SYBASE®
An SAP® Company

プログラマーズガイド

Perl 用 Adaptive Server® Enterprise データベース・ドライバ バージョン 15.7 SP100

ドキュメント ID： DC01818-01-1570100-01
改訂： 2013 年 5 月

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

Perl 用 Adaptive Server Enterprise データベースドライバ 1
Perl ドライバモジュール 1
Perl 用ドライバのインストールと設定 2
Perl アプリケーションの開発 2
DSN スタイル接続プロパティのサポート 2
現時点でサポートされているデータベースハンドル属性 6
Perl でサポートされているデータ型 9
複数文の使用 9
サポートされている文字の長さ 11
ロケールと文字セットの設定 11
動的 SQL のサポート、プレースホルダ、バインドパラメータ 11
ストアドプロシージャによるプレースホルダサポート 13
サポートされているプライベートドライバメソッド 16
デフォルトの日付変換と表示フォーマット 16
text と image のデータ処理 17
エラー処理 19
セキュリティサービスの設定 20
例 21
Perl エラーメッセージ 28
その他のリソース 32
用語解説 35
索引 37

Perl 用 Adaptive Server Enterprise データベースドライバ

Perl スクリプト言語用の Adaptive Server® Enterprise データベースドライバを使用すると、Perl 開発者は Perl スクリプトを使用して、Adaptive Server のデータベースに接続し、情報をクエリしたり変更したりできます。

Perl ドライバモジュール

DBD::SybaseASE は Perl スクリプト言語用の Adaptive Server データベースドライバです。

Perl スクリプト言語用の DBD::SybaseASE データベースドライバは、汎用 Perl DBI インタフェースで呼び出され、Perl DBI API 呼び出しを CT-Library を使用した Open Client SDK 経由で、Adaptive Server が理解できる形式に変換します。

DBI と DBD::SybaseASE を使用することにより、Perl スクリプトで Adaptive Server Enterprise データベースサーバに直接アクセスできます。

汎用 Perl DBI API の仕様により、実際に使用されるデータベースから独立したデータベースインタフェースを提供するメソッドのセットを定義します。

Perl DBI のプログラマブル API 呼び出しについては、http://search.cpan.org/~timb/DBI-1.616/DBI.pm に説明があります。

注意： この DBI がなければ DBD::SybaseASE ドライバは動作しません。この DBI にはユーザに見えるすべての API が含まれています。

必要なコンポーネント
Perl プログラミング言語を使用して Adaptive Server データベースにアクセスするには、以下のコンポーネントが必要です。

- Perl のインストール - データベースベンダを意識しない汎用のコアデータベース API。
- DBD::SybaseASE - Perl スクリプト言語用のデータベースドライバ。
- CT-Library - (CT-Lib API) は Open Client スイートの一部です。CT-Library は Adaptive Server にコマンドを送信し、結果を処理します。
- Adaptive Server Enterprise
- Perl

バージョン要件
プラットフォームのサポートの詳細は、使用しているプラットフォームの『Software Developer's Kit/Open Server インストールガイド』を参照してください。

- Adaptive Server Enterprise - バージョン 15.7 以上。
- Open Client および Open Server - バージョン 15.7 以上。
- Perl - バージョン 5.14.0 または 5.14.1。
- DBD::SybaseASE ドライバ - 特定のバージョン要件はなし。
- CT-Library - (CT-Lib API) バージョン 15.7。
- Perl DBI - バージョン 1.616。
  Sybase® インストーラは、Perl がインストールされているか、ターゲットシステムにドライバ依存ソフトがインストールされているかをチェックしません。

注意： 使用するプラットフォームにリリースされた Perl ドライバのビルドモードは、Perl のインストールと DBI のビルドモードを決定します。たとえば、Linux ではスレッドが有効の 64 ビットモードでドライバがリリースされます。このため、スレッドを有効にしたフル 64 ビットモードで Perl を設定する必要があります。このビルドモード要件は DBI インタフェースにも適用されます。

Perl 用ドライバのインストールと設定

Perl 用のデータベースドライバは Sybase インストーラでインストールできるコンポーネントです。

Perl 用のデータベースドライバは、インストールタイプとして[カスタム]を選択した場合はオプションでインストールします。選択したインストールタイプが[標準]または[フル]の場合は、このドライバはデフォルトでインストールされます。インストールと設定のインストラクションについては、ご使用のプラットフォームに対応した『Software Developer's Kit/Open Server インストールガイド』を参照してください。

Perl アプリケーションの開発

Perl DBI API を使用して Perl アプリケーションを開発します。

DSN スタイル接続プロパティのサポート

ドライバは、接続時に特定の属性を設定するための DSN メカニズムを使用します。

DSN 属性構文は、Open Source DBD::Sybase ドライバと同じです。 したがって、Perl スクリプトを変更したり、DBD::Sybase と DBD::SybaseASE で異なるバージョ

ンを保持する必要はありません。 ただし、DBD::SybaseASE は、無効とみなされている一部の属性をサポートしません。 「現在サポートされていない DSN 構文」を参照してください。

SybaseASE ドライバの connect 構文

dbi:SybaseASE: セクションによってドライバのパッケージ名が取得されるので、次の構文でロードできます。

```
DBI->connect("dbi:SybaseASE:attr=value;attr=value", $user_id,
$password, %attrib);
```

DSN がドライバに渡されると、この部分が削除され、残りの文字列には、解析対象のキーと値のペアが格納されます。

注意： *$user_id* と *$password* のクレデンシャルは、別々の API 引数であり、DSN 文字列の一部ではありません。

%attrib 引数は省略可能で、接続時にオプションを設定する一連のキーと値のペアをカンマで区切って指定します。 これらは、**connect()** 呼び出し中にドライバに渡され、処理されます。 例を示します。

```
DBI->connect("dbi:SybaseASE:server=mumbles; user, password,
PrintError => 1, AutoCommit = 0);
```

属性とメソッド

サーバへの接続時にサポートされる属性は以下のとおりです。

属性	説明
server	接続先のサーバの名前を指定する。 ドライバは現在、このオプションが設定されていることを前提としている。 server を指定しない場合、ENV{"DSQUERY"} メカニズムを使用してサーバ名を取得する。
database	接続時に、サーバ内のどのデータベースをターゲットデータベースにするかを指定する。 database を指定しない場合、マスタデータベースが使用される。
hostname	値セクションで、このプロセスの `sysprocesses` テーブルに格納されるホスト名を指定する。 hostname を指定しない場合、Perl アプリケーションが実行されるホストが使用される。
language	この接続で使用するロケールを指定する。 language を指定しない場合、CS_LC_ALL という内部デフォルトロケールが使用される。
charset	この接続で使用する charset を指定する。 charset を指定しない場合、内部デフォルトの **utf8** が使用される。

属性	説明
host; port	interfaces ファイルのエントリを利用する代わりに、使用するホストとポートの組み合わせを指定する。 **注意：** Perl DSN 構文では、host と port は別々のオプションである。 次のような代替 DSN 形式は現在サポートされていない。 `host:port=mumbles:1234` interface ファイルを使用せずに host と port の DSN オプションを指定する場合、host と port だけで接続できなければならない。 DSN 属性 “server=” を host と port の組み合わせとともに指定すると、接続は失敗する。 そのため、host と port を使用するか、server のみを使用して、接続を確立する必要がある。 2 つの DSN 属性 (server と host/port) は相互に排他的である。
timeout	接続タイムアウト値を指定する。 timeout を指定しない場合は、0 か負の値を設定する。
loginTimeout	ログインタイムアウト値を秒単位で指定する。 デフォルト値は 60 秒。 この属性を有効にするには、**loginTimeout=*value in seconds*** を設定する。
tds_keepalive	接続の KEEP_ALIVE 属性を指定する。 この属性を有効にするには、**tds_keepalive=*1*** を設定する。
packetSize	接続の TDS パケットサイズを指定する。 ドライバで設定される下限値はデフォルトで 2048。 最大値はサーバにより決定され、ドライバでは設定されない。
maxConnect	許可される接続の数を増減する。 有効な値の範囲は 1 〜 128。デフォルトは 25。
encryptPassword	パスワード暗号化を使用するかどうかを指定する。 この属性を有効にするには、**encryptPassword=*1*** を設定する。
sslCAFile	`trusted.txt` ファイルの代替ロケーションを指定する。 最大 256 文字の絶対パスを指定する。
scriptName	アプリケーションを起動する最上位の Perl スクリプトの選択名を指定する。 この名前はアプリケーション名として `sysprocesses` テーブルに表示される。 この値を指定しない場合、Perl 内部環境から取得されたデフォルトのアプリケーション名が使用される。 この値には最大 256 文字を使用できる。 **注意：** SybaseASE ドライバに渡されるアプリケーション名は、DSN **scriptName** オプションを使用して設定されるか、Perl 内部環境から取得されます。
interfaces	Sybase interfaces ファイルの代替ロケーションを指定する。 **sslCAFile** オプションと **scriptName** オプションには同じ制約が適用される。

属性値は、ドライバが認識する限り何度でも繰り返せます。 不正な属性は **DBI->connect()** 呼び出しが失敗する原因になります。

注意： 属性名は Open Source Sybase Perl ドライバに準拠します。

DSN 固有の例:

```
$dbh = DBI->connect("dbi:SybaseASE:server=mumbles", $user, $passwd);
```

または、DSQUERY 環境変数を使用します。

```
my $srv = $ENV{"DSQUERY"};
$dbh = DBI->connect("dbi:SybaseASE:server=$srv",  $user, $passwd);
$dbh = DBI-
>connect("dbi:SybaseASE:host=tzedek.sybase.com;port=8100", $user,
$passwd);
$dbh = DBI->connect("dbi:SybaseASE:maxConnect=100", $user, $passwd);
$dbh = DBI->connect("dbi:SybaseASE:database=sybsystemprocs", $user,
$passwd);
$dbh = DBI->connect("dbi:SybaseASE:charset=iso_1", $user, $passwd);
$dbh = DBI->connect("dbi:SybaseASE:language=us_english", $user,
$passwd);
$dbh = DBI->connect("dbi:SybaseASE:packetSize=8192", $user,
$passwd);
$dbh = DBI->connect("dbi:SybaseASE:interfaces=/opt/sybase/
interfaces", $user, $passwd);
$dbh = DBI->connect("dbi:SybaseASE:loginTimeout=240", $user,
$passwd);
$dbh = DBI->connect("dbi:SybaseASE:timeout=240", $user, $passwd);
$dbh = DBI->connect("dbi:Sybase:scriptName=myScript", $user,
$password);
$dbh = DBI->connect("dbi:SybaseASE:hostname=pedigree", $user,
$password);
$dbh = DBI->connect("dbi:SybaseASE:encryptPassword=1", $user,
$password);
$dbh = DBI>connect("dbi:SybaseASE:sslCAFile=/usr/local/sybase/
trusted.txt", $user, $password,
AutoCommit => 1);
```

DSN 固有の組み合わせの例:

```
$dbh = DBI->connect("dbi:SybaseASE:server=mumbles,
database=tempdb;packetSize=8192;
language=us_english;charset=iso_1;encryptPassword=1", $user, $pwd,
AutoCommit=>1, PrintError => 0);
```

現在サポートされていない DSN 構文
以下の DSN 構文は現在サポートされていません。

- **tdsLevel**
- **kerberos**: 例

  ```
  $dbh = DBI->connect("dbi:SybaseASE:kerberos=$serverprincipal",
  '', '');
  ```

- **bulkLogin**: 例

```
$dbh = DBI->connect("dbi:SybaseASE:bulkLogin=1", $user,
$password);
```

- **serverType**

現時点でサポートされているデータベースハンドル属性

現時点でサポートされているデータベースハンドル属性を以下の表に示します。

属性	説明	デフォルト
dbh->{AutoCommit} = (0|1);	AutoCommit を無効または有効にする。	0 (オフ)
dbh->{LongTruncOK} = (0|1);	`text` 型および `image` 型のトランケーションを無効または有効にする。	0
dbh->{LongReadLen}=(int);	`text` データと `image` データのデフォルトの読み取りチャンクサイズを設定する。 例は次のとおり。 **dbh->{LongReadLen} = 64000.**	32767
dbh->{syb_show_sql} =(0|1);	この属性を設定すると、**$dbh->errstr** メカニズムによって返されるエラー文字列に現在の文が含まれる。	0
dbh->{syb_show_eed} = (0|1);	この属性を設定すると、**$dbh->errstr** によって返されるエラー文字列に拡張エラー情報が含まれる。	0
dbh->{syb_chained_txn} = (0|1);	この属性を設定すると、AutoCommit が無効のときに連鎖トランザクションが使用される。 この属性は **connect()** 呼び出し中にのみ使用する。 `$dbh = DBI->connect("dbi:SybaseASE:", $user, $pwd, {syb_chained_txn => 1});` AutoCommit が無効のときに **syb_chained_txn** を使用すると、現在のハンドルで強制コミットが行われる。 この属性を 0 に設定すると、必要に応じて、明示的な **BEGIN TRAN** が発行される。	0
dbh->{syb_use_bin_0x} = (0|1);	この属性を設定すると、結果文字列で `BINARY` 値と `VARBINARY` 値にプレフィックスとして '0x' が付く。	0
dbh->{syb_binary_images} = (0|1);	この属性を設定すると、`image` データがローバイナリ形式で返される。 設定しない場合、`image` データは 16 進文字列に変換される。	0

属性	説明	デフォルト
dbh->{syb_quoted_identifier}=(0|1);	引用符付き識別子として使用すれば、Sybase 予約語と競合する識別子を許可する。	0
dbh->{syb_rowcount}=(int);	ゼロ以外の値に設定すると、**SELECT** によって返されるロー、**UPDATE** 文または **DELETE** 文の影響を受けるローの数が *rowcount* 値に制限される。 設定を 0 に戻すと、制限が解除される。	0
dbh->{syb_flush_finish} = (0|1);	この属性を設定すると、ドライバは現在のコマンドに対して残っている結果を、実際にフェッチして排出する。 これは、ドライバによって発行される **ct_cancel()** コマンドの代わりに使用できる。	0
dbh->{syb_date_fmt} = datefmt string	このプライベートメソッドはデフォルトの日付変換と表示フォーマットを設定する。 「デフォルトの日付変換と表示フォーマット」を参照。	
dbh->{syb_err_handler}	エラーハンドラを実行するため、または通常のエラー処理を行う前に報告するために作成できる Perl サブルーチン。 特定の警告クラスに便利である。 「エラー処理」を参照。	0 (存在しない)
dbh->{syb_failed_db_fatal} = (0|1)	DSN に **database=*mumbles*** 属性/値のペアがあり、このデータベースが接続時に存在しない場合、**DBI->connect()** 呼び出しは失敗する。	0
dbh->{syb_no_child_con} =(0|1);	この属性を設定すると、ドライバは、**dbh** に対する複数のアクティブな文ハンドルを許可しない。 この場合、文の準備はできるが、別の文の準備が試行される前に実行して完了する必要がある。	0
dbh->{syb_cancel_request_on_error}=(0|1);	この属性を設定すると、複数文のセットを実行して、1 つの文が失敗した場合、**sth->execute()** は失敗する。	1 (オン)
dbh->{syb_bind_empty_string_as_null}= (0|1);	この属性を設定すると、NULLABLE カラム属性は、NULL 文字を表す空の文字列 (1 つのスペース) を返す。	0
dbh->{syb_disconnect_in_child} = (0|1);	閉じた接続を分岐をまたいで処理する。 子が無効になっている場合、DBI により接続は閉じられる。	0
dbh->{syb_enable_utf8} = (0|1);	この属性を設定すると、`UNICHAR`、`UNIVARCHAR`、および `UNITEXT` は **utf8** に変換される。	0

属性	説明	デフォルト
sth->syb_more_results} = (0|1);	「複数の結果セット」を参照。	
sth->{syb_result_type} = (0|1);	この属性を設定すると、記号による CS_ バージョンの代わりに数値による結果が返される。	0
sth->{syb_no_bind_blob} = (0|1);	この属性を設定すると、**sth->{fetch}** またはその他のバリエーションにおいて `image` カラムまたは `text` カラムが返されない。「text と image のデータ処理」を参照。	0
sth->{syb_do_proc_status} = (0|1);	**$sth->execute()** を強制的に実行し、SQL ストリームで実行されたストアドプロシージャのリターンステータスをフェッチする。 リターンステータスがゼロ以外の場合、**$sth->execute()** は `undef` を返す (つまり、失敗)。 この属性を設定しても、既存の文ハンドルには影響しない。ただし、この属性の設定後に作成された文ハンドルには影響する。 既存の **$sth** ハンドルの動作を取り消すには、次を実行する。**$sth->{syb_do_proc_status} = 0;**	0

参照：

- エラー処理 (19 ページ)
- text と image のデータ処理 (17 ページ)
- デフォルトの日付変換と表示フォーマット (16 ページ)
- 複数文の使用 (9 ページ)

サポートされていないデータベースハンドルオプション

以下のデータベースハンドルオプションはサポートされていません。

- **dbh->{syb_dynamic_supported}**
- **dbh->{syb_ocs_version}**
- **dbh->{syb_server_version}**
- **dbh->{syb_server_version_string}**
- **dbh->{syb_has_blk}**

注意： これらのオプションを使用する Perl スクリプトではエラーが生成されます。

Perl でサポートされているデータ型

Perl ドライバでは現在、文字列、数値、日付と時刻のデータ型がサポートされています。

文字列データ型	数値データ型	日付と時刻のデータ型
char	integer	datetime
varchar	smallint	date
binary	tinyint	time
varbinary	money	bigtime
text	smallmoney	bigdatetime
image	float	
unichar	real	
univarchar	double	
	numeric	
	decimal	
	bit	
	bigint	

注意： Perl は numeric データ型と decimal データ型を文字列として返します。 その他のデータ型はそれぞれのフォーマットで返します。

Sybase ASE ドライバが使用するデフォルトの時刻/日付のフォーマットはショートフォーマットです (例: Aug 7 2011 03:05PM)。

このフォーマットは C (デフォルト) ロケールに基づいています。 サポートされている他の日付/時刻のフォーマットについては、「デフォルトの日付変換と表示フォーマット」を参照してください。

参照：

- デフォルトの日付変換と表示フォーマット (16 ページ)

複数文の使用

Adaptive Server は 1 つのバッチで複数文の SQL を処理できます。

次に例を示します。

```
my $sth = $dbh->prepare("
    insert into publishers (col1, col2, col3) values (10, 12, 14)
    insert into publishers (col1, col2, col3) values (1, 2, 4)
```

```
    insert into publishers (col1, col2, col3) values (11, 13, 15)
    ");
my $rc = $sth->execute();
```

これらの文のいずれかが失敗すると、**sth->execute()** は undef を返します。**AutoCommit** がオンの場合、正常に完了した文によってデータがテーブルに挿入されることがあり、予想した結果になるとは限りません。

複数の結果セット
Perl ドライバでは、複数の文を 1 回の呼び出しで準備し、別の 1 回の呼び出しで実行できます。 たとえば、複数の select を含むストアドプロシージャを実行すると、複数の結果セットが返ります。

1 つの呼び出しで準備された複数の文の結果は、単一のデータストリームとしてクライアントに返されます。 個々の結果セットは通常の単一の結果セットとして処理されます。つまり、文ハンドルの **fetch()** メソッドが各セットの最後に undef を返します。

CT-Lib API **ct_fetch()** は CS_END_RESULTS を返し、ドライバが最後のローを取得した後でこれを undef に変換します。

ドライバにより、アプリケーションは **sth->{syb_result_type}** をチェックして結果セットを取得できます。 その後、**sth->{syb_more_results}** 文ハンドル属性を使用して、返される結果セットがほかにもまだあるかどうかを確認することができます。 **sth->{syb_results_type}** により返される (数) 値は次のいずれかです。

- CS_MSG_RESULT
- CS_PARAM_RESULT
- CS_STATUS_RESULT
- CS_COMPUTE_RESULT
- CS_ROW_RESULT

複数の結果セットの例:

```
do {
    while($a = $sth->fetch) {
    ..for example, display data..
    }
} while($sth->{syb_more_results});
```

複数の結果セットが想定される場合は、これを使用することをおすすめします。

注意： Perl ドライバは現在、**ct_cursor()** API を使用したカーソルをサポートしていません。 したがって、ドライバは CS_CURSOR_RESULT を報告しません。

DatabaseHandle (dbh) の複数のアクティブ文

$dbh にアクティブな文ハンドルがすでにある場合、**$dbh->prepare()** メソッドで新しい接続を開くことにより、1 つのデータベースハンドルで複数の文をアクティブにすることができます。

dbh->{syb_no_child_con} 属性は、この機能のオンとオフを制御します。 デフォルトでは、DatabaseHandle はオフです。これは、複数の文ハンドルがサポートされることを示します。 オンの場合、同じデータベースハンドル上で複数の文を使用することはできません。

注意： AutoCommit がオフの場合、 1 つの **$dbh** 上の複数の文ハンドルはサポートされません。 これにより、デッドロックの問題が発生するのを防ぐことができます。 また、複数の文ハンドルを同時に使用すると、複数の物理接続が使用されることになるので、トランザクションの整合性を確保できません。

サポートされている文字の長さ

さまざまなタイプの識別子でサポートされている文字の長さについて説明します。

テーブルやカラムなどの Sybase 識別子の名前の長さは、255 文字を超えてもかまいません。

TDS プロトコルの制限を受けるログイン、アプリケーション名、パスワードの長さは、30 文字を超えることはできません。

ロケールと文字セットの設定

DSN 属性 **charset** と **language** を使用して、Perl ドライバの CT-Library ロケールおよび文字セットを設定できます。

ドライバのデフォルトの文字セットは *UTF8*、デフォルトのロケールは *CS_LC_ALL* です。

動的 SQL のサポート、プレースホルダ、バインドパラメータ

Perl ドライバは、パラメータの使用など、動的な SQL をサポートします。

次に例を示します。

```
$sth = $dbh->prepare("select * from employee where empno = ?");

# Retrieve rows from employee where empno = 1024:
$sth->execute(1024);
while($data = $sth->fetch) {
    print "@$data¥n";
}
# Now get rows where empno = 2000:
$sth->execute(2000);
```

```
while($data = $sth->fetch) {
    print "@$data¥n";
}
```

注意： Perl ドライバは、'?' スタイルのパラメータをサポートしますが、':1' プレースホルダタイプをサポートしません。 プレースホルダを使用して `text` データ型および `image` データ型をバインドすることはできません。

DBD::SybaseASE は、**prepare()** メソッドに Open Client **ct_dynamic()** ファミリの API を使用します。 "?" スタイルのプレースホルダの制約と一般的な動的 SQL の使用方法については、『Sybase Open Client C プログラマーズガイド』を参照してください。

動的 SQL のサポートを示す別の例:

```
my $rc;
my $dbh;
my $sth;


# call do() method to execute a SQL statement.
#
$rc = $dbh->do("create table tt(string1 varchar(20), date datetime,
    val1 float, val2 numeric(7,2))");


$sth = $dbh->prepare("insert tt values(?, ?, ?, ?)");
$rc = $sth->execute("test12", "Jan 3 2012", 123.4, 222.33);


# alternate way, call bind_param() then execute without values in the
# execute statement.
$rc = $sth->bind_param(1, "another test");
$rc = $sth->bind_param(2, "Jan 25 2012");
$rc = $sth->bind_param(3, 444512.4);
$rc = $sth->bind_param(4, 2);
$rc = $sth->execute();

# and another execute, with args.....
$rc = $sth->execute("test", "Feb 30 2012", 123.4, 222.3334);
```

注意： 最後の文では、日付が無効なので、拡張エラー情報 (EED) がスローされます。 Perl スクリプトで、Adaptive Server エラーメッセージを **dbh->errstr** に書き込む前に **dbh->{syb_show_eed} =*1*** と設定します。

"?" スタイルのプレースホルダを示す別の例:

```
$sth = $dbh->prepare("select * from tt where date > ? and val1 > ?");
$rc = $sth->execute('Jan 1 2012', 120);

# go home....
$dbh->disconnect;
exit(0);
```

ストアドプロシージャによるプレースホルダサポート

Perl 用 Adaptive Server Enterprise データベースドライバは、入出力両方のパラメータを持つストアドプロシージャをサポートします。

ストアドプロシージャは、他の Transact-SQL 文と同じように処理されます。 ただし、Sybase ストアドプロシージャは、ストアドプロシージャコード内の return 文に対応するリターンステータスを含む追加の結果セットを返します。 数値 4043 の CS_STATUS_RESULT という名前のこの追加結果セットは単一ローであり、常に最後に返されます。

ドライバは、特殊な属性 **$sth->{syb_do_proc_status}** を使用してストアドプロシージャを処理できます。 この属性が設定されている場合、ドライバは追加結果セットを処理して、**$sth->{syb_proc_status}** にリターンステータス値を配置します。 結果セットが 0 以外の値の場合、エラーが生成されます。

例

```
$sth = $dbh->prepare("exec my_proc ¥@p1 = ?, ¥@p2 = ?");
    $sth->execute('one', 'two');
```

この例は、位置を指定するパラメータの使い方を示します。

```
$sth = $dbh->prepare("exec my_proc ?, ?");
  $sth->execute('one', 'two');
```

位置を指定するパラメータと名前付きパラメータを同じ prepare 文内に混在させることはできません。たとえば、次の文は最初のパラメータで失敗します。

```
$sth = $dbh->prepare("exec my_proc ¥@p1 = 1, ¥@p2 = ?");
```

ストアドプロシージャが出力パラメータを使用してデータを返す場合、まずそのパラメータを宣言しておくことが必要です。

```
$sth = $dbh->prepare(qq[declare @name varchar(50) exec getname abcd,
@name output]);
```

次の文のように、バインドされたパラメータを持つストアドプロシージャを呼び出すことはできません。

```
$sth = $dbh->prepare("exec my_proc ?");
  $sth->execute('foo');
```

次の文は機能します。

```
$sth = $dbh->prepare("exec my_proc 'foo'");
  $sth->execute('foo');
```

通常、ストアドプロシージャは複数の結果セットを返すので、syb_more_results が 0 になるまでループを使用してください。

```
do {
    while($data = $sth->fetch) {
```

```
        do something useful...
    }
 } while($sth->{syb_more_results});
```

パラメータの例

```
declare @id_value int, @id_name char(10)
    exec my_proc @name = 'a_string', @number = 1234,
        @id = @id_value OUTPUT, @out_name = @id_name OUTPUT
```

ストアドプロシージャが OUTPUT パラメータのみを返す場合、次の文を使用できます。

```
$sth = $dbh->prepare('select * .......');
    $sth->execute();
@results = $sth->syb_output_params();  #  this method is available in
SybaseASE.pm
```

これは、プロシージャコールですべての OUTPUT パラメータの配列を返し、他の結果を無視します。 OUTPUT パラメータがない場合、またはストアドプロシージャが失敗した場合、この配列は未定義になります。

一般的な例

```
$sth = $dbh->prepare("declare ¥@id_value int, ¥@id_name
        OUTPUT, @out_name = @id_name OUTPUT");
    $sth->execute();
    {
        while($d = $sth->fetch) {
            # 4042 is CS_PARAMS_RESULT
        if ($sth->{syb_result_type} == 4042) {
            $id_value  = $d->[0];
            $id_name   = $d->[1];
        }
    }
    redo if $sth->{syb_more_results};
}
```

OUTPUT パラメータは、特殊な結果セットでは 1 つのローとして返されます。

パラメータタイプ

ドライバは、パラメータごとに正しいパラメータタイプを確認するわけではありません。 全パラメータのデフォルト値は、**bind_param()** を使用して、サポートされているバインド型が設定されていない限り、ODBC スタイルの SQL_CHAR 値になります。

ドライバは次の ODBC スタイルのバインド型をサポートします。

- SQL_CHAR
- SQL_VARCHAR
- SQL_VARBINARY
- SQL_LONGVARCHAR

- SQL_LONGVARBINARY
- SQL_BINARY
- SQL_DATETIME
- SQL_DATE
- SQL_TIME
- SQL_TIMESTAMP
- SQL_BIT
- SQL_TINYINT
- SQL_SMALLINT
- SQL_INTEGER
- SQL_REAL
- SQL_FLOAT
- SQL_DECIMAL
- SQL_NUMERIC
- SQL_BIGINT
- SQL_WCHAR
- SQL_WLONGVARCHAR

ODBC 型は、対応する Adaptive Server データ型にドライバ内でマップされます。Sybase Adaptive Server Enterprise ODBC ドライバの『ユーザーズガイド 15.7』を参照してください。

特定の Adaptive Server でサポートされているデータ型の完全なリストを取得するには、ストアドプロシージャ **sp_datatype_info** を実行します。 次に例を示します。

```
$sth = $dbh->prepare("exec my_proc ¥@p1 = ?, ¥@p2 = ?");
    $sth->bind_param(1, 'one', SQL_CHAR);
    $sth->bind_param(2, 2.34, SQL_FLOAT);
    $sth->execute;
    ....
    $sth->execute('two', 3.456);
    etc...
```

注意： パラメータのカラムタイプを設定した後、文ハンドルを解放して再度実行しない限り、変更はできません。 SQL_NUMERIC データまたは SQL_DECIMAL データをバインドすると、総桁数または小数点以下桁数がターゲットパラメータ定義のサイズを超えた場合に、致命的な変換エラーが発生することがあります。

たとえば、ストアドプロシージャが次のように定義されているとします。

```
declare proc my_proc @p1 numeric(5,2) as...
    $sth = $dbh->prepare("exec my_proc ¥@p1 = ?");
    $sth->bind_param(1, 3.456, SQL_NUMERIC);
```

この場合、次のエラーが発生します。

```
DBD::SybaseASE::st execute failed: Server message number=241
severity=16 state=2 line=0 procedure=my_proc text=Scale error
```

```
during implicit conversion of NUMERIC value '3.456' to a
NUMERIC field.
```

これらのエラーを無視するには、**arithabort** オプションを次のように設定します。

```
$dbh->do("set arithabort off");
```

Adaptive Server のリファレンスマニュアルを参照してください。

サポートされているプライベートドライバメソッド

dbh->syb_isdead() は、接続の状態を表す true または false を返します。 false 戻り値は、特定のクラスまたは接続エラーを示します。つまり、接続の失敗を意味します。

$sth->syb_describe() は、現在の結果セットの各出力カラムの記述を含む配列を返します。 配列の各要素は、カラムを記述するハッシュの参照です。

次の例のように、NAME、TYPE、SYBTYPE、SYBMAXLENGTH、MAXLENGTH、SCALE、PRECISION、STATUS などの記述フィールドを設定できます。

```
$sth = $dbh->prepare("select name, uid from sysusers");
    $sth->execute;
    my @description = $sth->syb_describe;
    print "$description[0]->{NAME}¥n";                  # prints name
    print "$description[0]->{MAXLENGTH}¥n";    # prints 30
    etc, etc.
     ....
    while(my $row = $sth->fetch) {
    ....
}
```

注意： STATUS フィールドは、CS_CANBENULL、CS_HIDDEN、CS_IDENTITY、CS_KEY、CS_VERSION_KEY、CS_TIMESTAMP、CS_UPDATABLE、CS_UPDATECOL、CS_RETURN の値に対してテストできる文字列です。

Open Client のマニュアルを参照してください。

デフォルトの日付変換と表示フォーマット

syb_data_fmt() プライベートメソッドを使用して、独自のデフォルトの日付変換と表示フォーマットを設定できます。

Sybase の日付フォーマットは、クライアントのロケール設定によって異なります。 デフォルトの日付フォーマットは C ロケールに基づきます (例: Feb 16 2012 12:07PM)。

この同じデフォルトロケールで、次の追加入力フォーマットもサポートされます。

- 2/16/2012 12:07PM
- 2012/02/16 12:07
- 2012-02-16 12:07
- 20120216 12:07

日付入出力フォーマットを変更するには、引数が文字列の **dbh->{syb_date_fmt}** を使用します。

表 1 : サポートされている日付/時刻フォーマット

日付フォーマット	例
LONG	Nov 15 2011 11:30:11:496AM
SHORT	Nov 15 2011 11:30AM
DMY4_YYYY	Nov 15 2011
MDY1_YYYY	11/15/2011
DMY1_YYYY	15/11/2011
DMY2_YYYY	15.11.2011
DMY3_YYYY	15-11-2011
DMY4_YYYY	15 November 2011
HMS	11:30:11 AM
LONGMS	Nov 15 2011 11:30:33.532315PM

Perl 用 Adaptive Server Enterprise データベースドライバは、バージョン 15.7 までサポートしてきた日時の値のすべてをサポートします。

text と image のデータ処理

Perl 用 Adaptive Server Enterprise データベースドライバは、LONG/BLOB データの image 型と `text` 型をサポートします。 各データ型において、2GB のバイナリデータまで格納可能です。

text/image データのデフォルトサイズ制限は 32KB です。 この制限を変更するには **LongReadLen** 属性を使用します。これは、**fetch()** API の呼び出しにより設定されます。

text データまたは image データを挿入するためにバインドパラメータは使用できません。

標準 SQL を使用すると、image データは通常 16 進文字列に変換されますが、**syb_binary_images** ハンドル属性を使用してこの動作を変更できます。 また、

$binary = pack("H*", $hex_string); のような Perl 関数を使用してこの変換を実行することもできます。

DBI には BLOB スタイルの (text/image) データ型を処理するための API サポートがないため、SybaseASE.pm ファイルに格納されている関数セットをインストールし、アプリケーションレベルの Perl コードで使用して Open Client **ct_get_data()** スタイルの呼び出しを行うことができます。 **syb_ct_get_data()** と **syb_ct_send_data()** の呼び出しは、Adaptive Server 間で text データと image データを転送する Open Client 関数のラッパーです。

例

```
$sth->syb_ct_get_data($col, $dataref, $numbytes);
```

syb_ct_get_data() の呼び出しを使用して、image/text データをロー形式で 1 つずつ、またはまとめてフェッチすることができます。 この呼び出しを有効にするには、**dbh->{syb_no_bind_blob}** 文ハンドルを *1* に設定します。

syb_ct_get_data() の呼び出しは、クエリのカラム番号 (1 から始まる)、スカラ参照、およびバイト数の引数を受け取ります。 バイト数が 0 の場合、できるだけ多くのバイトが読み込まれます。 この呼び出しが機能するには、image/text カラムが select リストの最後に位置する必要があります。

呼び出しシーケンスは次のとおりです。

```
$sth = $dbh->prepare("select id, img from a_table where id = 1");
    $sth->{syb_no_bind_blob} = 1;
    $sth->execute;
    while($d = $sth->fetchrow_arrayref) {
        # The data is in the second column
        $len = $sth->syb_ct_get_data(2, ¥$img, 0);
}
```

syb_ct_get_data() は、フェッチされたバイト数を返します。データをまとめてフェッチする場合は、次の文を使用できます。

```
while(1) {
$len = $sth->syb_ct_get_data(2, $imgchunk, 1024);
... do something with the $imgchunk ...
 last if $len != 1024;
}
```

その他の TEXT/IMAGE API

syb_ct_data_info() API は、更新する image/text データ項目の CS_IODESC 構造体をフェッチまたは更新します。

次に例を示します。

```
$stat = syb_ct_data_info($action, $column, $attr)
```

- *$action* - CS_SET または CS_GET。
- *$column* - アクティブな select 文のカラム番号 (CS_SET オペレーションでは無視)。
- *$attr* - 構造体に値を設定するハッシュ参照。

最初に CS_GET を指定して **syb_ct_data_info()** を呼び出し、更新する image/text データ項目の CS_IODESC 構造体をフェッチする必要があります。 次に、**total_txtlen** 構造体要素の値を、挿入する image/text データの長さ (バイト単位) に更新します。 **log_on_update** を true に設定して、オペレーションの完全なロギングを有効にします。

CS_IODESC がフェッチされる image/text データが NULL である場合、CS_GET を指定して **syb_ct_data_info()** を呼び出すと失敗します。 CS_IODESC エントリを取得する前に、標準 SQL を使用して NULL 値を非 NULL 値 (たとえば、空の文字列) に更新します。

この例では、id カラムが 1 である image カラム内のデータを更新するとします。

1. このデータに対して CS_IODESC データを検出します。

```
$sth = $dbh->prepare("select img from imgtable where id = 1");
        $sth->execute;
    while($sth->fetch) {    # don't care about the data!
        $sth->syb_ct_data_info('CS_GET', 1);
    }
```

2. CS_IODESC 値で更新します。

```
$sth->syb_ct_prepare_send();
```

3. 挿入する新しいデータ項目のサイズを設定し、オペレーションのログが記録されないようにします。

```
$sth->syb_ct_data_info('CS_SET', 1, {total_txtlen
=> length($image), log_on_update => 0});
```

4. 次の文を使用して、1 つのまとまりとしてデータを転送します。

```
$sth->syb_ct_send_data($image, length($image));
```

5. 次の文を使用して、オペレーションをコミットします。

```
$sth->syb_ct_finish_send();
```

エラー処理

Perl および CT-Lib 用 Adaptive Server データベースドライバで発生したエラーはすべて、DBI レイヤに伝達されます。

例外には、ドライバの起動中、使用可能なコンテキストがまだない場合に報告する必要があるエラーや警告があります。

PrintError 属性が有効な場合、DBI レイヤは基本的なエラー報告を行います。 プログラムまたはシステムレベルの問題を追跡するには、DBI トレースメソッドを使用して DBI オペレーションのトレースを有効にします。

詳細なエラーメッセージ (サーバメッセージ) を追加する例は次のとおりです。

- アクティブな **dbh** で **dbh->{syb_show_sql} =*1*** を設定して、**$dbh->errstr** により返される文字列に現在の SQL 文を追加します。
- アクティブな **dbh** で **dbh->{syb_show_eed} =*1*** を設定して、**$dbh->errstr** により返される文字列に重複挿入エラーや無効な日付フォーマットなどの拡張エラー情報 (EED) を追加します。
- **syb_err_handler** 属性を使用して、標準エラーハンドラが処理を実行する前に呼び出される特定のエラーハンドラのコールバック (Perl サブルーチン) を設定します。 このサブルーチンが 0 を返す場合、エラーは無視されます。 これは、Transact-SQL の **PRINT** 文、**showplan** 出力、および **dbcc** 出力を処理する場合に便利です。
  このサブルーチンは、Sybase エラー番号、重大度、ステータス、SQL バッチの行番号、サーバ名 (存在する場合)、ストアドプロシージャ名 (存在する場合)、メッセージテキスト、SQL テキスト、およびタイプを表す文字列 "client" または "server" を含むパラメータを使用して呼び出されます。

セキュリティサービスの設定

`ocs.cfg` ファイルおよび `libtcl.cfg` ファイルを使用して、セキュリティオプションを設定します。

1. 接続については、`ocs.cfg` を使用してディレクトリとセキュリティのプロパティを設定します。

   注意： `ocs.cfg` ファイルで、ドライバ固有のオプションを設定できるように、アプリケーション名のエントリを追加します。
2. セキュリティサービスドライバとディレクトリサービスドライバをロードするように `libtcl.cfg` を編集します。
3. パスワードを暗号化するには、**encryptPassword** DSN オプションを使用します。 例:

```
DBI-
>connect("dbi:SybaseASE:server=mumbles;encryptPassword
=1", $user, $pwd);
```

例

サンプルプログラムを使用して、ストアドプロシージャの基本的な使用法を確認するとともに、pubs2 authors テーブルからローを取得します。

例 1

サンプルプログラムを使用して、Perl でのストアドプロシージャの基本的な使用法を確認します。

このプログラムは、サーバに接続し、2 つのストアドプロシージャを作成し、prepare を呼び出し、プロシージャをバインドまたは実行し、結果を STDOUT に出力した後、切断し、プログラムを終了します。

```
use strict;

use DBI qw(:sql_types);
use DBD::SybaseASE;

require_version DBI 1.51;

my $uid = "sa";
my $pwd = "";
my $srv = $ENV{"DSQUERY"} || die 'DSQUERY appears not set';
my $dbase = "tempdb";

my $dbh;
my $sth;
my $rc;

my $col1;
my $col2;
my $col3;
my $col4;

# Connect to the target server.
#
$dbh = DBI->connect("dbi:SybaseASE:server=$srv;database=$dbase",
    $uid, $pwd, {PrintError => 1});

# One way to exit if things fail.
#
if(!$dbh) {
    warn "Connection failed, check if your credentials are set
correctly?¥n";
    exit(0);
}

# Ignore errors on scale for numeric. There is one marked call below
# that will trigger a scale error in ASE. Current settings suppress
# this.
#
$dbh->do("set arithabort off")
```

```
        || die "ASE response not as expected";

# Drop the stored procedures in case they linger in ASE.
#
$dbh->do("if object_id('my_test_proc') != NULL drop proc
my_test_proc")
    || die "Error processing dropping of an object";

$dbh->do("if object_id('my_test_proc_2') != NULL drop proc
my_test_proc_2")
    || die "Error processing dropping of an object";

# Create a stored procedure on the fly for this example. This one
# takes input args and echo's them back.
#
$dbh->do(qq{
create proc my_test_proc ¥@col_one varchar(25), ¥@col_two int,
    ¥@col_three numeric(5,2), ¥@col_four date
as
    select ¥@col_one, ¥@col_two, ¥@col_three, ¥@col_four
}) || die "Could not create proc";

# Create another stored procedure on the fly for this example.
# This one takes dumps the pubs2..authors table. Note that the
# format used for printing is defined such that only four columns
#appear in the output list.
#
$dbh->do(qq{
create proc my_test_proc_2
as
    select * from pubs2..authors
}) || die "Could not create proc_2";

# Call a prepare stmt on the first proc.
#
$sth = $dbh->prepare("exec my_test_proc ¥@col_one = ?, ¥@col_two
= ?,
    ¥@col_three = ?, ¥@col_four = ?")
        || die "Prepare exec my_test_proc failed";

# Bind values to the columns. If SQL type is not given the default
# is SQL_CHAR. Param 3 gives scale errors if arithabort is disabled.
#
$sth->bind_param(1, "a_string");
$sth->bind_param(2, 2, SQL_INTEGER);
$sth->bind_param(3, 1.5411111, SQL_DECIMAL);
$sth->bind_param(4, "jan 12 2012", SQL_DATETIME);

# Execute the first proc.
#
$rc = $sth->execute || die "Could not execute my_test_proc";

# Print the bound args
#
```

```
dump_info($sth);

# Execute again, using different params.
#
$rc = $sth->execute("one_string", 25, 333.2, "jan 1 2012")
    || die "Could not execute my_test_proc";

dump_info($sth);

# Enable retrieving the proc status.
$sth->{syb_do_proc_status} = 1;

$rc = $sth->execute(undef, 0, 3.12345, "jan 2 2012")
    || die "Could not execute my_test_proc";
dump_info($sth);

$rc = $sth->execute("raisin", 1, 1.78, "jan 3 2012")
    || die "Could not execute my_test_proc";
dump_info($sth);

$rc = $sth->execute(undef, 0, 3.2233, "jan 4 2012")
    || die "Could not execute my_test_proc";
dump_info($sth);

$rc = $sth->execute(undef, 0, 3.2234, "jan 5 2012")
    || die "Could not execute my_test_proc";
dump_info($sth);

$rc = $sth->execute("raisin_2", 1, 3.2235, "jan 6 2012")
    || die "Could not execute my_test_proc";
dump_info($sth);

$rc = $sth->execute(undef, 0, 3.2236, "jan 7 2012")
    || die "Could not execute my_test_proc";
dump_info($sth);

# End of part one, generate blank line.
#
print "¥n";

# Undef the handles (not really needed but...).
#
undef $sth;
undef $rc;

# Prepare the second stored proc.
#
$sth = $dbh->prepare("exec my_test_proc_2")
        || die "Prepare exec my_test_proc_2 failed";

# Execute and print
#
$rc = $sth->execute || die "Could not execute my_test_proc_2";
dump_info($sth);
```

```
#
# An example of a display/print function.
#
sub dump_info {
    my $sth = shift;
    my @display;

    do {
    while(@display = $sth->fetchrow) {
     foreach (@display) {
        $_ = '' unless defined $_;
    }
     $col1 = $display[0];
     $col2 = $display[1];
     $col3 = $display[2];
     $col4 = $display[3];

    # Proc status is suppressed, assume proc
    # execution was always successful. Enable
    # by changing the write statement.
    #
    #write;
    write unless $col1 eq 0;
    }
} while($sth->{syb_more_results});
}

#
# The FORMAT template for this example.
#
format STDOUT_TOP =

Column1            Column2            Column3            Column4
------             ------             ------             ------
.

# Treat all data as left-justified strings
#
format STDOUT =
@<<<<<<<<<<<<   @<<<<<<<<<<<< @<<<<<<<<<<<<
@<<<<<<<<<<<<
$col1, $col2, $col3, $col4
.

# The End.....
#
$dbh->do("drop proc my_test_proc");
$dbh->do("drop proc my_test_proc_2");
$dbh->disconnect;
```

例 2

サンプルプログラムを使用して、pubs2 authors テーブルからローを取得し、tempdb に挿入して、バッチ挿入用の新しいローを追加します。 さらに、更新された authors テーブルを STDOUT に出力した後、切断し、終了します。

```
use strict;

use DBI ();
use DBD::SybaseASE ();

require_version DBI 1.51;

# trace(n) where n ranges from 0 - 15.
# use 2 for sufficient detail.
#DBI->trace(2); # 0 - 15, use 2 for sufficient detail

# Login credentials, handles and other variables.
#
my $uid = "sa";
my $pwd = "";
my $srv = $ENV{"DSQUERY"} || die 'DSQUERY appears not set';
my $dbase = "tempdb";
my $temp_table = "$dbase..authors";

my $rows;
my $col1;
my $col2;
my $dbh;
my $sth;
my $rc;

# Connect to the target server:
#
$dbh = DBI->connect("dbi:SybaseASE:server=$srv;database=$dbase",
    $uid, $pwd, {PrintError => 0, AutoCommit => 0})
    || die "Connect failed, did you set correct credentials?";

# Switch to the pubs2 database.
#
$rc = $dbh->do("use pubs2") || die "Could not change to pubs2";

# Retrieve 2 columns from pubs2..authors table.
#
$sth = $dbh->prepare(
    "select au_lname, city from authors where state = 'CA'")
    || die "Prepare select on authors table failed";

$rc = $sth->execute
    || die "Execution of first select statement failed";

# We may have rows now, present them.
#
$rows = dump_info($sth);
```

```
print "¥nTotal # rows: $rows¥n¥n";

# Switch back to tempdb, we take a copy of pubs2..authors
# and insert some rows and present these.
#
$rc = $dbh->do("use $dbase") || die "Could not change to $dbase";

# Drop the authors table in tempdb if present
#
$rc = $dbh->do("if object_id('$temp_table') != NULL drop table
$temp_table")
    || die "Could not drop $temp_table";

# No need to create a tempdb..authors table as the select into will
# do that.

$rc = $dbh->do("select * into $temp_table from pubs2..authors")
    || die "Could not select into table $temp_table";

# Example of a batch insert...
#
$sth = $dbh->prepare("
    insert into $temp_table
    (au_id, au_lname, au_fname, phone, address, city, state,
        country, postalcode) values
    ('172-39-1177', 'Simpson', 'John', '408 496-7223',
        '10936 Bigger Rd.', 'Menlo Park', 'CA', 'USA', '94025')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode) values
('212-49-4921', 'Greener', 'Morgen', '510 986-7020',
    '309 63rd St. #411', 'Oakland', 'CA', 'USA', '94618')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode ) values
('238-95-4766', 'Karson', 'Chernobyl', '510 548-7723',
    '589 Darwin Ln.', 'Berkeley', 'CA', 'USA', '94705')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode ) values
('267-41-4394', 'OLeary', 'Mich', '408 286-2428',
    '22 Cleveland Av. #14', 'San Jose', 'CA', 'USA', '95128')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode ) values
('274-80-4396', 'Straight', 'Shooter', '510 834-2919',
    '5420 College Av.', 'Oakland', 'CA', 'USA', '94609')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode ) values
```

```
('345-22-1785', 'Smiths', 'Neanderthaler', '913 843-0462',
    '15 Mississippi Dr.', 'Lawrence', 'KS', 'USA', '66044')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode ) values
('405-56-7012', 'Bennetson', 'Abra', '510 658-9932',
    '6223 Bateman St.', 'Berkeley', 'CA', 'USA', '94705')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode ) values
('427-17-2567', 'Dullest', 'Annie', '620 836-7128',
    '3410 Blonde St.', 'Palo Alto', 'CA', 'USA', '94301')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode ) values
('527-72-3246', 'Greene', 'Mstar', '615 297-2723',
    '22 Graybar House Rd.', 'Nashville', 'TN', 'USA', '37215')

insert into $temp_table
(au_id, au_lname, au_fname, phone, address, city, state,
    country, postalcode ) values
('672-91-3249', 'Yapan', 'Okiko', '925 935-4228',
    '3305 Silver Ct.', 'Walnut Creek', 'CA', 'USA', '94595')
");

$rc = $sth->execute || die "Could not insert row";

# Retrieve 2 columns from tempdb..authors table and present these
#
$sth = $dbh->prepare(
    "select au_lname, city from $temp_table where state = 'CA'")
    || die "Prepare select on $temp_table table failed";

$rc = $sth->execute
    || die "Execution of second select statement failed";

# Output
#
$rows = dump_info($sth);
print "¥nTotal # rows: $rows";
print "¥n";

sub dump_info {
    my $sth = shift;
    my @display;
    my $rows = 0;

while(@display = $sth->fetchrow) {
    $rows++;
    foreach (@display) {
    $_ = '' unless defined $_;
    }
```

```
    $col1 = $display[0];
    $col2 = $display[1];
    write;
    }
    $rows;
}

# The FORMAT template for this example.
#
format STDOUT_TOP =

Lastname            City
--------            -------
.

format STDOUT =

@<<<<<<<<<<<<    @<<<<<<<<<<<<
$col1, $col2
.

$dbh->disconnect;
```

Perl エラーメッセージ

内部ドライバ API または CT-Lib の API が失敗すると、Perl ドライバはエラーメッセージを生成します。 次に例を示します。

例 1 (メッセージ ID 3)

```
 OpenClient message: LAYER = (1) ORIGIN = (1) SEVERITY = (1) NUMBER =
(3)
 Message String: s_command_alloc() : SybaseASE : Internal error:
        The ct_cmd_alloc() api failed.
```

例 2 (メッセージ ID 14)

```
 OpenClient message: LAYER = (1) ORIGIN = (1) SEVERITY = (1) NUMBER =
(14)
 Message String: s_db_connect() : SybaseASE : Internal error:
        cannot change to database tempdb.
```

以下の表には次の 4 つのフィールドがあります。

- NUMBER = (<num>) フィールドに出力されるメッセージ ID
- ドライバで特定のコード位置に代入されるパラメータを含むメッセージテキスト
- 重大度 (警告または致命的なエラー)
- 考えられる対処方法または説明

メッセージ ID	メッセージテキスト	重大度	対処方法/説明
1	%1! unable to allocate memory.	致命的なエラー	システムリソースとメモリリソースをチェックする。 %1 には **cs_con_alloc()**、**ct_conn_alloc()**、**malloc()** などの関数名が含まれる。
2	%1! handle is null.	致命的なエラー	内部ドライバエラー。 接続または cmd ハンドルが NULL である。
3	The %1! api failed.	致命的なエラー	内部ドライバエラー。 無効な DSN 接続文字列または内部 CT-Lib が原因となっている可能性がある。 エラー文字列には、失敗した API の関数名が含まれる。
4	handle is null for statement id %1!.	予約済み	予約済み
5	failure for statement id %1!.	致命的なエラー	内部エラー。 **ct_dynamic()** API が、失敗した文 ID を解放しようとした。
6	send failure for statement id %1!.	予約済み	予約済み
7	%1! must be > 0.	致命的なエラー	DSN 文字列検証に失敗した。 DSN 文字列に不正な文字がないかどうかをチェックする。
8	%1! must be <= %2!, setting to maximum allowed value.	警告	現在許可されている接続数 (128) を超える接続を設定しようとした。
9	ct_config(CS_SET, %1!) failed. The supplied option %2! is illegal or missing.	致命的なエラー	無効なオプションが指定されたため、**ct_config()** API が失敗した。 DSN 文字列と 2 つ目のパラメータ内の文字列をチェックする。

メッセージ ID	メッセージテキスト	重大度	対処方法/説明
10	cs_locale(CS_SET, %1! %2!) failed.	致命的なエラー	**cs_locale()** API が失敗した。 エラー文字列はエラーの発生場所を示す。
11	cs_dt_info(CS_SET, CS_DT_CONVFMT) failed.	致命的なエラー	エラー文字列はエラーの発生場所を示す。
12	ct_debug(CS_SET CS_DBG_PROTOCOL) failed.	予約済み	予約済み
13	ct_con_props(CS_SET, %1!) failed. The supplied option %2! is illegal or missing.	致命的なエラー	**ct_con_props()** API が失敗した。 エラー文字列はエラーの発生場所を示す。
14	cannot change to database %1!.	致命的なエラー	ドライバで障害が発生した。 DSN をチェックして、指定したデータベース名が存在しているかどうかを確認する。
15	ct_command() failed for %1!.	致命的なエラー	**ct_command()** API が失敗した。 エラー文字列は、失敗した CMD のタイプを示す。
16	ct_send() failed for %1!.	致命的なエラー	**ct_send()** API が失敗した。 %1 内のエラー文字列は詳細情報を示す。
17	ct_describe() failed for column %1!.	致命的なエラー	**ct_describe()** API が失敗した。 エラー文字列はエラーの発生場所を示す。
18	ct_compute_info() failed on column %1! when describing column %2!.	致命的なエラー	**ct_compute_info()** API が失敗した。 エラー文字列は、関連するカラム番号とオペレーションタイプを示す。
19	conversion failed %1!.	警告	**cs_convert()** で障害が発生した。 エラー文字列はエラーの発生場所を示す。

メッセージ ID	メッセージテキスト	重大度	対処方法/説明
20	`ct_param() failed.`	致命的なエラー	**ct_param()** API が失敗した。 エラー文字列はエラーの場所を示す。
21	`ct_command() %1! failed for statement %2!.`	致命的なエラー	**ct_command()** API が失敗した。 エラー文字列は、文を含む、失敗した CMD のタイプを示す。
22	`ct_results() failed for %1!.`	致命的なエラー	**ct_results()** API が失敗した。エラー文字列は、ドライバ機能と失敗した文を示す。
23	`%1! command is ineffective with autocommit enabled.`	警告	エラー文字列は、試行された無効なコミットまたはロールバックを示す。
24	`ct_dynamic(CS_PREPARE) failed on statement %1!.`	致命的なエラー	動的 prepare が失敗した。 文の名前がエラー文字列に示される。
25	`ct_dynamic(CS_DESCRIBE INPUT) failed on statement %1!.`	致命的なエラー	動的 describe が失敗した。 文の名前がエラー文字列に示される。
26	`ct_dynamic(CS_EXECUTE) failed on statement %1!.`	致命的なエラー	動的 execute が失敗した。 文の名前がエラー文字列に示される。
27	`%1! database handle is inactive, not connected to server.`	致命的なエラー	無効なデータベースハンドルまたはアクティブでない接続を使用してサーバに接続しようとした。
28	`sub connections are not allowed.`	致命的なエラー	データベースハンドルがアクティブで使用中である場合、サブ接続は許可されない。

メッセージ ID	メッセージテキスト	重大度	対処方法/説明
29	`cannot bind placeholder %1!.`	致命的なエラー	プレースホルダをバインドしようとしてエラーが発生した。 エラー文字列は文を示す。
30	`unexpected cancel.`	致命的なエラー	ローの処理中に予期しないキャンセルタイプが検出された。
31	`unexpected return code from %1!.`	致命的なエラー	ローの処理中に予期しないリターンコードが検出された。
32	`invalid format %1! provided to syb_date_fmt.`	致命的なエラー	日付または時刻の変換前に無効な日付フォーマットが指定された。
33	`Fatal: multiple active statement handles on database handle without autocommit enabled.`	致命的なエラー	ユーザの Perl スクリプトで、オートコミットが有効でない状態で複数のアクティブなハンドルが使用された場合のユーザエラー。
34	`Fatal: invalid or unsupported DSN option provided.`	致命的なエラー	サポートされていないオプションまたは廃止されたオプションが Perl スクリプトに含まれている場合、DSN 解析で致命的なエラーが発生する。

その他のリソース

Perl ドライバの追加情報は以下を参照してください。

- DBI ドライバのビルド、テスト、インストール：
  http://dbi.perl.org/
- Perl DBI ユーザプログラマブル API 呼び出し：
  http://search.cpan.org/~timb/DBI-1.616/DBI.pm
- Open Client/Open Server の設定情報のマニュアル：
  『Open Client/Server 設定ガイド UNIX 版』の「設定ファイル」
- 特定の言語を使用し、その国の慣習に従って実行できるようにする、システム設定の観点からのアプリケーションの初期化：

『Open Client/Server 設定ガイド UNIX 版』の「ローカライゼーション」

- すべての Open Client/Server 製品についてのプラットフォームに関連した問題：
  『Open Client/Server プログラマーズガイド補足 UNIX 版』
- Open Client/Server ランタイム設定ファイルの使用方法：
  『Open Client Client-Library/C リファレンスマニュアル』の「ランタイム設定ファイルの使い方」の「Open Client/Server ランタイム設定ファイルの構文」
- アプリケーションで複数言語と文化的慣習のサポートを有効化するには：
  『Open Client/Open Server 開発者用国際化ガイド UNIX 版』の「国際化とローカライゼーションの概要」
- プラットフォームのサポート：
  使用しているプラットフォームの『Software Developer’s Kit/Open Server インストールガイド』

用語解説

スクリプト言語に特有の用語集

- **Client-Library –** Open Client の一部で、クライアントアプリケーションを記述するためのルーチンの集まり。Client-Library は、Sybase 製品ラインのカーソルや他の高度な機能を取り込むように設計されています。
- **CPAN –** Perl の包括的なアーカイブネットワーク。Perl のソフトウェアとマニュアルを大規模に集めた Web サイトのこと。http://www.cpan.org を参照してください。
- **CS-Library –** Client-Library と Server-Library のアプリケーションの両方で役立つユーティリティルーチンの集まり。Open Client および Open Server の両方に含まれています。
- **CT-Library –** (CT-Lib API) は Open Client スイートの一部であり、スクリプトアプリケーションで Adaptive Server に接続するために必要です。
- **DBD –** ベンダ固有のデータベースドライバで、DBI データベース API 呼び出しをターゲットデータベース SDK が理解できる形式に変換します。
- **DBI –** データベースのベンダを意識しない汎用のコアデータベース API で、Perl アプリケーションでデータベースにアクセスする現在の標準です。http://dbi.perl.org を参照してください。
- **ドライバ –** DBD::SybaseASE を構成する Perl と C コードの集まり。
- **拡張またはモジュール –** Perl 言語は Perl で記述されたモジュールまたは Perl と C の組み合わせで拡張できます。このマニュアルでは拡張とモジュールは同じものを指します。
- **Perl ディレクトリツリー –** 次のいずれかです。
  - システムにオペレーティングシステムがインストールされ、設定済みの場合に、バイナリモジュールとしてインストールされる完全な Perl のインストール。完全な Perl インストールはシステム (Perl) ツリーと呼ばれることがあり、システムアカウント (ルート、管理者) が所有します。
  - システムアカウント以外のユーザのソースから構成され、通常はシステム Perl ツリーとは別の場所にインストールされる、プライベート Perl (ディレクトリ) ツリー。これを使用すると、システムツリーに影響を与えずに新機能とバグ修正のテストができます。プライベートディレクトリは通常、ツリーを構成したアカウントが所有します。
- **Perl スクリプト –** Perl はシステムとデータベースの管理者によって広く使用されているスクリプト言語です。http://www.perl.org を参照してください。

- **スレッド (thread) –** Open Server アプリケーションからライブラリコードまでの実行のパス。また、スタック領域、ステータス情報およびイベントハンドラに対応するパス。
- **Transact-SQL –** データベース言語 SQL の機能拡張バージョン。アプリケーションは、Transact-SQL を使用して、Adaptive Server Enterprise と通信できます。

索引

A

Adaptive Server Enterprise
    データベースドライバ 1

C

connect 構文 2

い

インストールオプション 2

こ

コンポーネント
    説明 1
    必要な 1

す

スレッディング 1

そ

属性
    データベースハンドル 3
    メソッド 3
属性とメソッド 3
その他のリソース 32

は

バージョン要件 1

よ

用語解説 35