラムから値を GUID としてフェッチします。

GetInt メソッド [117 ページ]
カラムから値を integer としてフェッチします。

GetIntWithType メソッド [119 ページ]
カラムから値を、指定された整数型としてフェッチします。

GetResultSetSchema() メソッド [121 ページ]
結果セットに関する情報を取得するために使用できるオブジェクトを返します。

GetRowCount(ul_u_long) メソッド [121 ページ]
テーブルのローの数を取得します。

GetState() メソッド [122 ページ]
カーソルの内部ステータスを取得します。

GetString メソッド [123 ページ]
カラムから値を NULL で終了する文字列としてフェッチします。

GetStringChunk メソッド [125 ページ]
カラムから文字列チャンクを取得します。

GetStringLength メソッド [127 ページ]
カラムの値の文字列長さを取得します。

IsNull メソッド [130 ページ]
カラムが NULL であるかどうかをチェックします。

Last() メソッド [131 ページ]
カーソルを最後のローに移動します。

Next() メソッド [131 ページ]
カーソルをロー 1 つ分進めます。

Previous() メソッド [132 ページ]

カーソルをロー 1 つ分戻します。

Relative(ul_fetch_offset) メソッド [132 ページ]
カーソルを、現在のカーソルの位置から、offset で指定したロー数分移動します。

SetBinary メソッド [133 ページ]
カラムを ul_binary 値に設定します。

SetBytes(ul_column_num cid, const ul_byte * value, size_t len) [134 ページ]

SetIntWithType メソッド [135 ページ]
カラムを、指定された整数型の整数値に設定します。

SetDateTime メソッド [138 ページ]
カラムを DECL_DATETIME 値に設定します。

SetDefault メソッド [139 ページ]
カラムをそのデフォルト値に設定します。

SetDouble メソッド [141 ページ]
カラムを double 値に設定します。

SetFloat メソッド [143 ページ]
カラムを float 値に設定します。

SetGuid メソッド [144 ページ]
カラムを GUID 値に設定します。

SetInt メソッド [146 ページ]
カラムを整数値に設定します。

SetNull メソッド [148 ページ]
カラムを NULL に設定します。

SetString メソッド [149 ページ]
カラムを文字列値に設定します。

Update() メソッド [151 ページ]
現在の行を更新します。

UpdateBegin() メソッド [152 ページ]
カラムの設定に使用される更新モードを選択します。

1.7.1 AfterLast() メソッド

カーソルを最後のローの後に移動します。

構文

```
public virtual bool AfterLast ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.7.2 AppendByteChunk メソッド

バイトをカラムに追加します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	AppendByteChunk(const char *, const ul_byte *, size_t) [98 ページ]	バイトをカラムに追加します。
public virtual bool	AppendByteChunk(ul_column_num, const ul_byte *, size_t) [99 ページ]	バイトをカラムに追加します。

このセクションの内容:

AppendByteChunk(const char *, const ul_byte *, size_t) メソッド [98 ページ]
バイトをカラムに追加します。

AppendByteChunk(ul_column_num, const ul_byte *, size_t) メソッド [99 ページ]
バイトをカラムに追加します。

1.7.2.1 AppendByteChunk(const char *, const ul_byte *, size_t) メソッド

バイトをカラムに追加します。

構文

```
public virtual bool AppendByteChunk (
    const char * cname,
    const ul_byte * value,
    size_t valueSize
)
```

パラメータ

cname カラム名。
value 追加するバイトのチャンク。
valueSize バイトのチャンクのサイズ (バイト単位)。

戻り値

成功した場合は true、失敗した場合は false。

備考

AppendBinaryChunk メソッド呼び出しにより現時点で書き込み済みのカラムの末尾に、指定のバイトが追加されます。

関連情報

AppendByteChunk(ul_column_num, const ul_byte *, size_t) メソッド [99 ページ]

1.7.2.2 AppendByteChunk(ul_column_num, const ul_byte *, size_t) メソッド

バイトをカラムに追加します。

構文

```
public virtual bool AppendByteChunk (
    ul_column_num cid,
    const ul_byte * value,
    size_t valueSize
)
```

パラメータ

cid 1 から始まるカラムの順序数。
value 追加するバイトのチャンク。
valueSize バイトのチャンクのサイズ (バイト単位)。

戻り値

成功した場合は true、失敗した場合は false。

備考

AppendBinaryChunk メソッド呼び出しにより現時点で書き込み済みのカラムの末尾に、指定のバイトが追加されます。

1.7.3 AppendStringChunk メソッド

文字列のチャンクをカラムに追加します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	AppendStringChunk(const char *, const char *, size_t) [100 ページ]	文字列のチャンクをカラムに追加します。
public virtual bool	AppendStringChunk(ul_column_num, const char *, size_t) [101 ページ]	文字列のチャンクをカラムに追加します。

このセクションの内容:

AppendStringChunk(const char *, const char *, size_t) メソッド [100 ページ]
文字列のチャンクをカラムに追加します。

AppendStringChunk(ul_column_num, const char *, size_t) メソッド [101 ページ]
文字列のチャンクをカラムに追加します。

1.7.3.1 AppendStringChunk(const char *, const char *, size_t) メソッド

文字列のチャンクをカラムに追加します。

構文

```
public virtual bool AppendStringChunk (
    const char * cname,
    const char * value,
    size_t len
```

```
)
```

パラメータ

cname カラム名。
value 追加する文字列のチャンク。
len 省略可能です。文字列のチャンクの長さ (バイト単位)、または文字列が NULL で終了する場合は UL_NULL_TERMINATED_STRING 定数。

戻り値

成功した場合は true、失敗した場合は false。

備考

このメソッドは、AppendStringChunk メソッド呼び出しにより現時点で書き込み済みの文字列の末尾に、指定の文字列を追加します。

関連情報

AppendStringChunk(ul_column_num, const char *, size_t) メソッド [101 ページ]

1.7.3.2 AppendStringChunk(ul_column_num, const char *, size_t) メソッド

文字列のチャンクをカラムに追加します。

構文

```
public virtual bool AppendStringChunk (
    ul_column_num cid,
    const char * value,
    size_t len
)
```

パラメータ

cid 1 から始まるカラムの順序数。
value 追加する文字列のチャンク。
len 省略可能です。文字列のチャンクの長さ (バイト単位)、または文字列が NULL で終了する場合は UL_NULL_TERMINATED_STRING 定数。

戻り値

成功した場合は true、失敗した場合は false。

備考

このメソッドは、AppendStringChunk メソッド呼び出しにより現時点で書き込み済みの文字列の末尾に、指定の文字列を追加します。

1.7.4 BeforeFirst() メソッド

カーソルを最初のローの前に移動します。

構文

```
public virtual bool BeforeFirst ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.7.5 Close() メソッド

このオブジェクトを破棄します。

構文

```
public virtual void Close ()
```

1.7.6 Delete() メソッド

現在のローを削除し、次の有効なローに移動します。

構文

```
public virtual bool Delete ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.7.7 DeleteNamed(const char *) メソッド

現在のローを削除し、次の有効なローに移動します。

構文

```
public virtual bool DeleteNamed (const char * tableName)
```

パラメータ

tableName テーブル名またはその相関 (同じテーブル名を共有する複数のカラムがデータベースに存在する場合に必要)。

戻り値

成功した場合は true、失敗した場合は false。

1.7.8 First() メソッド

カーソルを最初のローに移動します。

構文

```
public virtual bool First ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.7.9 GetBinary メソッド

カラムから値を ul_binary 値としてフェッチします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	GetBinary(const char *, p_ul_binary, size_t) [105 ページ]	カラムから値を ul_binary 値としてフェッチします。
public virtual bool	GetBinary(ul_column_num, p_ul_binary, size_t) [105 ページ]	カラムから値を ul_binary 値としてフェッチします。

このセクションの内容:

GetBinary(const char *, p_ul_binary, size_t) メソッド [105 ページ]
カラムから値を ul_binary 値としてフェッチします。

GetBinary(ul_column_num, p_ul_binary, size_t) メソッド [105 ページ]
カラムから値を ul_binary 値としてフェッチします。

1.7.9.1 GetBinary(const char *, p_ul_binary, size_t) メソッド

カラムから値を ul_binary 値としてフェッチします。

構文

```
public virtual bool GetBinary (
    const char * cname,
    p_ul_binary dst,
    size_t len
)
```

パラメータ

cname カラム名。
dst ul_binary の結果。
len ul_binary オブジェクトのサイズ。

戻り値

値が正常にフェッチされた場合は true。

1.7.9.2 GetBinary(ul_column_num, p_ul_binary, size_t) メソッド

カラムから値を ul_binary 値としてフェッチします。

構文

```
public virtual bool GetBinary (
    ul_column_num cid,
    p_ul_binary dst,
    size_t len
)
```

パラメータ

cid 1 から始まるカラムの順序数。
dst ul_binary の結果。
len ul_binary オブジェクトのサイズ。

戻り値

値が正常にフェッチされた場合は true。

1.7.10 GetBinaryLength メソッド

カラムの値のバイナリ長さを取得します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual size_t	GetBinaryLength(const char *) [106 ページ]	カラムの値のバイナリ長さを取得します。
public virtual size_t	GetBinaryLength(ul_column_num) [107 ページ]	カラムの値のバイナリ長さを取得します。

このセクションの内容:

GetBinaryLength(const char *) メソッド [106 ページ]
カラムの値のバイナリ長さを取得します。

GetBinaryLength(ul_column_num) メソッド [107 ページ]
カラムの値のバイナリ長さを取得します。

1.7.10.1 GetBinaryLength(const char *) メソッド

カラムの値のバイナリ長さを取得します。

構文

```
public virtual size_t GetBinaryLength (const char * cname)
```

パラメータ

cname カラム名。

戻り値

binary としてのカラムの値のサイズ。

1.7.10.2 GetBinaryLength(ul_column_num) メソッド

カラムの値のバイナリ長さを取得します。

構文

```
public virtual size_t GetBinaryLength (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

binary としてのカラムの値のサイズ。

1.7.11 GetBytes(ul_column_num cid, ul_byte * dst, size_t len) メソッド

カラムから値をバイト配列としてフェッチします。

構文

```
public virtual size_t GetBytes (
    ul_column_num cid,
    ul_byte * dst,
    size_t len
)
```

パラメータ

cid 1 から始まるカラムの順序数。

dst バイトを保持するバッファ。
len バッファのサイズ (バイト単位)。

戻り値

宛先のバッファにコピーされたバイト数。値が NULL の場合は 0 を返します。

1.7.12 GetByteChunk メソッド

カラムからバイナリチャンクを取得します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual size_t	GetByteChunk(const char *, ul_byte *, size_t, size_t) [108 ページ]	カラムからバイナリチャンクを取得します。
public virtual size_t	GetByteChunk(ul_column_num, ul_byte *, size_t, size_t) [109 ページ]	カラムからバイナリチャンクを取得します。

このセクションの内容:

GetByteChunk(const char *, ul_byte *, size_t, size_t) メソッド [108 ページ]
カラムからバイナリチャンクを取得します。

GetByteChunk(ul_column_num, ul_byte *, size_t, size_t) メソッド [109 ページ]
カラムからバイナリチャンクを取得します。

1.7.12.1 GetByteChunk(const char *, ul_byte *, size_t, size_t) メソッド

カラムからバイナリチャンクを取得します。

構文

```
public virtual size_t GetByteChunk (
    const char * cname,
    ul_byte * dst,
    size_t len,
    size_t offset
```

```
)
```

パラメータ

cname カラム名。
dst バイトを保持するバッファ。
len バッファのサイズ (バイト単位)。
offset 値内でのオフセット (読み込み開始位置)、または前回の読み込みが終了したところから続行する場合は UL_BLOB_CONTINUE 定数。

戻り値

宛先のバッファにコピーされたバイト数。dst 値が NULL の場合は、残りのバイト数が返されます。カラムが NULL のときは、dst パラメータに空の文字列が返されます。IsNull メソッドを使用して、NULL と空の文字列を区別してください。

備考

0 が返された場合は、値の最後に到達しました。

関連情報

IsNull(ul_column_num) メソッド [131 ページ]

1.7.12.2 GetByteChunk(ul_column_num, ul_byte *, size_t, size_t) メソッド

カラムからバイナリチャンクを取得します。

構文

```
public virtual size_t GetByteChunk (
    ul_column_num cid,
    ul_byte * dst,
    size_t len,
    size_t offset
)
```

パラメータ

cid 1 から始まるカラムの順序数。
dst バイトを保持するバッファ。
len バッファのサイズ (バイト単位)。
offset 値内でのオフセット (読み込み開始位置)、または前回の読み込みが終了したところから続行する場合は UL_BLOB_CONTINUE 定数。

戻り値

宛先のバッファにコピーされたバイト数。dst 値が NULL の場合は、残りのバイト数が返されます。カラムが NULL のときは、dst パラメータに空の文字列が返されます。IsNull メソッドを使用して、NULL と空の文字列を区別してください。

備考

0 が返された場合は、値の最後に到達しました。

関連情報

IsNull(ul_column_num) メソッド [131 ページ]

1.7.13 GetConnection() メソッド

接続オブジェクトを取得します。

構文

```
public virtual ULConnection * GetConnection ()
```

戻り値

この結果セットに関連付けられている ULConnection オブジェクト。

1.7.14 GetDateTime メソッド

カラムから値を DECL_DATETIME としてフェッチします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	GetDateTime(const char *, DECL_DATETIME *) [111 ページ]	カラムから値を DECL_DATETIME としてフェッチします。
public virtual bool	GetDateTime(ul_column_num, DECL_DATETIME *) [112 ページ]	カラムから値を DECL_DATETIME としてフェッチします。

このセクションの内容:

GetDateTime(const char *, DECL_DATETIME *) メソッド [111 ページ]
カラムから値を DECL_DATETIME としてフェッチします。

GetDateTime(ul_column_num, DECL_DATETIME *) メソッド [112 ページ]
カラムから値を DECL_DATETIME としてフェッチします。

1.7.14.1 GetDateTime(const char *, DECL_DATETIME *) メソッド

カラムから値を DECL_DATETIME としてフェッチします。

構文

```
public virtual bool GetDateTime (
    const char * cname,
    DECL_DATETIME * dst
)
```

パラメータ

cname カラム名。
dst DECL_DATETIME 値。

戻り値

値が正常にフェッチされた場合は true。

1.7.14.2 GetDateTime(ul_column_num, DECL_DATETIME *) メソッド

カラムから値を DECL_DATETIME としてフェッチします。

構文

```
public virtual bool GetDateTime (
    ul_column_num cid,
    DECL_DATETIME * dst
)
```

パラメータ

cid 1 から始まるカラムの順序数。
dst DECL_DATETIME 値。

戻り値

値が正常にフェッチされた場合は true。

1.7.15 GetDouble メソッド

カラムから値を double としてフェッチします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual ul_double	GetDouble(const char *) [113 ページ]	カラムから値を double としてフェッチします。

変更子とタイプ	オーバーロード名	説明
public virtual ul_double	GetDouble(ul_column_num) [113 ページ]	カラムから値を double としてフェッチします。

このセクションの内容:

1.7.15.1 GetDouble(const char *) メソッド

カラムから値を double としてフェッチします。

構文

```
public virtual ul_double GetDouble (const char * cname)
```

パラメータ

cname カラム名。

戻り値

double としてのカラム値。

1.7.15.2 GetDouble(ul_column_num) メソッド

カラムから値を double としてフェッチします。

構文

```
public virtual ul_double GetDouble (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

double としてのカラム値。

1.7.16 GetFloat メソッド

カラムから値を float としてフェッチします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual ul_real	GetFloat(const char *) [114 ページ]	カラムから値を float としてフェッチします。
public virtual ul_real	GetFloat(ul_column_num) [115 ページ]	カラムから値を float としてフェッチします。

このセクションの内容:

GetFloat(const char *) メソッド [114 ページ]
カラムから値を float としてフェッチします。

GetFloat(ul_column_num) メソッド [115 ページ]
カラムから値を float としてフェッチします。

1.7.16.1 GetFloat(const char *) メソッド

カラムから値を float としてフェッチします。

構文

```
public virtual ul_real GetFloat (const char * cname)
```

パラメータ

cname カラム名。

戻り値

float としてのカラム値。

1.7.16.2 GetFloat(ul_column_num) メソッド

カラムから値を float としてフェッチします。

構文

```
public virtual ul_real GetFloat (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

float としてのカラム値。

1.7.17 GetGuid メソッド

カラムから値を GUID としてフェッチします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	GetGuid(const char *, GUID *) [116 ページ]	カラムから値を float としてフェッチします。
public virtual bool	GetGuid(ul_column_num, GUID *) [117 ページ]	カラムから値を GUID としてフェッチします。

このセクションの内容:

1.7.17.1 GetGuid(const char *, GUID *) メソッド

カラムから値を GUID としてフェッチします。

構文

```
public virtual bool GetGuid (
    const char * cname,
    GUID * dst
)
```

パラメータ

cname カラム名。
dst GUID 値。

戻り値

値が正常にフェッチされた場合は true。

1.7.17.2 GetGuid(ul_column_num, GUID *) メソッド

カラムから値を GUID としてフェッチします。

構文

```
public virtual bool GetGuid (
    ul_column_num cid,
    GUID * dst
)
```

パラメータ

cid 1 から始まるカラムの順序数。
dst GUID 値。

戻り値

値が正常にフェッチされた場合は true。

1.7.18 GetInt メソッド

カラムから値を integer としてフェッチします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual ul_s_long	GetInt(const char *) [118 ページ]	カラムから値を integer としてフェッチします。

変更子とタイプ	オーバーロード名	説明
public virtual ul_s_long	GetInt(ul_column_num) [118 ページ]	カラムから値を integer としてフェッチします。

このセクションの内容:

GetInt(const char *) メソッド [118 ページ]
カラムから値を integer としてフェッチします。

GetInt(ul_column_num) メソッド [118 ページ]
カラムから値を integer としてフェッチします。

1.7.18.1 GetInt(const char *) メソッド

カラムから値を integer としてフェッチします。

構文

```
public virtual ul_s_long GetInt (const char * cname)
```

パラメータ

cname カラム名。

戻り値

整数としてのカラム値。

1.7.18.2 GetInt(ul_column_num) メソッド

カラムから値を integer としてフェッチします。

構文

```
public virtual ul_s_long GetInt (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

整数としてのカラム値。

1.7.19 GetIntWithType メソッド

カラムから値を、指定された整数型としてフェッチします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual ul_s_big	GetIntWithType(const char *, ul_column_storage_type) [119 ページ]	カラムから値を、指定された整数型としてフェッチします。
public virtual ul_s_big	GetIntWithType(ul_column_num, ul_column_storage_type) [120 ページ]	カラムから値を、指定された整数型としてフェッチします。

このセクションの内容:

GetIntWithType(const char *, ul_column_storage_type) メソッド [119 ページ]
カラムから値を、指定された整数型としてフェッチします。

GetIntWithType(ul_column_num, ul_column_storage_type) メソッド [120 ページ]
カラムから値を、指定された整数型としてフェッチします。

1.7.19.1 GetIntWithType(const char *, ul_column_storage_type) メソッド

カラムから値を、指定された整数型としてフェッチします。

構文

```
public virtual ul_s_big GetIntWithType (
    const char * cname,
    ul_column_storage_type type
)
```

パラメータ

cname カラム名。
type フェッチするときの整数型。

戻り値

整数としてのカラム値。

備考

次に、type パラメータに使用できる整数値のリストを示します。

- UL_TYPE_BIT
- UL_TYPE_TINY
- UL_TYPE_S_SHORT
- UL_TYPE_U_SHORT
- UL_TYPE_S_LONG
- UL_TYPE_U_LONG
- UL_TYPE_S_BIG
- UL_TYPE_U_BIG

1.7.19.2 GetIntWithType(ul_column_num, ul_column_storage_type) メソッド

カラムから値を、指定された整数型としてフェッチします。

構文

```
public virtual ul_s_big GetIntWithType (
    ul_column_num cid,
    ul_column_storage_type type
)
```

パラメータ

cid 1 から始まるカラムの順序数。
type フェッチするときの整数型。

戻り値

整数としてのカラム値。

備考

次に、type パラメータに使用できる整数値のリストを示します。

- UL_TYPE_BIT
- UL_TYPE_TINY
- UL_TYPE_S_SHORT
- UL_TYPE_U_SHORT
- UL_TYPE_S_LONG
- UL_TYPE_U_LONG
- UL_TYPE_S_BIG
- UL_TYPE_U_BIG

1.7.20 GetResultSetSchema() メソッド

結果セットに関する情報を取得するために使用できるオブジェクトを返します。

構文

```
public virtual const ULResultSetSchema & GetResultSetSchema ()
```

戻り値

結果セットに関する情報を取得するために使用できる ULResultSetSchema オブジェクト。

1.7.21 GetRowCount(ul_u_long) メソッド

テーブルのローの数を取得します。

構文

```
public virtual ul_u_long GetRowCount (ul_u_long threshold)
```

パラメータ

threshold カウントするローの数の制限。無限を表すには 0 を設定します。

戻り値

テーブル内のローの数。

備考

このメソッドは、"SELECT COUNT(*) FROM table" 文を実行するのと同じです。

1.7.22 GetState() メソッド

カーソルの内部ステータスを取得します。

構文

```
public virtual UL_RS_STATE GetState ()
```

戻り値

カーソルのステータス

1.7.23 GetString メソッド

カラムから値を NULL で終了する文字列としてフェッチします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	GetString(const char *, char *, size_t) [123 ページ]	カラムから値を NULL で終了する文字列としてフェッチします。
public virtual bool	GetString(ul_column_num, char *, size_t) [124 ページ]	カラムから値を NULL で終了する文字列としてフェッチします。

このセクションの内容:

GetString(const char *, char *, size_t) メソッド [123 ページ]
カラムから値を NULL で終了する文字列としてフェッチします。

GetString(ul_column_num, char *, size_t) メソッド [124 ページ]
カラムから値を NULL で終了する文字列としてフェッチします。

1.7.23.1 GetString(const char *, char *, size_t) メソッド

カラムから値を NULL で終了する文字列としてフェッチします。

構文

```
public virtual bool GetString (
    const char * cname,
    char * dst,
    size_t len
)
```

パラメータ

cname カラム名。
dst 文字列値を保持するバッファ。文字列は、トランケートされても NULL で終了します。
len バッファのサイズ (バイト単位)。

戻り値

値が正常にフェッチされた場合は true。

備考

値全体を保持できるほど大きくない場合、文字列はバッファ内でトランケートされます。

1.7.23.2 GetString(ul_column_num, char *, size_t) メソッド

カラムから値を NULL で終了する文字列としてフェッチします。

構文

```
public virtual bool GetString (
    ul_column_num cid,
    char * dst,
    size_t len
)
```

パラメータ

cid 1 から始まるカラムの順序数。
dst 文字列値を保持するバッファ。文字列は、トランケートされても NULL で終了します。
len バッファのサイズ (バイト単位)。

戻り値

値が正常にフェッチされた場合は true。

備考

値全体を保持できるほど大きくない場合、文字列はバッファ内でトランケートされます。

1.7.24 GetStringChunk メソッド

カラムから文字列チャンクを取得します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual size_t	GetStringChunk(const char *, char *, size_t, size_t) [125 ページ]	カラムから文字列チャンクを取得します。
public virtual size_t	GetStringChunk(ul_column_num, char *, size_t, size_t) [126 ページ]	カラムから文字列チャンクを取得します。

このセクションの内容:

GetStringChunk(const char *, char *, size_t, size_t) メソッド [125 ページ]
カラムから文字列チャンクを取得します。

GetStringChunk(ul_column_num, char *, size_t, size_t) メソッド [126 ページ]
カラムから文字列チャンクを取得します。

1.7.24.1 GetStringChunk(const char *, char *, size_t, size_t) メソッド

カラムから文字列チャンクを取得します。

構文

```
public virtual size_t GetStringChunk (
    const char * cname,
    char * dst,
    size_t len,
    size_t offset
)
```

パラメータ

cname カラム名。
dst 文字列のチャンクを保持するバッファ。文字列は、トランケートされても NULL で終了します。
len バッファのサイズ (バイト単位)。
offset 値内でのオフセット (読み込み開始位置)、または前回の読み込みが終了したところから続行する場合は UL_BLOB_CONTINUE 定数。

戻り値

宛先のバッファにコピーされたバイト数 (NULL ターミネータを含まない)。dst 値を NULL に設定した場合は、文字列の残りのバイト数が返されます。カラムが NULL のときは、dst パラメータに空の文字列が返されます。IsNull メソッドを使用して、NULL と空の文字列を区別してください。

備考

0 が返された場合は、値の最後に到達しました。

関連情報

IsNull(ul_column_num) メソッド [131 ページ]

1.7.24.2 GetStringChunk(ul_column_num, char *, size_t, size_t) メソッド

カラムから文字列チャンクを取得します。

構文

```
public virtual size_t GetStringChunk (
    ul_column_num cid,
    char * dst,
    size_t len,
    size_t offset
)
```

パラメータ

cid 1 から始まるカラムの順序数。

dst 文字列のチャンクを保持するバッファ。文字列は、トランケートされても NULL で終了します。

len バッファのサイズ (バイト単位)。

offset 値内でのオフセット (読み込み開始位置)、または前回の読み込みが終了したところから続行する場合は UL_BLOB_CONTINUE 定数に設定します。

戻り値

宛先のバッファにコピーされたバイト数 (NULL ターミネータを含まない)。dst 値を NULL に設定した場合は、文字列の残りのバイト数が返されます。カラムが NULL のときは、dst パラメータに空の文字列が返されます。IsNull メソッドを使用して、NULL と空の文字列を区別してください。

備考

0 が返された場合は、値の最後に到達しました。

関連情報

IsNull(ul_column_num) メソッド [131 ページ]

1.7.25 GetStringLength メソッド

カラムの値の文字列長さを取得します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual size_t	GetStringLength(const char *) [128 ページ]	カラムの値の文字列長さを取得します。
public virtual size_t	GetStringLength(ul_column_num) [129 ページ]	カラムの値の文字列長さを取得します。

このセクションの内容:

GetStringLength(const char *) メソッド [128 ページ]
カラムの値の文字列長さを取得します。

GetStringLength(ul_column_num) メソッド [129 ページ]
カラムの値の文字列長さを取得します。

1.7.25.1 GetStringLength(const char *) メソッド

カラムの値の文字列長さを取得します。

構文

```
public virtual size_t GetStringLength (const char * cname)
```

パラメータ

cname カラム名。

戻り値

いずれかの GetString メソッドによって返される文字列を保持するために必要なバイト数または文字数 (NULL ターミネータを含まない)。

備考

次の例は、カラムの文字列長の取得方法を示します。

```
len = result_set->GetStringLength( cid );
dst = new char[ len + 1 ];
result_set->GetString( cid, dst, len + 1 );
```

ワイド文字の場合の使用方法を次に示します。

```
len = result_set->GetStringLength( cid );
dst = new ul_wchar[ len + 1 ];
result_set->GetString( cid, dst, len + 1 );
```

関連情報

GetString(ul_column_num, char *, size_t) メソッド [124 ページ]

1.7.25.2 GetStringLength(ul_column_num) メソッド

カラムの値の文字列長さを取得します。

構文

```
public virtual size_t GetStringLength (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

いずれかの GetString メソッドによって返される文字列を保持するために必要なバイト数または文字数 (NULL ターミネータを含まない)。

備考

次の例は、カラムの文字列長の取得方法を示します。

```
len = result_set->GetStringLength( cid );
dst = new char[ len + 1 ];
result_set->GetString( cid, dst, len + 1 );
```

ワイド文字の場合の使用方法を次に示します。

```
len = result_set->GetStringLength( cid );
dst = new ul_wchar[ len + 1 ];
result_set->GetString( cid, dst, len + 1 );
```

関連情報

GetString(ul_column_num, char *, size_t) メソッド [124 ページ]

1.7.26 IsNull メソッド

カラムが NULL であるかどうかをチェックします。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	IsNull(const char *) [130 ページ]	カラムが NULL であるかどうかをチェックします。
public virtual bool	IsNull(ul_column_num) [131 ページ]	カラムが NULL であるかどうかをチェックします。

このセクションの内容:

1.7.26.1 IsNull(const char *) メソッド

カラムが NULL であるかどうかをチェックします。

構文

```
public virtual bool IsNull (const char * cname)
```

パラメータ

cname カラム名。

戻り値

カラムの値が NULL の場合は true。

1.7.26.2 IsNull(ul_column_num) メソッド

カラムが NULL であるかどうかをチェックします。

構文

```
public virtual bool IsNull (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムの値が NULL の場合は true。

1.7.27 Last() メソッド

カーソルを最後のローに移動します。

構文

```
public virtual bool Last ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.7.28 Next() メソッド

カーソルをロー 1 つ分進めます。

構文

```
public virtual bool Next ()
```

戻り値

カーソルが正常に進められる場合は、true。true が返されても、カーソルが次のローに正常に移動したときに、エラーが送信されることがあります。たとえば SELECT 式の評価中に変換エラーが発生する可能性があります。この場合、カラム値を取得するときにもエラーが返されます。カーソルを進められなかった場合は、false が返されます。たとえば、次のローが存在しない可能性があります。この場合、移動後のカーソル位置は最後のローの後ろに設定されます。

1.7.29 Previous() メソッド

カーソルをロー 1 つ分戻します。

構文

```
public virtual bool Previous ()
```

戻り値

カーソルをロー 1 つ分戻せた場合は、true。カーソルを戻せなかった場合は、false。移動後のカーソル位置は、最初のローの前に設定されます。

1.7.30 Relative(ul_fetch_offset) メソッド

カーソルを、現在のカーソルの位置から、offset で指定したロー数分移動します。

構文

```
public virtual bool Relative (ul_fetch_offset offset)
```

パラメータ

offset 移動するローの数。

戻り値

成功した場合は true、失敗した場合は false。

1.7.31 SetBinary メソッド

カラムを ul_binary 値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetBinary(const char *, p_ul_binary) [133 ページ]	カラムを ul_binary 値に設定します。
public virtual bool	SetBinary(ul_column_num, p_ul_binary) [134 ページ]	カラムを ul_binary 値に設定します。

このセクションの内容:

SetBinary(const char *, p_ul_binary) メソッド [133 ページ]
カラムを ul_binary 値に設定します。

SetBinary(ul_column_num, p_ul_binary) メソッド [134 ページ]
カラムを ul_binary 値に設定します。

1.7.31.1 SetBinary(const char *, p_ul_binary) メソッド

カラムを ul_binary 値に設定します。

構文

```
public virtual bool SetBinary (
    const char * cname,
    p_ul_binary value
)
```

パラメータ

cname カラム名。
value ul_binary 値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。

戻り値

成功した場合は true、失敗した場合は false。

1.7.31.2 SetBinary(ul_column_num, p_ul_binary) メソッド

カラムを ul_binary 値に設定します。

構文

```
public virtual bool SetBinary (
    ul_column_num cid,
    p_ul_binary value
)
```

パラメータ

cid 1 から始まるカラムの順序数。

value ul_binary 値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。

戻り値

成功した場合は true、失敗した場合は false。

1.7.32 SetBytes(ul_column_num cid, const ul_byte * value, size_t len)

構文

```
public virtual size_t SetBytes (
    ul_column_num cid,
    ul_byte * dst,
    size_t len
)
```

パラメータ

cid 1 から始まるカラムの順序数。
dst byte 配列値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。
len 配列値のサイズを返します。

戻り値

成功した場合は true、失敗した場合は false。

1.7.33 SetIntWithType メソッド

カラムを、指定された整数型の整数値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetIntWithType(const char *, ul_s_big, ul_column_storage_type) [136 ページ]	カラムを、指定された整数型の整数値に設定します。
public virtual bool	SetIntWithType(ul_column_num, ul_s_big, ul_column_storage_type) [137 ページ]	カラムを、指定された整数型の整数値に設定します。

このセクションの内容:

SetIntWithType(const char *, ul_s_big, ul_column_storage_type) メソッド [136 ページ]
カラムを、指定された整数型の整数値に設定します。

SetIntWithType(ul_column_num, ul_s_big, ul_column_storage_type) メソッド [137 ページ]
カラムを、指定された整数型の整数値に設定します。

1.7.33.1 SetIntWithType(const char *, ul_s_big, ul_column_storage_type) メソッド

カラムを、指定された整数型の整数値に設定します。

構文

```
public virtual bool SetIntWithType (
    const char * cname,
    ul_s_big value,
    ul_column_storage_type type
)
```

パラメータ

cname カラム名。
value 整数値。
type この値を処理するときの整数型。

戻り値

成功した場合は true、失敗した場合は false。

備考

次に、value パラメータに使用できる整数値のリストを示します。

- UL_TYPE_BIT
- UL_TYPE_TINY
- UL_TYPE_S_SHORT
- UL_TYPE_U_SHORT
- UL_TYPE_S_LONG
- UL_TYPE_U_LONG
- UL_TYPE_S_BIG
- UL_TYPE_U_BIG

1.7.33.2 SetIntWithType(ul_column_num, ul_s_big, ul_column_storage_type) メソッド

カラムを、指定された整数型の整数値に設定します。

構文

```
public virtual bool SetIntWithType (
    ul_column_num cid,
    ul_s_big value,
    ul_column_storage_type type
)
```

パラメータ

cid 1 から始まるカラムの順序数。

value 整数値。

type この値を処理するときの整数型。

戻り値

成功した場合は true、失敗した場合は false。

備考

次に、value パラメータに使用できる整数値のリストを示します。

- UL_TYPE_BIT
- UL_TYPE_TINY
- UL_TYPE_S_SHORT
- UL_TYPE_U_SHORT
- UL_TYPE_S_LONG
- UL_TYPE_U_LONG
- UL_TYPE_S_BIG
- UL_TYPE_U_BIG

1.7.34 SetDateTime メソッド

カラムを DECL_DATETIME 値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetDateTime(const char *, DECL_DATETIME *) [138 ページ]	カラムを DECL_DATETIME 値に設定します。
public virtual bool	SetDateTime(ul_column_num, DECL_DATETIME *) [139 ページ]	カラムを DECL_DATETIME 値に設定します。

このセクションの内容:

SetDateTime(const char *, DECL_DATETIME *) メソッド [138 ページ]
カラムを DECL_DATETIME 値に設定します。

SetDateTime(ul_column_num, DECL_DATETIME *) メソッド [139 ページ]
カラムを DECL_DATETIME 値に設定します。

1.7.34.1 SetDateTime(const char *, DECL_DATETIME *) メソッド

カラムを DECL_DATETIME 値に設定します。

構文

```
public virtual bool SetDateTime (
    const char * cname,
    DECL_DATETIME * value
)
```

パラメータ

cname カラム名。
value DECL_DATETIME 値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。

戻り値

成功した場合は true、失敗した場合は false。

1.7.34.2 SetDateTime(ul_column_num, DECL_DATETIME *) メソッド

カラムを DECL_DATETIME 値に設定します。

構文

```
public virtual bool SetDateTime (
    ul_column_num cid,
    DECL_DATETIME * value
)
```

パラメータ

cid 1 から始まるカラムの順序数。

value DECL_DATETIME 値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。

戻り値

成功した場合は true、失敗した場合は false。

1.7.35 SetDefault メソッド

カラムをそのデフォルト値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetDefault(const char *) [140 ページ]	カラムをそのデフォルト値に設定します。

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetDefault(ul_column_num) [140 ページ]	カラムをそのデフォルト値に設定します。

このセクションの内容:

1.7.35.1 SetDefault(const char *) メソッド

カラムをそのデフォルト値に設定します。

構文

```
public virtual bool SetDefault (const char * cname)
```

パラメータ

cname カラム名。

戻り値

成功した場合は true、失敗した場合は false。

1.7.35.2 SetDefault(ul_column_num) メソッド

カラムをそのデフォルト値に設定します。

構文

```
public virtual bool SetDefault (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

成功した場合は true、失敗した場合は false。

1.7.36 SetDouble メソッド

カラムを double 値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetDouble(const char *, ul_double) [141 ページ]	カラムを double 値に設定します。
public virtual bool	SetDouble(ul_column_num, ul_double) [142 ページ]	カラムを double 値に設定します。

このセクションの内容:

SetDouble(const char *, ul_double) メソッド [141 ページ]
カラムを double 値に設定します。

SetDouble(ul_column_num, ul_double) メソッド [142 ページ]
カラムを double 値に設定します。

1.7.36.1 SetDouble(const char *, ul_double) メソッド

カラムを double 値に設定します。

構文

```
public virtual bool SetDouble (
    const char * cname,
    ul_double value
)
```

パラメータ

cname カラム名。
value double 値。

戻り値

成功した場合は true、失敗した場合は false。

1.7.36.2 SetDouble(ul_column_num, ul_double) メソッド

カラムを double 値に設定します。

構文

```
public virtual bool SetDouble (
    ul_column_num cid,
    ul_double value
)
```

パラメータ

cid 1 から始まるカラムの順序数。
value double 値。

戻り値

成功した場合は true、失敗した場合は false。

1.7.37 SetFloat メソッド

カラムを float 値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetFloat(const char *, ul_real) [143 ページ]	カラムを float 値に設定します。
public virtual bool	SetFloat(ul_column_num, ul_real) [144 ページ]	カラムを float 値に設定します。

このセクションの内容:

SetFloat(const char *, ul_real) メソッド [143 ページ]
カラムを float 値に設定します。

SetFloat(ul_column_num, ul_real) メソッド [144 ページ]
カラムを float 値に設定します。

1.7.37.1 SetFloat(const char *, ul_real) メソッド

カラムを float 値に設定します。

構文

```
public virtual bool SetFloat (
    const char * cname,
    ul_real value
)
```

パラメータ

cname カラム名。
value float 値。

戻り値

成功した場合は true、失敗した場合は false。

1.7.37.2 SetFloat(ul_column_num, ul_real) メソッド

カラムを float 値に設定します。

構文

```
public virtual bool SetFloat (
    ul_column_num cid,
    ul_real value
)
```

パラメータ

cid 1 から始まるカラムの順序数。
value float 値。

戻り値

成功した場合は true、失敗した場合は false。

1.7.38 SetGuid メソッド

カラムを GUID 値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetGuid(const char *, GUID *) [145 ページ]	カラムを GUID 値に設定します。

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetGuid(ul_column_num, GUID *) [145 ページ]	カラムを GUID 値に設定します。

このセクションの内容:

1.7.38.1 SetGuid(const char *, GUID *) メソッド

カラムを GUID 値に設定します。

構文

```
public virtual bool SetGuid (
    const char * cname,
    GUID * value
)
```

パラメータ

cname カラム名。
value GUID 値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。

戻り値

成功した場合は true、失敗した場合は false。

1.7.38.2 SetGuid(ul_column_num, GUID *) メソッド

カラムを GUID 値に設定します。

構文

```
public virtual bool SetGuid (
```

```
    ul_column_num cid,
    GUID * value
)
```

パラメータ

cid 1 から始まるカラムの順序数。

value GUID 値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。

戻り値

成功した場合は true、失敗した場合は false。

1.7.39 SetInt メソッド

カラムを整数値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetInt(const char *, ul_s_long) [147 ページ]	カラムを整数値に設定します。
public virtual bool	SetInt(ul_column_num, ul_s_long) [147 ページ]	カラムを整数値に設定します。

このセクションの内容:

SetInt(const char *, ul_s_long) メソッド [147 ページ]
カラムを整数値に設定します。

SetInt(ul_column_num, ul_s_long) メソッド [147 ページ]
カラムを整数値に設定します。

1.7.39.1 SetInt(const char *, ul_s_long) メソッド

カラムを整数値に設定します。

構文

```
public virtual bool SetInt (
    const char * cname,
    ul_s_long value
)
```

パラメータ

cname カラム名。
value 符号付き整数値。

戻り値

成功した場合は true、失敗した場合は false。

1.7.39.2 SetInt(ul_column_num, ul_s_long) メソッド

カラムを整数値に設定します。

構文

```
public virtual bool SetInt (
    ul_column_num cid,
    ul_s_long value
)
```

パラメータ

cid 1 から始まるカラムの順序数。
value 符号付き整数値。

戻り値

成功した場合は true、失敗した場合は false。

1.7.40 SetNull メソッド

カラムを NULL に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetNull(const char *) [148 ページ]	カラムを NULL に設定します。
public virtual bool	SetNull(ul_column_num) [149 ページ]	カラムを NULL に設定します。

このセクションの内容:

SetNull(const char *) メソッド [148 ページ]
カラムを NULL に設定します。

SetNull(ul_column_num) メソッド [149 ページ]
カラムを NULL に設定します。

1.7.40.1 SetNull(const char *) メソッド

カラムを NULL に設定します。

構文

```
public virtual bool SetNull (const char * cname)
```

パラメータ

cname カラム名。

戻り値

成功した場合は true、失敗した場合は false。

1.7.40.2 SetNull(ul_column_num) メソッド

カラムを NULL に設定します。

構文

```
public virtual bool SetNull (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

成功した場合は true、失敗した場合は false。

1.7.41 SetString メソッド

カラムを文字列値に設定します。

オーバロードリスト

変更子とタイプ	オーバーロード名	説明
public virtual bool	SetString(const char *, const char *, size_t) [150 ページ]	カラムを文字列値に設定します。
public virtual bool	SetString(ul_column_num, const char *, size_t) [151 ページ]	カラムを文字列値に設定します。

このセクションの内容:

SetString(const char *, const char *, size_t) メソッド [150 ページ]

カラムを文字列値に設定します。

SetString(ul_column_num, const char *, size_t) メソッド [151 ページ]
カラムを文字列値に設定します。

1.7.41.1 SetString(const char *, const char *, size_t) メソッド

カラムを文字列値に設定します。

構文

```
public virtual bool SetString (
    const char * cname,
    const char * value,
    size_t len
)
```

パラメータ

cname カラム名。
value 文字列値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。
len 省略可能です。文字列の長さ (バイト単位)、または文字列が NULL で終了する場合は UL_NULL_TERMINATED_STRING 定数。設定された len 値が 32K より大きい場合は、SQLE_INVALID_PARAMETER 定数が設定されます。サイズの大きい文字列の場合は、代わりに AppendStringChunk メソッドが呼び出されます。

戻り値

成功した場合は true、失敗した場合は false。

関連情報

AppendStringChunk(ul_column_num, const char *, size_t) メソッド [101 ページ]

1.7.41.2 SetString(ul_column_num, const char *, size_t) メソッド

カラムを文字列値に設定します。

構文

```
public virtual bool SetString (
    ul_column_num cid,
    const char * value,
    size_t len
)
```

パラメータ

cid 1 から始まるカラムの順序数。

value 文字列値。NULL を渡すことは、SetNull メソッドを呼び出すことと同じです。

len 省略可能です。文字列の長さ (バイト単位)、または文字列が NULL で終了する場合は UL_NULL_TERMINATED_STRING 定数。設定された len 値が 32K より大きい場合は、SQLE_INVALID_PARAMETER 定数が設定されます。サイズの大きい文字列の場合は、代わりに AppendStringChunk メソッドが呼び出されます。

戻り値

成功した場合は true、失敗した場合は false。

関連情報

AppendStringChunk(ul_column_num, const char *, size_t) メソッド [101 ページ]

1.7.42 Update() メソッド

現在の行を更新します。

構文

```
public virtual bool Update ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.7.43 UpdateBegin() メソッド

カラムの設定に使用される更新モードを選択します。

構文

```
public virtual bool UpdateBegin ()
```

戻り値

成功した場合は true、失敗した場合は false。

備考

更新モードの場合、プライマリキー内のカラムの修正はできません。

1.8 ULResultSetSchema クラス

Ultra Light の結果セットのスキーマを表します。

構文

```
public class ULResultSetSchema
```

メンバー

ULResultSetSchema のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public virtual ul_column_num	GetColumnCount() [154 ページ]	結果セットまたはテーブル内のカラム数を取得します。
public virtual ul_column_num	GetColumnID(const char *) [154 ページ]	1 から始まるカラム ID を名前から取得します。
public virtual const char *	GetColumnName(ul_column_num, ul_column_name_type) [155 ページ]	1 から始まる ID を指定してカラムの名前を取得します。
public virtual size_t	GetColumnPrecision(ul_column_num) [156 ページ]	数値カラムの精度を取得します。
public virtual size_t	GetColumnScale(ul_column_num) [156 ページ]	数値カラムの位取りを取得します。
public virtual size_t	GetColumnSize(ul_column_num) [157 ページ]	カラムのサイズを取得します。
public virtual ul_column_sql_type	GetColumnSQLType(ul_column_num) [157 ページ]	カラムの SQL の型を取得します。
public virtual ul_column_storage_type	GetColumnType(ul_column_num) [158 ページ]	カラムの記憶タイプまたはホスト変数の型を取得します。
public virtual ULConnection *	GetConnection() [158 ページ]	ULConnection オブジェクトを取得します。
public virtual bool	IsAliased(ul_column_num) [158 ページ]	結果セット内のカラムにエイリアスが付与されているかどうかを示します。

このセクションの内容:

GetColumnCount() メソッド [154 ページ]
結果セットまたはテーブル内のカラム数を取得します。

GetColumnID(const char *) メソッド [154 ページ]
1 から始まるカラム ID を名前から取得します。

GetColumnName(ul_column_num, ul_column_name_type) メソッド [155 ページ]
1 から始まる ID を指定してカラムの名前を取得します。

GetColumnPrecision(ul_column_num) メソッド [156 ページ]
数値カラムの精度を取得します。

GetColumnScale(ul_column_num) メソッド [156 ページ]
数値カラムの位取りを取得します。

GetColumnSize(ul_column_num) メソッド [157 ページ]
カラムのサイズを取得します。

GetColumnSQLType(ul_column_num) メソッド [157 ページ]
カラムの SQL の型を取得します。

GetColumnType(ul_column_num) メソッド [158 ページ]
カラムの記憶タイプまたはホスト変数の型を取得します。

GetConnection() メソッド [158 ページ]
ULConnection オブジェクトを取得します。

IsAliased(ul_column_num) メソッド [158 ページ]
結果セット内のカラムにエイリアスが付与されているかどうかを示します。

1.8.1 GetColumnCount() メソッド

結果セットまたはテーブル内のカラム数を取得します。

構文

```
public virtual ul_column_num GetColumnCount ()
```

戻り値

結果セットまたはテーブル内のカラム数。

1.8.2 GetColumnID(const char *) メソッド

1 から始まるカラム ID を名前から取得します。

構文

```
public virtual ul_column_num GetColumnID (const char * columnName)
```

パラメータ

columnName カラムの名前。

戻り値

カラムが存在しない場合は 0。それ以外で、カラム名が存在しない場合は SQLE_COLUMN_NOT_FOUND。

1.8.3 GetColumnName(ul_column_num, ul_column_name_type) メソッド

1 から始まる ID を指定してカラムの名前を取得します。

構文

```
public virtual const char * GetColumnName (
    ul_column_num cid,
    ul_column_name_type type
)
```

パラメータ

cid 1 から始まるカラムの順序数。

type カラム名の必要な型。

戻り値

存在する場合は、カラム名を格納する文字列バッファへのポインタ。このポインタは静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を一時的に保持したい場合はその値のコピーを作成する必要があります。カラムが存在しない場合は NULL が返され、SQLE_COLUMN_NOT_FOUND が設定されます。

備考

選択した型、および SELECT 文でのカラムの宣言方法によっては、カラム名が [table-name].[column-name] という形式で返されることがあります。

type パラメータを使用して、どの型のカラム名を返すかを指定します。

関連情報

ul_column_name_type 列挙 [183 ページ]

1.8.4 GetColumnPrecision(ul_column_num) メソッド

数値カラムの精度を取得します。

構文

```
public virtual size_t GetColumnPrecision (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムが数値型ではないか、カラムが存在しない場合は、0 を返します。カラム名が存在しない場合は、SQLE_COLUMN_NOT_FOUND が設定されます。カラム型が数値ではない場合は、SQLE_DATATYPE_NOT_ALLOWED が設定されます。

1.8.5 GetColumnScale(ul_column_num) メソッド

数値カラムの位取りを取得します。

構文

```
public virtual size_t GetColumnScale (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムが数値型ではないか、カラムが存在しない場合は、0 を返します。カラム名が存在しない場合は、SQLE_COLUMN_NOT_FOUND が設定されます。カラム型が数値ではない場合は、SQLE_DATATYPE_NOT_ALLOWED が設定されます。

1.8.6 GetColumnSize(ul_column_num) メソッド

カラムのサイズを取得します。

構文

```
public virtual size_t GetColumnSize (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムが存在しないか、カラムの型が可変長ではない場合は、0 を返します。カラム名が存在しない場合は、SQLE_COLUMN_NOT_FOUND が設定されます。カラム型が UL_SQLTYPE_CHAR か UL_SQLTYPE_BINARY でない場合は、SQLE_DATATYPE_NOT_ALLOWED が設定されます。

1.8.7 GetColumnSQLType(ul_column_num) メソッド

カラムの SQL の型を取得します。

構文

```
public virtual ul_column_sql_type GetColumnSQLType (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムが存在しない場合は、UL_SQLTYPE_BAD_INDEX。

1.8.8 GetColumnType(ul_column_num) メソッド

カラムの記憶タイプまたはホスト変数の型を取得します。

構文

```
public virtual ul_column_storage_type GetColumnType (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムが存在しない場合は、UL_TYPE_BAD_INDEX。

1.8.9 GetConnection() メソッド

ULConnection オブジェクトを取得します。

構文

```
public virtual ULConnection * GetConnection ()
```

戻り値

この結果セットスキーマに関連付けられている ULConnection オブジェクト。

1.8.10 IsAliased(ul_column_num) メソッド

結果セット内のカラムにエイリアスが付与されているかどうかを示します。

構文

```
public virtual bool IsAliased (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムのエイリアスが使用されている場合は true、そうでない場合は false。

1.9 ULTable クラス

Ultra Light データベース内のテーブルを表します。

構文

```
public class ULTable : ULResultSet
```

メンバー

ULTable のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public virtual bool	DeleteAllRows() [164 ページ]	テーブルからすべてのローを削除します。
public virtual bool	Find(ul_column_num) [165 ページ]	現在のインデックスに基づいて、テーブルを順方向にスキャンして完全一致のルックアップを実行します。
public virtual bool	FindBegin() [165 ページ]	検索モードを開始することで、テーブルで新規に検索呼び出しを実行する準備を行います。
public virtual bool	FindFirst(ul_column_num) [166 ページ]	現在のインデックスに基づいて、テーブルを順方向にスキャンして完全一致のルックアップを実行します。
public virtual bool	FindLast(ul_column_num) [166 ページ]	現在のインデックスに基づいて、テーブルを逆方向にスキャンして完全一致のルックアップを実行します。
public virtual bool	FindNext(ul_column_num) [167 ページ]	インデックスに完全に一致する次のローを取得します。

変更子とタイプ	メソッド	説明
public virtual bool	FindPrevious(ul_column_num) [168 ページ]	インデックスに完全に一致する前のローを取得します。
public virtual ULTableSchema *	GetTableSchema() [168 ページ]	テーブルに関するスキーマ情報の取得に使用できる ULTableSchema オブジェクトを返します。
public virtual bool	Insert() [168 ページ]	新しいローをテーブルに挿入します。
public virtual bool	InsertBegin() [169 ページ]	設定されたカラムに対する挿入モードを選択します。
public virtual bool	Lookup(ul_column_num) [169 ページ]	現在のインデックスに基づいて、テーブルを順方向にスキャンしてルックアップを実行します。
public virtual bool	LookupBackward(ul_column_num) [170 ページ]	現在のインデックスに基づいて、テーブルを逆方向にスキャンしてルックアップを実行します。
public virtual bool	LookupBegin() [171 ページ]	テーブルで新規に検索を実行する準備を行います。
public virtual bool	LookupForward(ul_column_num) [171 ページ]	現在のインデックスに基づいて、テーブルを順方向にスキャンしてルックアップを実行します。
public virtual bool	TruncateTable() [172 ページ]	テーブルをトランケートし、STOP SYNCHRONIZATION DELETE を一時的にアクティブにします。

ULResultSet から継承されたメンバー

変更子とタイプ	メンバー	説明
public virtual bool	AfterLast() [97 ページ]	カーソルを最後のローの後に移動します。
public virtual bool	AppendByteChunk(ul_column_num, const ul_byte *, size_t) [99 ページ]	バイトをカラムに追加します。
public virtual bool	AppendByteChunk(const char *, const ul_byte *, size_t) [98 ページ]	バイトをカラムに追加します。
public virtual bool	AppendStringChunk(ul_column_num, const char *, size_t) [101 ページ]	文字列のチャンクをカラムに追加します。
public virtual bool	AppendStringChunk(const char *, const char *, size_t) [100 ページ]	文字列のチャンクをカラムに追加します。
public virtual bool	BeforeFirst() [102 ページ]	カーソルを最初のローの前に移動します。
public virtual void	Close() [102 ページ]	このオブジェクトを破棄します。
public virtual bool	Delete() [103 ページ]	現在のローを削除し、次の有効なローに移動します。
public virtual bool	DeleteNamed(const char *) [103 ページ]	現在のローを削除し、次の有効なローに移動します。
public virtual bool	First() [104 ページ]	カーソルを最初のローに移動します。

変更子とタイプ	メンバー	説明
public virtual bool	GetBinary(ul_column_num, p_ul_binary, size_t) [105 ページ]	カラムから値を ul_binary 値としてフェッチします。
public virtual bool	GetBinary(const char *, p_ul_binary, size_t) [105 ページ]	カラムから値を ul_binary 値としてフェッチします。
public virtual size_t	GetBinaryLength(ul_column_num) [107 ページ]	カラムの値のバイナリ長さを取得します。
public virtual size_t	GetBinaryLength(const char *) [106 ページ]	カラムの値のバイナリ長さを取得します。
public virtual size_t	GetByteChunk(ul_column_num, ul_byte *, size_t, size_t) [109 ページ]	カラムからバイナリチャンクを取得します。
public virtual size_t	GetByteChunk(const char *, ul_byte *, size_t, size_t) [108 ページ]	カラムからバイナリチャンクを取得します。
public virtual ULConnection *	GetConnection() [110 ページ]	接続オブジェクトを取得します。
public virtual bool	GetDateTime(ul_column_num, DECL_DATETIME *) [112 ページ]	カラムから値を DECL_DATETIME としてフェッチします。
public virtual bool	GetDateTime(const char *, DECL_DATETIME *) [111 ページ]	カラムから値を DECL_DATETIME としてフェッチします。
public virtual ul_double	GetDouble(ul_column_num) [113 ページ]	カラムから値を double としてフェッチします。
public virtual ul_double	GetDouble(const char *) [113 ページ]	カラムから値を double としてフェッチします。
public virtual ul_real	GetFloat(ul_column_num) [115 ページ]	カラムから値を float としてフェッチします。
public virtual ul_real	GetFloat(const char *) [114 ページ]	カラムから値を float としてフェッチします。
public virtual bool	GetGuid(ul_column_num, GUID *) [117 ページ]	カラムから値を GUID としてフェッチします。
public virtual bool	GetGuid(const char *, GUID *) [116 ページ]	カラムから値を GUID としてフェッチします。
public virtual ul_s_long	GetInt(ul_column_num) [118 ページ]	カラムから値を integer としてフェッチします。
public virtual ul_s_long	GetInt(const char *) [118 ページ]	カラムから値を integer としてフェッチします。
public virtual ul_s_big	GetIntWithType(ul_column_num, ul_column_storage_type) [120 ページ]	カラムから値を、指定された整数型としてフェッチします。
public virtual ul_s_big	GetIntWithType(const char *, ul_column_storage_type) [119 ページ]	カラムから値を、指定された整数型としてフェッチします。
public virtual const ULResultSetSchema &	GetResultSetSchema() [121 ページ]	結果セットに関する情報を取得するために使用できるオブジェクトを返します。
public virtual ul_u_long	GetRowCount(ul_u_long) [121 ページ]	テーブルのローの数を取得します。
public virtual UL_RS_STATE	GetState() [122 ページ]	カーソルの内部ステータスを取得します。

変更子とタイプ	メンバー	説明
public virtual bool	GetString(ul_column_num, char *, size_t) [124 ページ]	カラムから値を NULL で終了する文字列としてフェッチします。
public virtual bool	GetString(const char *, char *, size_t) [123 ページ]	カラムから値を NULL で終了する文字列としてフェッチします。
public virtual size_t	GetStringChunk(ul_column_num, char *, size_t, size_t) [126 ページ]	カラムから文字列チャンクを取得します。
public virtual size_t	GetStringChunk(const char *, char *, size_t, size_t) [125 ページ]	カラムから文字列チャンクを取得します。
public virtual size_t	GetStringLength(ul_column_num) [129 ページ]	カラムの値の文字列長さを取得します。
public virtual size_t	GetStringLength(const char *) [128 ページ]	カラムの値の文字列長さを取得します。
public virtual bool	IsNull(ul_column_num) [131 ページ]	カラムが NULL であるかどうかをチェックします。
public virtual bool	IsNull(const char *) [130 ページ]	カラムが NULL であるかどうかをチェックします。
public virtual bool	Last() [131 ページ]	カーソルを最後のローに移動します。
public virtual bool	Next() [131 ページ]	カーソルをロー 1 つ分進めます。
public virtual bool	Previous() [132 ページ]	カーソルをロー 1 つ分戻します。
public virtual bool	Relative(ul_fetch_offset) [132 ページ]	カーソルを、現在のカーソルの位置から、offset で指定したロー数分移動します。
public virtual bool	SetBinary(ul_column_num, p_ul_binary) [134 ページ]	カラムを ul_binary 値に設定します。
public virtual bool	SetBinary(const char *, p_ul_binary) [133 ページ]	カラムを ul_binary 値に設定します。
public virtual bool	SetDateTime(ul_column_num, DECL_DATETIME *) [139 ページ]	カラムを DECL_DATETIME 値に設定します。
public virtual bool	SetDateTime(const char *, DECL_DATETIME *) [138 ページ]	カラムを DECL_DATETIME 値に設定します。
public virtual bool	SetDefault(ul_column_num) [140 ページ]	カラムをそのデフォルト値に設定します。
public virtual bool	SetDefault(const char *) [140 ページ]	カラムをそのデフォルト値に設定します。
public virtual bool	SetDouble(ul_column_num, ul_double) [142 ページ]	カラムを double 値に設定します。
public virtual bool	SetDouble(const char *, ul_double) [141 ページ]	カラムを double 値に設定します。
public virtual bool	SetFloat(ul_column_num, ul_real) [144 ページ]	カラムを float 値に設定します。
public virtual bool	SetFloat(const char *, ul_real) [143 ページ]	カラムを float 値に設定します。

変更子とタイプ	メンバー	説明
public virtual bool	SetGuid(ul_column_num, GUID *) [145 ページ]	カラムを GUID 値に設定します。
public virtual bool	SetGuid(const char *, GUID *) [145 ページ]	カラムを GUID 値に設定します。
public virtual bool	SetInt(ul_column_num, ul_s_long) [147 ページ]	カラムを整数値に設定します。
public virtual bool	SetInt(const char *, ul_s_long) [147 ページ]	カラムを整数値に設定します。
public virtual bool	SetIntWithType(ul_column_num, ul_s_big, ul_column_storage_type) [137 ページ]	カラムを、指定された整数型の整数値に設定します。
public virtual bool	SetIntWithType(const char *, ul_s_big, ul_column_storage_type) [136 ページ]	カラムを、指定された整数型の整数値に設定します。
public virtual bool	SetNull(ul_column_num) [149 ページ]	カラムを NULL に設定します。
public virtual bool	SetNull(const char *) [148 ページ]	カラムを NULL に設定します。
public virtual bool	SetString(ul_column_num, const char *, size_t) [151 ページ]	カラムを文字列値に設定します。
public virtual bool	SetString(const char *, const char *, size_t) [150 ページ]	カラムを文字列値に設定します。
public virtual bool	Update() [151 ページ]	現在の行を更新します。
public virtual bool	UpdateBegin() [152 ページ]	カラムの設定に使用される更新モードを選択します。

このセクションの内容:

DeleteAllRows() メソッド [164 ページ]
テーブルからすべてのローを削除します。

Find(ul_column_num) メソッド [165 ページ]
現在のインデックスに基づいて、テーブルを順方向にスキャンして完全一致のルックアップを実行します。

FindBegin() メソッド [165 ページ]
検索モードを開始することで、テーブルで新規に検索呼び出しを実行する準備を行います。

FindFirst(ul_column_num) メソッド [166 ページ]
現在のインデックスに基づいて、テーブルを順方向にスキャンして完全一致のルックアップを実行します。

FindLast(ul_column_num) メソッド [166 ページ]
現在のインデックスに基づいて、テーブルを逆方向にスキャンして完全一致のルックアップを実行します。

FindNext(ul_column_num) メソッド [167 ページ]
インデックスに完全に一致する次のローを取得します。

FindPrevious(ul_column_num) メソッド [168 ページ]
インデックスに完全に一致する前のローを取得します。

GetTableSchema() メソッド [168 ページ]

テーブルに関するスキーマ情報の取得に使用できる ULTableSchema オブジェクトを返します。

Insert() メソッド [168 ページ]
新しいローをテーブルに挿入します。

InsertBegin() メソッド [169 ページ]
設定されたカラムに対する挿入モードを選択します。

Lookup(ul_column_num) メソッド [169 ページ]
現在のインデックスに基づいて、テーブルを順方向にスキャンしてルックアップを実行します。

LookupBackward(ul_column_num) メソッド [170 ページ]
現在のインデックスに基づいて、テーブルを逆方向にスキャンしてルックアップを実行します。

LookupBegin() メソッド [171 ページ]
テーブルで新規に検索を実行する準備を行います。

LookupForward(ul_column_num) メソッド [171 ページ]
現在のインデックスに基づいて、テーブルを順方向にスキャンしてルックアップを実行します。

TruncateTable() メソッド [172 ページ]
テーブルをトランケートし、STOP SYNCHRONIZATION DELETE を一時的にアクティブにします。

1.9.1 DeleteAllRows() メソッド

テーブルからすべてのローを削除します。

構文

```
public virtual bool DeleteAllRows ()
```

戻り値

成功した場合は true、失敗した場合は false。たとえば、テーブルが開いていない場合や SQL エラーが発生した場合は、false が返されます。

備考

アプリケーションによっては、テーブル内のローをすべて削除してから、新しいデータセットをテーブルにダウンロードする方が便利なことがあります。接続で stop sync プロパティが設定されている場合は、削除されたローが同期されません。

注記

別の接続からのコミットされていない挿入は削除されません。このような挿入は、他の接続が DeleteAllRows メソッドを呼び出した後にロールバックを実行した場合にも削除されません。

インデックスを使用しないでこのテーブルを開いた場合、テーブルは読み込み専用とみなされ、データを削除できません。

1.9.2 Find(ul_column_num) メソッド

現在のインデックスに基づいて、テーブルを順方向にスキャンして完全一致のルックアップを実行します。

構文

```
public virtual bool Find (ul_column_num ncols)
```

パラメータ

ncols 複合インデックスの場合に、検索中に使用するカラムの数。

戻り値

インデックスの値に一致するローがない場合は、カーソルの位置が最後のローの後ろに設定され、false が返されます。

備考

検索する値を指定するには、インデックスのカラムごとに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。

1.9.3 FindBegin() メソッド

検索モードを開始することで、テーブルで新規に検索呼び出しを実行する準備を行います。

構文

```
public virtual bool FindBegin ()
```

戻り値

成功した場合は true、失敗した場合は false。

備考

テーブルを開くのに使用されたインデックス内のカラムのみを設定できます。インデックスを使用しないでテーブルを開いた場合は、このメソッドを呼び出せません。

1.9.4 FindFirst(ul_column_num) メソッド

現在のインデックスに基づいて、テーブルを順方向にスキャンして完全一致のルックアップを実行します。

構文

```
public virtual bool FindFirst (ul_column_num ncols)
```

パラメータ

ncols 複合インデックスの場合に、検索中に使用するカラムの数。

戻り値

インデックスの値に一致するローがない場合は、カーソルの位置が最後のローの後ろに設定され、false が返されます。

備考

検索する値を指定するには、インデックスのカラムごとに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。

1.9.5 FindLast(ul_column_num) メソッド

現在のインデックスに基づいて、テーブルを逆方向にスキャンして完全一致のルックアップを実行します。

構文

```
public virtual bool FindLast (ul_column_num ncols)
```

パラメータ

ncols 複合インデックスの場合に、検索中に使用するカラムの数。

戻り値

インデックスの値に一致するローがない場合は、カーソルの位置が最初のローの前に設定され、false が返されます。

備考

検索する値を指定するには、インデックスのカラムごとに値を設定します。カーソルは、インデックスの値と完全に一致した最初のローで停止します。

1.9.6 FindNext(ul_column_num) メソッド

インデックスに完全に一致する次のローを取得します。

構文

```
public virtual bool FindNext (ul_column_num ncols)
```

パラメータ

ncols 複合インデックスの場合に、検索中に使用するカラムの数。

戻り値

それ以上インデックスに一致するローがない場合は、false。この場合、カーソルは最後のローの後ろに配置されます。

1.9.7 FindPrevious(ul_column_num) メソッド

インデックスに完全に一致する前のローを取得します。

構文

```
public virtual bool FindPrevious (ul_column_num ncols)
```

パラメータ

ncols 複合インデックスの場合に、検索中に使用するカラムの数。

戻り値

それ以上インデックスに一致するローがない場合は、false。この場合、カーソルは最初のローの前に配置されます。

1.9.8 GetTableSchema() メソッド

テーブルに関するスキーマ情報の取得に使用できる ULTableSchema オブジェクトを返します。

構文

```
public virtual ULTableSchema * GetTableSchema ()
```

戻り値

テーブルに関するスキーマ情報の取得に使用できる ULTableSchema オブジェクト。

1.9.9 Insert() メソッド

新しいローをテーブルに挿入します。

構文

```
public virtual bool Insert ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.9.10 InsertBegin() メソッド

設定されたカラムに対する挿入モードを選択します。

構文

```
public virtual bool InsertBegin ()
```

戻り値

成功した場合は true、失敗した場合は false。

備考

Set メソッド呼び出しによって代わりの値が指定されない場合、すべてのカラムは挿入時に初期値に設定されます。

1.9.11 Lookup(ul_column_num) メソッド

現在のインデックスに基づいて、テーブルを順方向にスキャンしてルックアップを実行します。

構文

```
public virtual bool Lookup (ul_column_num ncols)
```

パラメータ

ncols 複合インデックスのための、ルックアップで使用するカラムの数。

戻り値

ルックアップ後のカーソル位置が最後のローの後ろに設定された場合は false。

備考

検索する値を指定するには、インデックスのカラムごとに値を設定します。カーソルは、インデックスの値に一致するか、それより少ない値の最後のローで停止します。複合インデックスの場合、ncols パラメータはルックアップで使用するカラムの数を指定します。

1.9.12 LookupBackward(ul_column_num) メソッド

現在のインデックスに基づいて、テーブルを逆方向にスキャンしてルックアップを実行します。

構文

```
public virtual bool LookupBackward (ul_column_num ncols)
```

パラメータ

ncols 複合インデックスのための、ルックアップで使用するカラムの数。

戻り値

ルックアップ後のカーソル位置が最初のローの前に設定された場合は false。

備考

検索する値を指定するには、インデックスのカラムごとに値を設定します。カーソルは、インデックスの値に一致するか、それより少ない値の最後のローで停止します。複合インデックスの場合、ncols パラメータはルックアップで使用するカラムの数を指定します。

1.9.13 LookupBegin() メソッド

テーブルで新規に検索を実行する準備を行います。

構文

```
public virtual bool LookupBegin ()
```

戻り値

成功した場合は true、失敗した場合は false。

備考

テーブルを開くのに使用されたインデックス内のカラムのみを設定できます。インデックスを使用しないでテーブルを開いた場合は、このメソッドを呼び出せません。

1.9.14 LookupForward(ul_column_num) メソッド

現在のインデックスに基づいて、テーブルを順方向にスキャンしてルックアップを実行します。

構文

```
public virtual bool LookupForward (ul_column_num ncols)
```

パラメータ

ncols 複合インデックスのための、ルックアップで使用するカラムの数。

戻り値

ルックアップ後のカーソル位置が最後のローの後ろに設定された場合は false。

備考

検索する値を指定するには、インデックスのカラムごとに値を設定します。カーソルは、インデックスの値に一致するか、それより少ない値の最後のローで停止します。複合インデックスの場合、ncols パラメータはルックアップで使用するカラムの数を指定します。

1.9.15 TruncateTable() メソッド

テーブルをトランケートし、STOP SYNCHRONIZATION DELETE を一時的にアクティブにします。

構文

```
public virtual bool TruncateTable ()
```

戻り値

成功した場合は true、失敗した場合は false。

1.10 ULTableSchema クラス

Ultra Light のテーブルのスキーマを表します。

構文

```
public class ULTableSchema : ULResultSetSchema
```

メンバー

ULTableSchema のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

メソッド

変更子とタイプ	メソッド	説明
public virtual void	Close() [175 ページ]	このオブジェクトを破棄します。

変更子とタイプ	メソッド	説明
public virtual const char *	GetColumnDefault(ul_column_num) [175 ページ]	カラムのデフォルト値が存在する場合は取得します。
public virtual ul_column_default_type	GetColumnDefaultType(ul_column_num) [175 ページ]	カラムのデフォルトの型を取得します。
public virtual bool	GetGlobalAutoincPartitionSize(ul_column_num, ul_u_big *) [176 ページ]	分割サイズを取得します。
public virtual ul_index_num	GetIndexCount() [177 ページ]	テーブル内のインデックス数を取得します。
public virtual ULIndexSchema *	GetIndexSchema(const char *) [177 ページ]	名前を指定してインデックスのスキーマを取得します。
public virtual const char *	GetName() [178 ページ]	テーブルの名前を取得します。
public virtual ULIndexSchema *	GetNextIndex(ul_index_iter *) [178 ページ]	テーブル内の次のインデックス (スキーマ) を取得します。
public virtual const char *	GetOptimalIndex(ul_column_num) [179 ページ]	カラム値を検索するのに最適なインデックスを特定します。
public virtual ULIndexSchema *	GetPrimaryKey() [179 ページ]	テーブルのプライマリキーを取得します。
public virtual const char *	GetPublicationPredicate(const char *) [180 ページ]	文字列としてのパブリケーション述部を取得します。
public virtual ul_table_sync_type	GetTableSyncType() [180 ページ]	テーブルの同期タイプを取得します。
public virtual bool	InPublication(const char *) [181 ページ]	テーブルが、指定されたパブリケーションに含まれているかどうかをチェックします。
public virtual bool	IsColumnInIndex(ul_column_num, const char *) [181 ページ]	カラムが、指定されたインデックスに含まれているかどうかをチェックします。
public virtual bool	IsColumnNullable(ul_column_num) [182 ページ]	指定されたカラムが NULL 入力可能であるかどうかをチェックします。

ULResultSetSchema から継承されたメンバー

変更子とタイプ	メンバー	説明
public virtual ul_column_num	GetColumnCount() [154 ページ]	結果セットまたはテーブル内のカラム数を取得します。
public virtual ul_column_num	GetColumnID(const char *) [154 ページ]	1 から始まるカラム ID を名前から取得します。
public virtual const char *	GetColumnName(ul_column_num, ul_column_name_type) [155 ページ]	1 から始まる ID を指定してカラムの名前を取得します。
public virtual size_t	GetColumnPrecision(ul_column_num) [156 ページ]	数値カラムの精度を取得します。
public virtual size_t	GetColumnScale(ul_column_num) [156 ページ]	数値カラムの位取りを取得します。
public virtual size_t	GetColumnSize(ul_column_num) [157 ページ]	カラムのサイズを取得します。

変更子とタイプ	メンバー	説明
public virtual ul_column_sql_type	GetColumnSQLType(ul_column_num) [157 ページ]	カラムの SQL の型を取得します。
public virtual ul_column_storage_type	GetColumnType(ul_column_num) [158 ページ]	カラムの記憶タイプまたはホスト変数の型を取得します。
public virtual ULConnection *	GetConnection() [158 ページ]	ULConnection オブジェクトを取得します。
public virtual bool	IsAliased(ul_column_num) [158 ページ]	結果セット内のカラムにエイリアスが付与されているかどうかを示します。

このセクションの内容:

Close() メソッド [175 ページ]
このオブジェクトを破棄します。

GetColumnDefault(ul_column_num) メソッド [175 ページ]
カラムのデフォルト値が存在する場合は取得します。

GetColumnDefaultType(ul_column_num) メソッド [175 ページ]
カラムのデフォルトの型を取得します。

GetGlobalAutoincPartitionSize(ul_column_num, ul_u_big *) メソッド [176 ページ]
分割サイズを取得します。

GetIndexCount() メソッド [177 ページ]
テーブル内のインデックス数を取得します。

GetIndexSchema(const char *) メソッド [177 ページ]
名前を指定してインデックスのスキーマを取得します。

GetName() メソッド [178 ページ]
テーブルの名前を取得します。

GetNextIndex(ul_index_iter *) メソッド [178 ページ]
テーブル内の次のインデックス (スキーマ) を取得します。

GetOptimalIndex(ul_column_num) メソッド [179 ページ]
カラム値を検索するのに最適なインデックスを特定します。

GetPrimaryKey() メソッド [179 ページ]
テーブルのプライマリキーを取得します。

GetPublicationPredicate(const char *) メソッド [180 ページ]
文字列としてのパブリケーション述部を取得します。

GetTableSyncType() メソッド [180 ページ]
テーブルの同期タイプを取得します。

InPublication(const char *) メソッド [181 ページ]
テーブルが、指定されたパブリケーションに含まれているかどうかをチェックします。

IsColumnInIndex(ul_column_num, const char *) メソッド [181 ページ]
カラムが、指定されたインデックスに含まれているかどうかをチェックします。

IsColumnNullable(ul_column_num) メソッド [182 ページ]

指定されたカラムが NULL 入力可能であるかどうかをチェックします。

1.10.1 Close() メソッド

このオブジェクトを破棄します。

構文

```
public virtual void Close ()
```

1.10.2 GetColumnDefault(ul_column_num) メソッド

カラムのデフォルト値が存在する場合は取得します。

構文

```
public virtual const char * GetColumnDefault (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

デフォルト値。カラムにデフォルト値がない場合は、空の文字列を返します。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。

1.10.3 GetColumnDefaultType(ul_column_num) メソッド

カラムのデフォルトの型を取得します。

構文

```
public virtual ul_column_default_type GetColumnDefaultType (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムのデフォルトの型。

関連情報

ul_column_default_type 列挙 [182 ページ]

1.10.4 GetGlobalAutoincPartitionSize(ul_column_num, ul_u_big *) メソッド

分割サイズを取得します。

構文

```
public virtual bool GetGlobalAutoincPartitionSize (
    ul_column_num cid,
    ul_u_big * size
)
```

パラメータ

cid 1 から始まるカラムの順序数。

size 出力パラメータ。カラムの分割サイズ。テーブルのすべてのグローバルオートインクリメントカラムは、同じグローバルオートインクリメントの分割サイズを共有します。

戻り値

成功した場合は true、失敗した場合は false。

1.10.5 GetIndexCount() メソッド

テーブル内のインデックス数を取得します。

構文

```
public virtual ul_index_num GetIndexCount ()
```

戻り値

テーブル内のインデックス数。

備考

インデックスの ID とカウントは、スキーマのアップグレード中に変更されることがあります。インデックスを正しく識別するには、名前でアクセスするか、キャッシュされている ID とカウントをスキーマのアップグレード後に再表示します。

1.10.6 GetIndexSchema(const char *) メソッド

名前を指定してインデックスのスキーマを取得します。

構文

```
public virtual ULIndexSchema * GetIndexSchema (const char * indexName)
```

パラメータ

indexName インデックスの名前。

戻り値

指定されたインデックスの ULIndexSchema オブジェクト、または、このオブジェクトが存在しない場合は NULL。

1.10.7 GetName() メソッド

テーブルの名前を取得します。

構文

```
public virtual const char * GetName ()
```

戻り値

テーブル名。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。

1.10.8 GetNextIndex(ul_index_iter *) メソッド

テーブル内の次のインデックス (スキーマ) を取得します。

構文

```
public virtual ULIndexSchema * GetNextIndex (ul_index_iter * iter)
```

パラメータ

iter 繰り返し変数へのポインタ。

戻り値

ULIndexSchema オブジェクト。または、反復が完了したときは NULL。

備考

最初の呼び出しの前に、iter 値を ul_index_iter_start 定数に初期化します。

関連情報

ul_index_iter_start 変数 [187 ページ]

1.10.9 GetOptimalIndex(ul_column_num) メソッド

カラム値を検索するのに最適なインデックスを特定します。

構文

```
public virtual const char * GetOptimalIndex (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

インデックスの名前、またはカラムがインデックス付けされていない場合は NULL。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値をしばらく保持したい場合は、そのコピーを作成してください。

1.10.10 GetPrimaryKey() メソッド

テーブルのプライマリキーを取得します。

構文

```
public virtual ULIndexSchema * GetPrimaryKey ()
```

戻り値

テーブルのプライマリキーの ULIndexSchema オブジェクト。

1.10.11 GetPublicationPredicate(const char *) メソッド

文字列としてのパブリケーション述部を取得します。

構文

```
public virtual const char * GetPublicationPredicate (const char * pubName)
```

パラメータ

pubName パブリケーションの名前。

戻り値

指定されたパブリケーションのパブリケーション述部文字列。この値は静的バッファを指します。静的バッファの内容は、それ以降の Ultra Light の呼び出しによって変更される可能性があるため、値を保持したい場合は、そのコピーを作成してください。

1.10.12 GetTableSyncType() メソッド

テーブルの同期タイプを取得します。

構文

```
public virtual ul_table_sync_type GetTableSyncType ()
```

戻り値

テーブル同期タイプ。

備考

このメソッドは、テーブルが同期に参加する方法、および CREATE TABLE 文の SYNCHRONIZE 制約句によってテーブルを作成するときの定義方法を示します。

関連情報

ul_table_sync_type 列挙 [185 ページ]

1.10.13 InPublication(const char *) メソッド

テーブルが、指定されたパブリケーションに含まれているかどうかをチェックします。

構文

```
public virtual bool InPublication (const char * pubName)
```

パラメータ

pubName パブリケーションの名前。

戻り値

テーブルがパブリケーションに含まれている場合は true、含まれていない場合は false。

1.10.14 IsColumnInIndex(ul_column_num, const char *) メソッド

カラムが、指定されたインデックスに含まれているかどうかをチェックします。

構文

```
public virtual bool IsColumnInIndex (
    ul_column_num cid,
    const char * indexName
)
```

パラメータ

cid 1 から始まるカラムの順序数。
indexName インデックスの名前。

戻り値

カラムがインデックスに含まれている場合は true、含まれていない場合は false。

1.10.15 IsColumnNullable(ul_column_num) メソッド

指定されたカラムが NULL 入力可能であるかどうかをチェックします。

構文

```
public virtual bool IsColumnNullable (ul_column_num cid)
```

パラメータ

cid 1 から始まるカラムの順序数。

戻り値

カラムが NULL 入力可能である場合は true、そうでない場合は false を返します。

1.11 ul_column_default_type 列挙

カラムのデフォルト型を識別します。

構文

```
enum ul_column_default_type
```

メンバー

メンバー名	説明
ul_column_default_none	カラムにデフォルト値はありません。

メンバー名	説明
ul_column_default_autoincrement	カラムのデフォルトは AUTOINCREMENT です。
ul_column_default_global_autoincrement	カラムのデフォルトは GLOBAL AUTOINCREMENT です。
ul_column_default_current_timestamp	カラムのデフォルトは CURRENT TIMESTAMP です。
ul_column_default_current_utc_timestamp	カラムのデフォルトは CURRENT UTC TIMESTAMP です。
ul_column_default_current_time	カラムのデフォルトは CURRENT TIME です。
ul_column_default_current_date	カラムのデフォルトは CURRENT DATE です。
ul_column_default_newid	カラムのデフォルトは NEWID() です。
ul_column_default_other	カラムのデフォルトはユーザ指定の定数です。

関連情報

GetColumnDefaultType(ul_column_num) メソッド [175 ページ]

1.12 ul_column_name_type 列挙

結果セットの記述時にカラムの名前を取得する方法を制御する値を指定します。

構文

```
enum ul_column_name_type
```

メンバー

メンバー名	説明
ul_name_type_sql	SELECT 文の場合は、エイリアスまたは相関名を返します。 テーブルの場合は、カラム名を返します。
ul_name_type_sql_column_only	SELECT 文の場合は、エイリアスまたは相関名を返し、指定されたテーブルの名前を除外します。 テーブルの場合は、カラム名を返します。
ul_name_type_base_table	確認できる場合は、基本となるテーブルの名前を返します。 このテーブルがデータベーススキーマに存在しない場合は、空の文字列を返します。

メンバー名	説明
ul_name_type_base_column	確認できる場合は、基本となるカラムの名前を返します。 このカラムがデータベーススキーマに存在しない場合は、空の文字列を返します。
ul_name_type_qualified	ULResultSetSchema.GetColumnName メソッドと一緒に使用したときに、基本となる修飾カラム名を確認できる場合は、このカラム名を返します。 返される名前は、次の値のいずれかであり、この順序で確認されます。 1. 表現される相関テーブル 2. 表現されるテーブルのカラムの名前 3. カラムのエイリアス名 4. 空の文字列
ul_name_type_base	GetColumnName メソッドと一緒に使用したときにテーブルの名前で修飾されたカラムの名前が返されることを示します。 取得されるカラム名がクエリのベーステーブルに関連付けられている場合は、ベーステーブル名がカラムの修飾子として使用されます (つまり、base_table_name.column_name 値が返されます)。取得されるカラム名がクエリの相関テーブルのカラムを表している場合は、相関名がカラムの修飾子として使用されます (つまり、correl_table_name.col_name 値が返されます)。カラムにエイリアスがある場合は、エイリアスを使用しているカラムの修飾名が返されますが、エイリアスは、修飾名の一部ではありません。それ以外の場合は、空の文字列が返されます。

関連情報

GetColumnName(ul_column_num, ul_column_name_type) メソッド [155 ページ]

1.13 ul_index_flag 列挙

インデックスのプロパティを識別するフラグ (ビットフィールド)。

構文

```
enum ul_index_flag
```

メンバー

メンバー名	説明	値
ul_index_flag_primary_key	インデックスはプライマリキーです。	0x0001
ul_index_flag_unique_key	インデックスはプライマリキーであるか、一意性制約に対して作成されたインデックスです (NULL は許可されません)。	0x0002
ul_index_flag_unique_index	インデックスは UNIQUE フラグで作成されました (またはプライマリキーです)。	0x0004
ul_index_flag_foreign_key	インデックスは外部キーです。	0x0010
ul_index_flag_foreign_key_nullable	外部キーは NULL を許可します。	0x0020
ul_index_flag_foreign_key_check_on_commit	参照整合性チェックがコミット時に (挿入時や更新時ではなく) 実行されます。	0x0040

関連情報

GetIndexFlags() メソッド [76 ページ]

1.14 ul_table_sync_type 列挙

テーブルの同期タイプを識別します。

構文

```
enum ul_table_sync_type
```

メンバー

メンバー名	説明
ul_table_sync_on	変更されたすべてのローが同期されます (デフォルトの動作)。 このイニシャライザは、CREATE TABLE 文の SYNCHRONIZE ON 句に対応します。

メンバー名	説明
ul_table_sync_off	テーブルは同期されません。 このイニシャライザは、CREATE TABLE 文の SYNCHRONIZE OFF 句に対応します。
ul_table_sync_upload_all_rows	未変更のローを含め、常にすべてのローをアップロードします。 このイニシャライザは、CREATE TABLE 文の SYNCHRONIZE ALL 句に対応します。
ul_table_sync_download_only	変更がアップロードされることはありません。 このイニシャライザは、CREATE TABLE 文の SYNCHRONIZE DOWNLOAD 句に対応します。

関連情報

GetTableSyncType() メソッド [180 ページ]

1.15 UL_BLOB_CONTINUE 変数

ULResultSet.GetStringChunk メソッドまたは ULResultSet.GetByteChunk メソッドを使用してデータを読み込む場合に使用します。

構文

```
#define UL_BLOB_CONTINUE
```

備考

この値は、読み込むデータのチャンクが、最後のチャンクが読み込まれた位置から続いている必要があることを示します。

関連情報

GetStringChunk(ul_column_num, char *, size_t, size_t) メソッド [126 ページ]
GetByteChunk(ul_column_num, ul_byte *, size_t, size_t) メソッド [109 ページ]

1.16 ul_index_iter_start 変数

テーブル内のインデックス反復を初期化する場合に GetNextIndex メソッドで使用されます。

構文

```
#define ul_index_iter_start
```

関連情報

GetNextIndex(ul_index_iter *) メソッド [178 ページ]

1.17 ul_publication_iter_start 変数

データベースでのパブリケーション反復を初期化する場合に GetNextPublication メソッドで使用されます。

構文

```
#define ul_publication_iter_start
```

関連情報

GetNextPublication(ul_publication_iter *) メソッド [62 ページ]

1.18 ul_table_iter_start 変数

データベースでのテーブル反復を初期化する場合に GetNextTable メソッドで使用されます。

構文

```
#define ul_table_iter_start
```

関連情報

GetNextTable(ul_table_iter *) メソッド [63 ページ]

2 このマニュアルの印刷、再生、および再配布

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. ここに示したものとそれ以外のすべての版権と商標の表示をすべてのコピーに含めること。
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一切の行為をしないこと。

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。

重要免責事項および法的情報

コードサンプル

この文書に含まれるソフトウェアコード及び / 又はコードライン / 文字列 (「コード」) はすべてサンプルとしてのみ提供されるものであり、本稼動システム環境で使用することが目的ではありません。「コード」は、特定のコードの構文及び表現規則を分かりやすく説明及び視覚化することのみを目的としています。SAP は、この文書に記載される「コード」の正確性及び完全性の保証を行いません。更に、SAP は、「コード」の使用により発生したエラー又は損害が SAP の故意又は重大な過失が原因で発生させたものでない限り、そのエラー又は損害に対して一切責任を負いません。

アクセシビリティ

この SAP 文書に含まれる情報は、公開日現在のアクセシビリティ基準に関する SAP の最新の見解を表明するものであり、ソフトウェア製品のアクセシビリティ機能の確実な提供方法に関する拘束力のあるガイドラインとして意図されるものではありません。SAP は、この文書に関する一切の責任を明確に放棄するものです。ただし、この免責事項は、SAP の意図的な違法行為または重大な過失による場合は、適用されません。さらに、この文書により SAP の直接的または間接的な契約上の義務が発生することは一切ありません。

ジェンダーニュートラルな表現

SAP 文書では、可能な限りジェンダーニュートラルな表現を使用しています。文脈により、文書の読者は「あなた」と直接的な呼ばれ方をされたり、ジェンダーニュートラルな名詞 (例: 「販売員」又は「勤務日数」) で表現されます。ただし、男女両方を指すとき、三人称単数形の使用が避けられない又はジェンダーニュートラルな名詞が存在しない場合、SAP はその名詞又は代名詞の男性形を使用する権利を有します。これは、文書を分かりやすくするためです。

インターネットハイパーリンク

SAP 文書にはインターネットへのハイパーリンクが含まれる場合があります。これらのハイパーリンクは、関連情報を見いだすヒントを提供することが目的です。SAP は、この関連情報の可用性や正確性又はこの情報が特定の目的に役立つことの保証は行いません。SAP は、関連情報の使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。すべてのリンクは、透明性を目的に分類されています (http://help.sap.com/disclaimer を参照)。

go.sap.com/registration/contact.html