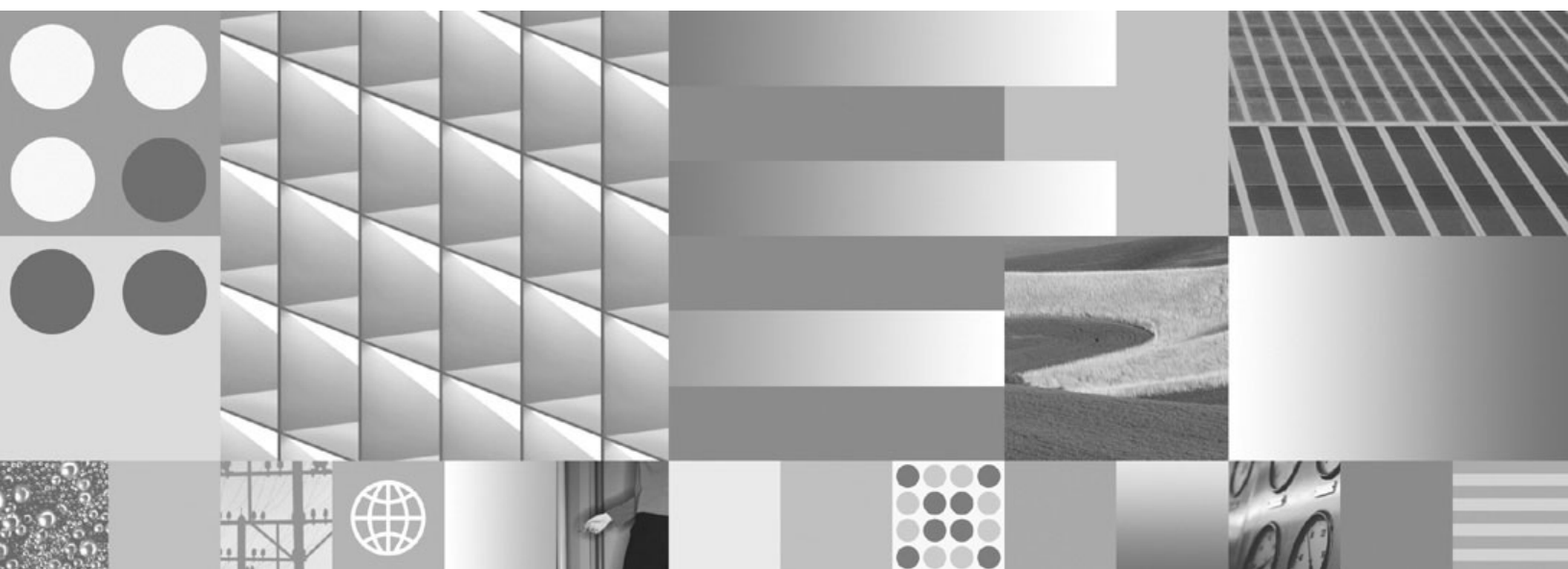


**IBM DB2 9.7  
for Linux, UNIX, and Windows**



バージョン 9 リリース 7



**Perl、PHP、Python および Ruby on Rails アプリケーションの  
開発**

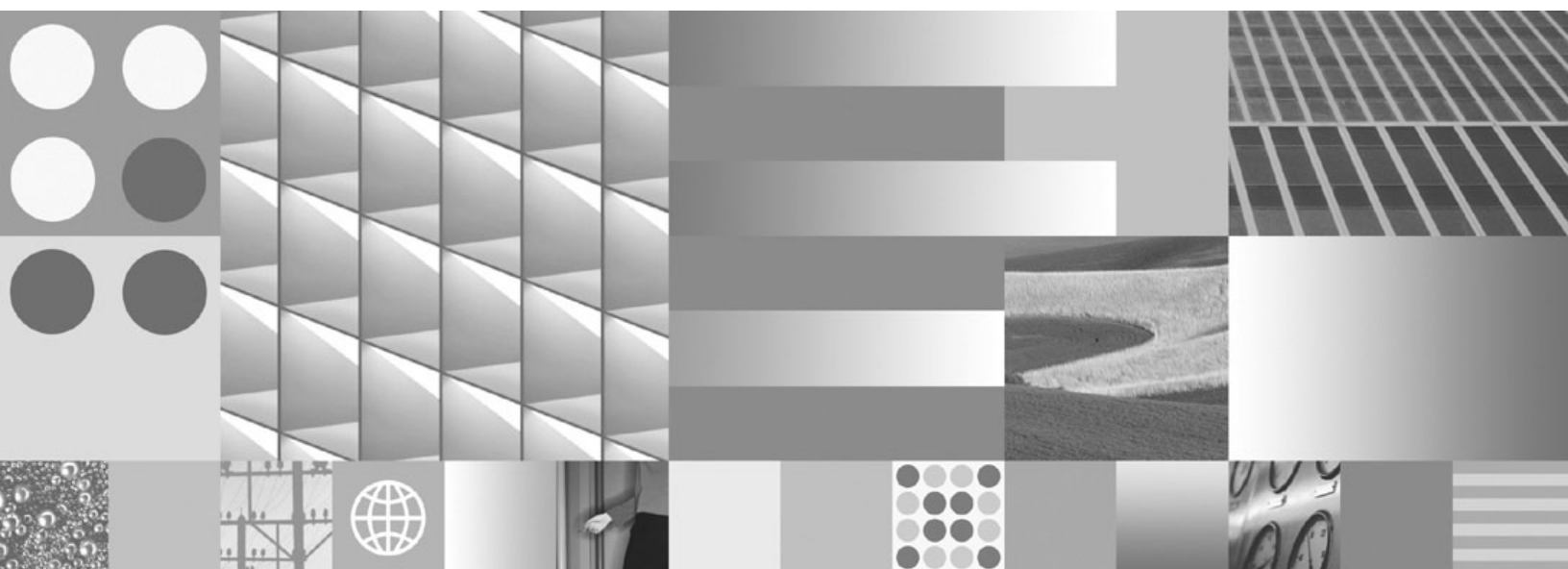
**2010 年 9 月更新版**



**IBM DB2 9.7**  
**for Linux, UNIX, and Windows**



バージョン 9 リリース 7



**Perl、PHP、Python および Ruby on Rails アプリケーションの  
開発**

**2010 年 9 月更新版**

**ご注意**

本書および本書で紹介する製品をご使用になる前に、89 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

IBM 資料は、オンラインでご注文いただくことも、ご自分の国または地域の IBM 担当員を通してお求めいただくこともできます。

- オンラインで資料を注文するには、[www.ibm.com/shop/publications/order](http://www.ibm.com/shop/publications/order) にある IBM Publications Center をご利用ください。
- ご自分の国または地域の IBM 担当員を見つけるには、[www.ibm.com/planetwide](http://www.ibm.com/planetwide) にある IBM Directory of Worldwide Contacts をお調べください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

**原典：** SC27-2447-01  
IBM DB2 9.7  
for Linux, UNIX, and Windows  
Version 9 Release 7  
Developing Perl, PHP, Python, and Ruby on Rails Applications  
Updated September, 2010

**発行：** 日本アイ・ビー・エム株式会社

**担当：** トランスレーション・サービス・センター

第1刷 2010.9

© Copyright IBM Corporation 2006, 2010.

# 目次

## 第 1 章 Perl アプリケーションの開発 . . . 1

Perl でのプログラミングに関する考慮事項 . . . . .	1
Perl ダウンロードおよび関連リソース . . . . .	1
Perl でのデータベース接続 . . . . .	2
Perl での結果のフェッチ . . . . .	3
Perl のパラメーター・マーカー . . . . .	4
Perl の SQLSTATE および SQLCODE 変数 . . . . .	5
Perl の制約事項 . . . . .	5
pureXML と Perl . . . . .	5
Perl サンプル・プログラムの実行 . . . . .	8
Perl アプリケーションからのルーチンの実行 . . . . .	8

## 第 2 章 PHP アプリケーションの開発 11

IBM データ・サーバー用の PHP アプリケーション 開発 . . . . .	11
PHP ダウンロードおよび関連リソース . . . . .	12
PHP 環境のセットアップ . . . . .	12
PHP でのアプリケーション開発 (ibm_db2) . . . . .	16
PHP でのアプリケーション開発 (PDO) . . . . .	33

## 第 3 章 Python アプリケーションの開発 47

IBM データ・サーバーのための Python、SQLAlchemy、および Django Framework ア プリケーション開発 . . . . .	47
Python ダウンロードおよび関連リソース . . . . .	48
IBM データ・サーバー用の Python 環境のセット アップ . . . . .	48
ibm_db を使用した Python でのアプリケーション 開発 . . . . .	51

## 第 4 章 Ruby on Rails アプリケーション の開発 . . . . . 65

IBM_DB Ruby ドライバーおよび Rails アダプター . . . . .	65
IBM Data Servers on Rails 入門情報 . . . . .	66
IBM_DB アダプターおよびドライバーを Ruby gem としてインストールする . . . . .	66

IBM データ・サーバーへの Rails アプリケーショ ン接続の構成 . . . . .	71
IBM Ruby ドライバーおよびトラステッド・コン テキスト . . . . .	71
IBM_DB Rails アダプターの従属関係とその影響 . . . . .	72
IBM_DB Ruby ドライバーおよび Rails アダプタ ーは JRuby でサポートされない . . . . .	73
ActiveRecord-JDBC と IBM_DB アダプター . . . . .	73
DB2 on Rails でのヒープ・サイズの考慮事項 . . . . .	74

## 付録 A. DB2 技術情報の概説 . . . . . 75

DB2 テクニカル・ライブラリー (ハードコピーまた は PDF 形式) . . . . .	76
DB2 の印刷資料の注文方法 . . . . .	79
コマンド行プロセッサから SQL 状態ヘルプを表 示する . . . . .	80
異なるバージョンの DB2 インフォメーション・セ ンターへのアクセス . . . . .	80
DB2 インフォメーション・センターでの希望する言 語でのトピックの表示 . . . . .	81
コンピューターまたはイントラネット・サーバーに インストールされた DB2 インフォメーション・セ ンターの更新 . . . . .	81
コンピューターまたはイントラネット・サーバーに インストールされた DB2 インフォメーション・セ ンターの手動更新 . . . . .	83
DB2 チュートリアル . . . . .	85
DB2 トラブルシューティング情報 . . . . .	85
ご利用条件 . . . . .	86

## 付録 B. 特記事項 . . . . . 89

## 索引 . . . . . 93



---

# 第 1 章 Perl アプリケーションの開発

---

## Perl でのプログラミングに関する考慮事項

Perl Database Interface (DBI) は、Perl で作成されたクライアント・アプリケーション用のデータベース・アクセスを提供する、オープン・スタンダードのアプリケーション・プログラミング・インターフェース (API) です。Perl DBI は、プラットフォームに依存しないデータベース・インターフェースを提供する、関数、変数、規則のセットを定義しています。

IBM® DB2® Database Driver for Perl DBI (DBD::DB2 ドライバー。  
<http://www.ibm.com/software/data/db2/perl> から入手可能) と Perl DBI モジュール  
(<http://www.perl.com> から入手可能) を使用することにより、Perl を使用する DB2  
アプリケーションを作成できます。

Perl はインタプリタ言語であり、Perl DBI モジュールは動的 SQL を使用する  
ため、DB2 アプリケーションのプロトタイプを素早く作成および修正する上  
で、Perl は理想的な言語です。Perl DBI モジュールは、CLI および JDBC と大  
変よく似たインターフェースを使用するため、Perl プロトタイプを簡単に CLI お  
よび JDBC に移植することができます。

大部分のデータベース・ベンダーは、Perl DBI モジュールのデータベース・ドラ  
イバーを提供しています。このため、Perl を使用して、さまざまなデータベース・  
サーバーのデータにアクセスするアプリケーションを作成することもできます。例  
えば、DBD::Oracle データベース・ドライバを使用して Oracle データベースに接  
続する Perl DB2 アプリケーションを作成し、Oracle データベースからデータをフ  
ェッチし、DBD::DB2 データベース・ドライバを使用して DB2 データベースに  
データを挿入することができます。

サポートされるデータベース・サーバー、インストール手順、および前提条件につ  
いては、<http://www.ibm.com/software/data/db2/perl> を参照してください。

## Perl ダウンロードおよび関連リソース

IBM データ・サーバーへアクセスする Perl アプリケーションの開発に役立つ、い  
くつかのリソースを入手できます。

表 1. Perl ダウンロードおよび関連リソース

ダウンロード	
Perl Database Interface (DBI) モジュール	<a href="http://www.perl.com">http://www.perl.com</a>
DBD::DB2 ドライバー	<a href="http://www.ibm.com/software/data/db2/perl">http://www.ibm.com/software/data/db2/perl</a>
IBM Data Server Driver Package (DS Driver)	<a href="http://www.ibm.com/software/data/support/data-server-clients/index.html">http://www.ibm.com/software/data/support/data-server-clients/index.html</a>
API 資料	
DBI API 資料	<a href="http://search.cpan.org/~timb/DBI/DBI.pm">http://search.cpan.org/~timb/DBI/DBI.pm</a>
関連リソース	

表 1. Perl ダウンロードおよび関連リソース (続き)

ダウンロード	
DB2 Perl Database Interface for LUW 技術情報 (README およびインストールの指示を含む)	<a href="http://www.ibm.com/software/data/db2/perl">http://www.ibm.com/software/data/db2/perl</a>
Perl ドライバーのバグ報告システム	<a href="http://rt.cpan.org/">http://rt.cpan.org/</a>
IBM の Open Source チームへのバグの報告	<a href="mailto:opendev@us.ibm.com">opendev@us.ibm.com</a>

## Perl でのデータベース接続

DBD::DB2 ドライバーは、DBI API によって定義された標準データベース接続機能のサポートを提供します。

Perl が DBI モジュールをロードできるようにするには、アプリケーションに次の行を含める必要があります。

```
use DBI;
```

DBI モジュールは、DBI->connect ステートメントを使用してデータベース・ハンドルを作成すると、DBD::DB2 ドライバーを自動的にロードします。以下の構文を使用します。

```
my $dbhhandle = DBI->connect('dbi:DB2:dsn', $userID, $password);
```

詳細は次のとおりです。

### \$dbhhandle

connect ステートメントが戻すデータベース・ハンドル。

### dsn

(ローカル接続用) DB2 データベース・ディレクトリーにカタログされている DB2 別名

(リモート接続用) リモート・ホストへの接続のためのホスト名、ポート番号、プロトコル、ユーザー ID、およびパスワードを含む、完全な接続ストリング

### \$userID

データベースへの接続で使用するユーザー ID

### \$password

データベースへの接続で使用するユーザー ID のパスワード

DBI API について詳しくは、<http://search.cpan.org/~timb/DBI/DBI.pm><http://search.cpan.org/~timb/DBI/DBI.pm> を参照してください。

## 例

例 1: ローカル・ホスト上のデータベースに接続します (クライアントとサーバーが同じワークステーション上にあります)

```
use DBI;

$DATABASE = 'dbname';
$USERID = 'username';
$PASSWORD = 'password';
```



```
my $dbh = DBI->connect("dbi:DB2:$DATABASE", $USERID, $PASSWORD, {PrintError => 0})
or die "Couldn't connect to database: " . DBI->errstr;

$dbh->disconnect;
```

例 2: リモート・ホスト上のデータベースに接続します (クライアントとサーバーが別々のワークステーション上にあります)

```
use DBI;

$DSN="DATABASE=sample; HOSTNAME=host; PORT=60000; PROTOCOL=TCPIP; UID=username;
PWD=password";

my $dbh = DBI->connect("dbi:DB2:$DSN", $USERID, $PASSWORD, {PrintError => 0})
or die "Couldn't connect to database: " . DBI->errstr;

$dbh->disconnect;
```

## Perl での結果のフェッチ

Perl DBI モジュールは、データベースへの接続、SQL ステートメントの準備と発行、および結果セットからの行のフェッチを行うためのメソッドを提供します。

SQL 照会から結果をフェッチする手順です。

**制約事項:** Perl DBI モジュールは動的 SQL しかサポートしていないため、Perl DB2 アプリケーションではホスト変数は使用できません。

結果をフェッチするには、次のようにします。

1. DBI->connect ステートメントを使用してデータベースに接続することにより、データベース・ハンドルを作成します。
2. 作成したデータベース・ハンドルからステートメント・ハンドルを作成します。例えば、prepare メソッドを呼び出して、ストリング引数として SQL ステートメントを渡すことにより、データベース・ハンドルからステートメント・ハンドル \$sth を戻すことができます。以下に示す Perl ステートメントは、その例です。

```
my $sth = $dbh->prepare(
    'SELECT firstme, lastname
     FROM employee '
);
```

3. ステートメント・ハンドルで execute メソッドを呼び出すことにより、SQL ステートメントを発行します。execute メソッドが正常に呼び出されると、結果セットがステートメント・ハンドルに関連付けられます。例えば、以下の Perl ステートメントを使用すると、前の例で準備したステートメントを実行できます。

```
#Note: $rc represents the return code for the execute call
my $rc = $sth->execute();
```

4. fetchrow メソッドを呼び出して、ステートメント・ハンドルに関連付けられた結果セットから行をフェッチします。Perl DBI は、列ごとに値を 1 つ持つ配列として行を戻します。例えば、以下の Perl ステートメントを使用すると、前の例にあるステートメント・ハンドルからすべての行を戻すことができます。

```
while (($firstme, $lastname) = $sth->fetchrow()) {
    print "$firstme $lastname\n";
}
```

以下の例は、データベースに接続し、Perl で作成されたアプリケーションから SELECT ステートメントを発行する方法を示しています。

```
#!/usr/bin/perl
use DBI;

my $database='dbi:DB2:sample';
my $user='';
my $password='';

my $dbh = DBI->connect($database, $user, $password)
    or die "Can't connect to $database: $DBI::errstr";

my $sth = $dbh->prepare(
    q{ SELECT firstnme, lastname
      FROM employee }
    )
    or die "Can't prepare statement: $DBI::errstr";

my $rc = $sth->execute
    or die "Can't execute statement: $DBI::errstr";

print "Query will return $sth->{NUM_OF_FIELDS} fields.¥n¥n";
print "$sth->{NAME}->[0]: $sth->{NAME}->[1]¥n";

while (($firstnme, $lastname) = $sth->fetchrow()) {
    print "$firstnme: $lastname¥n";
}

# check for problems that might have terminated the fetch early
warn $DBI::errstr if $DBI::err;

$sth->finish;
$dbh->disconnect;
```

## Perl のパラメーター・マーカー

Perl DBI モジュールは、変数入力用のパラメーター・マーカーを含む、準備済みのステートメントの実行をサポートしています。SQL ステートメントにパラメーター・マーカーを入れるには、疑問符 (?) 文字を使用するか、名前の前にコロンを付けます (:name)。

以下の Perl コードは、SELECT ステートメントの WHERE 節のパラメーター・マーカーを受け入れるステートメント・ハンドルを作成します。その後、このコードは、入力値 25000 および 35000 を使用して 2 度ステートメントを実行し、パラメーター・マーカーを置換します。

```
my $sth = $dbh->prepare(
    'SELECT firstnme, lastname
     FROM employee
     WHERE salary > ?'
    );

my $rc = $sth->execute(25000);

.
.
.

my $rc = $sth->execute(35000);
```

## Perl の SQLSTATE および SQLCODE 変数

Perl DBI モジュールは、Perl DBI データベースまたはステートメント・ハンドルに関連した SQLSTATE および SQLCODE を戻すためのメソッドを提供します。

Perl DBI のデータベース・ハンドルまたはステートメント・ハンドルに関連付けられた SQLSTATE を戻すには、`state` メソッドを呼び出します。例えば、データベース・ハンドル `$dbhhandle` に関連付けられた SQLSTATE を戻すには、次の Perl ステートメントをアプリケーションに組み込みます。

```
my $sqlstate = $dbhhandle->state;
```

Perl DBI のデータベース・ハンドルまたはステートメント・ハンドルに関連付けられた SQLCODE を戻すには、`err` メソッドを呼び出します。Perl DBI のデータベース・ハンドルまたはステートメント・ハンドルに関連付けられた SQLCODE のメッセージを戻すには、`errstr` メソッドを呼び出します。例えば、データベース・ハンドル `$dbhhandle` に関連付けられた SQLCODE を戻すには、次の Perl ステートメントをアプリケーションに組み込みます。

```
my $sqlcode = $dbhhandle->err;
```

## Perl の制約事項

Perl でのアプリケーション開発で使用可能なサポートには、いくつかの制約事項が適用されます。

Perl DBI モジュールがサポートするのは、動的 SQL だけです。複数回ステートメントを実行する必要がある場合には、ステートメントを準備する `prepare` 呼び出しを発行して、Perl アプリケーションのパフォーマンスを改善することができます。

Perl ではマルチスレッド・データベース・アクセスはサポートされていません。

ワークステーションにインストールする DBD::DB2 ドライバー・バージョンの制限に関する最新情報については、DBD::DB2 パッケージにある CAVEATS ファイルを参照してください。

## pureXML と Perl

DBD::DB2 ドライバーは、DB2 pureXML<sup>®</sup> をサポートしています。pureXML のサポートにより、DBD::DB2 ドライバーを通してデータへのより直接的なアクセスが可能になります。また、アプリケーションとデータベース間の通信がより透過性のあるものとなり、アプリケーション・ロジックの削減に役立ちます。

pureXML のサポートがあるため、DB2 データベースに XML 文書を直接挿入できます。データベースに XML 文書を挿入すると pureXML パーサーが自動的に実行されるので、アプリケーションが XML 文書を解析する必要はもはやありません。文書の解析処理がアプリケーション外で行われるため、アプリケーションのパフォーマンスが向上し、保守の労力も削減されます。DBD::DB2 ドライバーで XML 保管データを取り出す作業も簡単です。BLOB またはレコードを使用してデータにアクセスできます。

DB2 Perl Database Interface に関する情報や、最新の DBD::DB2 ドライバーをダウンロードする方法については、<http://www.ibm.com/software/data/db2/perl> を参照してください。

## 例

以下の例は、pureXML を使用する Perl プログラムです。

```
#!/usr/bin/perl
use DBI;
use strict ;

# Use DBD:DB2 module:
#   to create a simple DB2 table with an XML column
#   Add one row of data
#   retrieve the XML data as a record or a LOB (based on $datatype).

# NOTE: the DB2 SAMPLE database must already exist.

my $database='dbi:DB2:sample';
my $user='';
my $password='';

my $datatype = "record" ;
# $datatype = "LOB" ;

my $dbh = DBI->connect($database, $user, $password)
    or die "Can't connect to $database: $DBI::errstr";

# For LOB datatype, LongReadLen = 0 -- no data is retrieved on initial fetch
$dbh->{LongReadLen} = 0 if $datatype eq "LOB" ;

# SQL CREATE TABLE to create test table
my $stmt = "CREATE TABLE xmlTest (id INTEGER, data XML)";
my $sth = $dbh->prepare($stmt);
$sth->execute();

#insert one row of data into table
insertData() ;

# SQL SELECT statement returns home phone element from XML data
$stmt = qq(
    SELECT XMLQUERY ('
        ¥$d/*:customerinfo/*:phone[¥@type = "home"] '
        passing data as "d")
    FROM xmlTest
) ;

# prepare and execute SELECT statement
$sth = $dbh->prepare($stmt);
$sth->execute();

# Print data returned from select statement
if($datatype eq "LOB") {
    printLOB() ;
}
else {
    printRecord() ;
}

# Drop table
$stmt = "DROP TABLE xmlTest" ;
$sth = $dbh->prepare($stmt);
$sth->execute();

warn $DBI::errstr if $DBI::err;

$sth->finish;
$dbh->disconnect;
```

```

#####

sub printRecord {
    print "output data as as record¥n" ;

    while( my @row = $sth->fetchrow )
    {
        print $row[0] . "¥n";
    }

    warn $DBI::errstr if $DBI::err;
}

sub printLOB {
    print "output as Blob data¥n" ;

    my $offset = 0;
    my $buff="";
    $sth->fetch();
    while( $buff = $sth->blob_read(1,$offset,1000000) ) {
        print $buff;
        $offset+=length($buff);
        $buff="";
    }
    warn $DBI::errstr if $DBI::err;
}

sub insertData {

    # insert a row of data
    my $xmlInfo = qq(¥'
<customerinfo xmlns="http://posample.org" Cid="1011">
  <name>Bill Jones</name>
  <addr country="Canada">
    <street>5 Redwood</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>M6W 1E9</pcode-zip>
  </addr>
  <phone type="work">416-555-9911</phone>
  <phone type="home">416-555-1212</phone>
</customerinfo>
¥');

    my $catID = 1011 ;

    # SQL statement to insert data.
    my $Sql = qq(
    INSERT INTO xmlTest (id, data)
      VALUES($catID, $xmlInfo )
    );

    $sth = $dbh->prepare( $Sql )
    or die "Can't prepare statement: $DBI::errstr";

    my $rc = $sth->execute
    or die "Can't execute statement: $DBI::errstr";

    # check for problems
    warn $DBI::errstr if $DBI::err;
}

```

## Perl サンプル・プログラムの実行

Perl アプリケーションを作成する方法を示している、Perl サンプル・プログラムを入手できます。

Perl サンプル・プログラムを実行する前に、Perl DBI 用の最新の DB2::DB2 ドライバーをインストールする必要があります。最新のドライバーの入手方法については、<http://www.ibm.com/software/data/db2/perl> を参照してください。

DB2 データベース用の Perl サンプル・プログラムは、`sqllib/samples/perl` ディレクトリにあります。

コマンド行で Perl サンプル・プログラムに対して Perl インタープリターを実行するには、以下を行います。

インタープリター名とプログラム名 (ファイル拡張子を含む) を入力します。

- サーバーにローカル接続している場合:

```
perl dbauth.pl
```

- リモート・クライアントから接続している場合:

```
perl dbauth.pl sample <userid> <password>
```

サンプル・プログラムの中には、サポート・ファイルを実行しなければならないものもあります。例えば、`tbse1` サンプル・プログラムでは、`tbse1create.db2` CLP スクリプトで作成される複数の表が必要です。`tbse1init` スクリプト (UNIX®)、または `tbse1init.bat` バッチ・ファイル (Windows®) は、`tbse1drop.db2` を呼び出して表をドロップしてから (存在する場合)、`tbse1create.db2` を呼び出して表を作成します。そのため、`tbse1` サンプル・プログラムを実行するには、以下のコマンドを発行します。

- サーバーにローカル接続している場合:

```
tbse1init  
perl tbse1.pl
```

- リモート・クライアントから接続している場合:

```
tbse1init  
perl tbse1.pl sample <userid> <password>
```

注: リモート・クライアントの場合は、`tbse1init` または `tbse1init.bat` ファイルの接続ステートメントを修正して、`db2 connect to sample user <userid> using <password>` のように、自分のユーザー ID とパスワードをハードコーディングする必要があります。

## Perl アプリケーションからのルーチンの実行

DB2 クライアント・アプリケーションは、サポートされるホスト言語または SQL プロシージャで作成されたルーチン (ストアド・プロシージャおよびユーザー定義関数) にアクセスできます。例えば、サンプル・プログラム `spclient.pl` は、SQL プロシージャ `spserver` 共用ライブラリーにアクセスできます (データベースに存在する場合)。

ホスト言語ルーチンを構築するには、サーバー上に適切なコンパイラーをセットアップしておく必要があります。SQL プロシージャには、コンパイラーは不要です。共用ライブラリーは、サーバー上でのみ構築でき、リモート・クライアントからは構築できません。

共用ライブラリーに SQL プロシージャを作成してから、そのプロシージャに Perl アプリケーションからアクセスするには、以下を行います。

1. ライブラリーで SQL プロシージャを作成し、カタログします。例えば、サーバー上の `samples/sqlp1` ディレクトリーに移動し、次のコマンドを実行して、`spserver` ライブラリーに SQL プロシージャを作成し、カタログします。

```
db2 connect to sample
db2 -td@ -vf spserver.db2
```

2. `perl` サンプル・ディレクトリー (リモート・クライアント・ワークステーション上でも構いません) に戻り、`spserver` 共用ライブラリーにアクセスするクライアント・プログラムに対して、Perl インタープリターを実行します。

- サーバーにローカル接続している場合:

```
perl spclient
```

- リモート・クライアントから接続している場合:

```
perl spclient sample <userid> <password>
```





---

## 第 2 章 PHP アプリケーションの開発

---

### IBM データ・サーバー用の PHP アプリケーション開発

PHP: Hypertext Preprocessor (PHP) とは、Web アプリケーションの開発のために広く使用されているインタープリター型プログラミング言語です。PHP は学習しやすく、実用的なソリューションに焦点を合わせており、Web アプリケーションで一般に最も必要とされる機能をサポートしているため、Web 開発で広く使用される言語となりました。

PHP はモジュラー言語であり、拡張モジュールを使用することによって、使用できる機能をカスタマイズできます。これらの拡張モジュールを使用すれば、XML の読み取り、書き込み、および操作、SOAP クライアント/サーバーの作成、およびサーバーとブラウザーとの間の通信の暗号化などのタスクを単純化できます。ただし、PHP の最も一般的な拡張モジュールは、データベースへの読み取り/書き込みアクセスを提供するものであり、これにより動的なデータベース・ドリブンの Web サイトを簡単に作成できます。

IBM は、IBM データ・サーバー・データベースにアクセスするための、以下の PHP 拡張モジュールを提供しています。

#### **ibm\_db2**

プロシージャ型アプリケーション・プログラミング・インターフェース (API)。これは通常のデータベースの作成、読み取り、更新、および書き込み操作に加え、データベース・メタデータへの広範なアクセスも行います。ibm\_db2 拡張モジュールは、PHP 4 または PHP 5 のいずれかでコンパイルできます。この拡張モジュールは、IBM によって作成、保守、およびサポートされています。

#### **pdo\_ibm**

PDO (PHP Data Objects) 拡張モジュール用のドライバー。これは、PHP 5.1 で導入された標準オブジェクト指向データベース・インターフェースによる、IBM データ・サーバー・データベースへのアクセスを提供します。

これらの拡張は、IBM Data Server Driver Package (DS Driver) バージョン 1.7.0 に含まれています。IBM DB2 バージョン 9.7 for Linux®, UNIX, and Windows への接続において、このバージョンまたはそれ以降のバージョンがサポートされています。ibm\_db2 拡張モジュールのバージョンをチェックするには、以下のコマンドを発行することができます。

```
php --re ibm_db2
```

ibm\_db2 と pdo\_ibm の最新バージョンは、PHP Extension Community Library (PECL) (<http://pecl.php.net/>) から入手できます。

PHP アプリケーションは、以下の IBM データ・サーバーのデータベースにアクセスすることができます。

- IBM DB2 バージョン 9.1 for Linux, UNIX, and Windows、Fix Pack 2 以降

- IBM DB2 Universal Database™ (DB2 UDB) バージョン 8 for Linux, UNIX, and Windows、 Fixpak 15 以降
- i5/OS® V5R3 での IBM DB2 Universal Database へのリモート接続
- IBM i 5.4 以降での IBM DB2 へのリモート接続
- IBM DB2 for z/OS® バージョン 8 およびバージョン 9 へのリモート接続

3 番目の拡張モジュールである Unified ODBC は、これまで DB2 データベース・システムへのアクセスを提供してきました。ただし新しいアプリケーションの場合、ibm\_db2 および pdo\_ibm は Unified ODBC を上回るパフォーマンスおよび安定度における大きな利点があるため、これらのいずれかを使用してください。ibm\_db2 拡張モジュール API を使用すれば、Unified ODBC 用に以前に作成されたアプリケーションの移植は、ほぼ、アプリケーションのソース・コード全体にわたって odbc\_関数名を db2\_ に変更するだけで容易に行うことができます。

## PHP ダウンロードおよび関連リソース

IBM データ・サーバー用の PHP アプリケーション開発に役立つ多くのリソースが入手可能です。

表 2. PHP ダウンロードおよび関連リソース

ダウンロード	
完全な PHP ソース・コード <sup>1</sup>	<a href="http://www.php.net/downloads.php">http://www.php.net/downloads.php</a>
PHP Extension Community Library (PECL) からの ibm_db2 および pdo_ibm	<a href="http://pecl.php.net/">http://pecl.php.net/</a>
IBM Data Server Driver Package (DS Driver)	<a href="http://www.ibm.com/software/data/support/data-server-clients/index.html">http://www.ibm.com/software/data/support/data-server-clients/index.html</a>
Zend Server	<a href="http://www.zend.com/en/products/server/downloads">http://www.zend.com/en/products/server/downloads</a>
資料	
PHP Manual	<a href="http://www.php.net/docs.php">http://www.php.net/docs.php</a>
ibm_db2 API 資料	<a href="http://www.php.net/ibm_db2">http://www.php.net/ibm_db2</a>
PDO API 資料	<a href="http://php.net/manual/en/book.pdo.php">http://php.net/manual/en/book.pdo.php</a>
関連リソース	
PHP Web サイト	<a href="http://www.php.net/">http://www.php.net/</a>

1. Windows バイナリーを含みます。ほとんどの Linux ディストリビューションには、既にプリコンパイルされた PHP が付属しています。

## PHP 環境のセットアップ

PHP のプリコンパイルされたバイナリー・バージョンをインストールし、IBM データ・サーバーのサポートを有効にすることにより、Linux、UNIX、または Windows オペレーティング・システムで PHP 環境をセットアップできます。

Linux、UNIX、または Windows オペレーティング・システム上で最も簡単なインストールおよび構成を行う方法として、実動システムで使用するための Zend Server

をダウンロード (<http://www.zend.com/en/products/server/downloads>) してインストールすることができます。パッケージ化の詳細情報が <http://www.zend.com/en/products/server/editions> に掲載されています。

Windows 上では、プリコンパイルされたバイナリー・バージョンの PHP が、<http://www.php.net/downloads.php> からダウンロードできます。ほとんどの Linux ディストリビューションには、プリコンパイル・バージョンの PHP が組み込まれています。プリコンパイル・バージョンの PHP が組み込まれていない UNIX オペレーティング・システム上では、ユーザー独自のバージョンの PHP をコンパイルできます。

インストールおよび PHP の構成について詳しくは、<http://www.php.net/manual/en/install.php> を参照してください。

## Windows での PHP 環境のセットアップ

IBM データ・サーバーに接続し、SQL ステートメントを実行するには、その前に PHP 環境をセットアップする必要があります。

以下のソフトウェアがシステムにインストールされている必要があります。

- Apache HTTP Server
- 次に示すいずれかのクライアント・タイプ: IBM Data Server Driver Package、IBM Data Server Client、または IBM Data Server Driver for ODBC and CLI

以下の手順では、プリコンパイルされたバイナリー・バージョンの PHP を手動でインストールし、Windows での IBM データ・サーバーのサポートを手動で使用可能にします。

Windows で PHP 環境をセットアップするには、以下のようにします。

1. <http://www.php.net> から PHP 5.2.x の zip パッケージ、および PECL モジュールの zip パッケージのコレクションの最新バージョンをダウンロードします。
2. PHP ZIP パッケージをインストール・ディレクトリーに解凍します。
3. PECL モジュール ZIP パッケージのコレクションを、PHP インストール・ディレクトリーの `%ext%` サブディレクトリーに解凍します。
4. `php.ini-recommended` ファイルのコピーを作成して、インストール・ディレクトリーに `php.ini` という名前の新しいファイルを作成します。
5. テキスト・エディターで `php.ini` ファイルを開き、以下の行を追加します。
  - PDO 拡張モジュールと `pdo_ibm` ドライバーを使用可能にするには、以下のようになります。

```
extension=php_pdo.dll
extension=php_pdo_ibm.dll
```
  - `ibm_db2` 拡張モジュールを使用可能にするには、以下のようになります。

```
extension=php_ibm_db2.dll
```
6. Apache HTTP Server 2.x を使用する場合、以下の行を `httpd.conf` ファイルに追加して、Apache HTTP Server 2.x での PHP サポートを使用可能にします。ここで、`phpdir` は PHP インストール・ディレクトリーのことを指します。

```
LoadModule php5_module 'phpdir/php5apache2.d11'  
AddType application/x-httpd-php .php  
PHPIniDir 'phpdir'
```

7. Apache HTTP Server を再始動して、変更した構成を有効にします。

注: メッセージ DB21085I または SQL09054 が出された場合、以下のことを実行できます。

- PHP を 64 ビット・モードで再ビルドします。
- `PHP_IBM_DB2_LIB` 変数と `PHP_PDO_IBM_LIB` 変数を、lib64 (デフォルト) ではなく lib32 を使用するように設定し、また、`LD_LIBRARY_PATH` を更新して lib32 を指すようにします。

これで、PHP 拡張モジュールがシステムにインストールされ、使用する準備ができました。

データ・サーバーに接続し、SQL ステートメントの実行を開始します。

## Linux または UNIX での PHP 環境のセットアップ

IBM データ・サーバーに接続し、SQL ステートメントを実行するには、その前に PHP 環境をセットアップする必要があります。

DB2 は、ibm\_db2 拡張モジュールまたは pdo\_ibm driver for the PHP Data Objects (PDO) 拡張モジュール (あるいはその両方) を使用することにより、PHP プログラミング言語で作成されたクライアント・アプリケーションのデータベース・アクセスをサポートしています。

以下のソフトウェアおよびファイルがシステムにインストールされている必要があります。

- Apache HTTP Server
- DB2 データベース開発ヘッダー・ファイルおよびライブラリー
- gcc コンパイラーおよび次に示す開発パッケージ: apache-devel、autoconf、automake、bison、flex、gcc、および libxml2-devel パッケージ
- 次に示すいずれかのクライアント・タイプ: IBM Data Server Driver Package、IBM Data Server Client、または IBM Data Server Driver for ODBC and CLI

この手順では、Linux または UNIX 上の DB2 をサポートするように、PHP をソースから手動でコンパイルおよびインストールします。

Linux または UNIX で PHP 環境をセットアップするには、以下のようになります。

1. <http://www.php.net> から、PHP 5.2.x または PHP 5.3.x の tarball の最新バージョンをダウンロードします。
2. 以下のコマンドを発行して、ファイルを untar します。  

```
tar -xjf php-5.x.x.tar.bz2
```
3. 新しく作成された php-5.x.x ディレクトリーに移動します。
4. `configure` コマンドを発行して、`Makefile` を構成します。カスタム・バージョンの PHP に含める機能と拡張モジュールを指定してください。標準的な `configure` コマンドには、以下のオプションが含まれます。

```
./configure --enable-cli --disable-cgi --with-apxs2=/usr/sbin/apxs2
--with-zlib --with-pdo-ibm=<sqlib>
```

configure のオプションには、以下の効果があります。

**--enable-cli**

PHP アクセスのコマンド行モードを使用可能にします。

**--disable-cgi**

PHP アクセスの CGI (Common Gateway Interface) モードを使用不可にします。

**--with-apxs2=/usr/sbin/apxs2**

PHP アクセスの Apache 2 動的共有オブジェクト (DSO) モードを使用可能にします。

**--with-zlib**

zlib 圧縮サポートを使用可能にします。

**--with-pdo-ibm=<sqlib>**

DB2 CLI・ライブラリーを使用する pdo\_ibm ドライバーがデータベース・システムにアクセスできるようにします。<sqlib> 設定は、DB2 がインストールされているディレクトリーを指します。

pdo\_ibm 拡張モジュールのソース・コードが PHP ソースの下の ext/ ディレクトリーにない場合、このフラグは無効になります。

--with-pdo-ibm を使用するには、pdo\_ibm ディレクトリーが存在していなければならない、その ext/ サブディレクトリー内に pdo\_ibm のソース・コードが含まれていなければならない。

5. make コマンドを発行して、ファイルをコンパイルします。
6. make install コマンドを発行して、ファイルをインストールします。  
configure コマンドを使用して PHP インストール・ディレクトリーを構成した方法によっては、このコマンドを正常に実行するために root 権限が必要となる場合があります。これにより、実行可能ファイルがインストールされ、PHP をサポートするように Apache HTTP Server 構成が更新されます。
7. root 権限を持つユーザーとして以下のコマンドを発行して、ibm\_db2 拡張モジュールをインストールします。

```
pec1 installibm_db2
```

このコマンドにより、PHP 用の ibm\_db2 拡張モジュールのダウンロード、構成、コンパイル、およびインストールが行われます。最新の拡張モジュールを使用することが勧められています。しかし、ibm\_db2 拡張モジュールを使用することもできます。それは、DB2 製品の一部として含まれています。

8. php.ini-recommended ファイルを、新しい PHP インストールの構成ファイル・パスにコピーします。構成ファイル・パスを判別するには、php -i コマンドを発行して、php.ini キーワードを探します。ファイルを php.ini に名前変更します。
9. 新しい php.ini ファイルをテキスト・エディターで開き、以下の行を追加します。*instance* は Linux または UNIX 上の DB2 インスタンスの名前を指しています。

- pdo\_ibm 用の DB2 環境を設定するには、以下のようになります。

```
PDO_IBM.db2_instance_name=instance
```

- `ibm_db2` 拡張モジュールを使用可能にして、DB2 環境を設定するには、次のようにします。

```
extension=ibm_db2.so  
ibm_db2.instance_name=instance
```

`extension` 変数は `extension` ディレクトリー (`extension_dir` 変数で指定される) からの相対パスとして指定しなければなりません。例えば、`extension_dir` が `$HOME/usr/php/ext` であり、`extension` が `$HOME/user/sqllib/php32` の場合、この項目は `extension=../../sqllib/php32/ibm_db2_5.2.1.so` のようになります。

10. Apache HTTP Server を再始動して、変更した構成を有効にします。

注: メッセージ DB21085I または SQL09054 が出された場合、以下のことを実行できます。

- PHP を 64 ビット・モードで再ビルドします。
- `PHP_IBM_DB2_LIB` 変数と `PHP_PDO_IBM_LIB` 変数を、`lib64` (デフォルト) ではなく `lib32` を使用するように設定し、また、`LD_LIBRARY_PATH` を更新して `lib32` を指すようにします。

## PHP でのアプリケーション開発 (ibm\_db2)

`ibm_db2` 拡張モジュールは、IBM データ・サーバー・データベース内のデータにアクセスし、それを操作するために役立つさまざまな PHP 関数を提供します。この拡張モジュールには、データベースへの接続、SQL ステートメントの実行と準備、結果セットからの行のフェッチ、ストアード・プロシージャの呼び出し、エラーの処理、およびメタデータの取得を行うための関数が含まれています。

### PHP での IBM データ・サーバー・データベースへの接続 (ibm\_db2)

SQL ステートメントを発行してデータの作成、更新、削除、または検索を行うには、その前に PHP アプリケーションからデータベースに接続する必要があります。 `ibm_db2` API を使用すると、カタログ式接続または直接 TCP/IP 接続のいずれかによって IBM データ・サーバー・データベースに接続できます。パフォーマンスを改善するために、持続的な接続を作成することもできます。

`ibm_db2` 拡張モジュールを介して IBM データ・サーバー・データベースに接続する前に、システム上に PHP 環境をセットアップし、`ibm_db2` 拡張モジュールを使用可能にする必要があります。

SQL ステートメントの呼び出しに使用できる接続リソースを戻すには、以下のいずれかの接続関数を呼び出します。

表 3. `ibm_db2` の接続関数

関数	説明
<code>db2_connect</code>	非持続的な接続を作成します。



表 3. *ibm\_db2* の接続関数 (続き)

関数	説明
db2_pconnect	持続的な接続を作成します。持続的な接続は複数の PHP 要求にまたがって開いたままになります。このため、資格情報のセットが同じであれば、後続の PHP スクリプト要求で接続を再利用することができます。

これらの関数に引数として渡されるデータベース値では、カタログ済みデータベース名、または直接 TCP/IP 接続用の完全なデータベース接続ストリングを指定できます。トランザクションをコミットするタイミング、戻される列名の大/小文字の区別、カーソル・タイプなどを制御するオプションの引数を指定できます。

接続の試行が失敗した場合は、`db2_conn_error` または `db2_stmt_errormsg` 関数を呼び出すことによって診断情報を取得できます。

`db2_connect` 関数を呼び出して接続を作成する場合、以下のいずれかの状況が発生すると、PHP はデータベースへの接続を閉じます。

- 接続に対して `db2_close` 関数が呼び出された
- 接続リソースが `NULL` に設定された
- PHP スクリプトが完了した

`db2_pconnect` 関数を呼び出して接続を作成する場合、PHP は指定された接続リソースに対する `db2_close` 関数の呼び出しをすべて無視し、それ以後の PHP スクリプトに対してデータベースへの接続を常に開いたままにします。

`ibm_db2` API については、<http://www.php.net/docs.php> を参照してください。

カタログしたデータベースに接続します。

```
<?php
$database = "sample";
$user = "db2inst1";
$password = "";

$conn = db2_connect($database, $user, $password);

if ($conn) {
echo "Connection succeeded.";
db2_close($conn);
}
else {
echo "Connection failed.";
}
?>
```

接続の試行が成功した場合は、SQL ステートメントを実行する `ibm_db2` 関数の呼び出し時に接続リソースを使用できます。次に、SQL ステートメントを準備および実行します。

### PHP アプリケーションでのトラステッド・コンテキスト (`ibm_db2`):

バージョン 9.5 フィックスパック 3 以降、`ibm_db2` 拡張モジュールでは、接続ストリング・キーワードを使用してトラステッド・コンテキストをサポートしています。

トラステッド・コンテキストにより、3 層のアプリケーションをより高速で、より安全に構築できます。ユーザーの ID は、監査およびセキュリティのために必ず

保存されます。セキュア接続が必要な場合、トラステッド・コンテキストを使用すれば、新しい接続を取得する必要がないため、パフォーマンスが向上します。

## 例

トラステッド・コンテキストを使用可能にし、ユーザーを切り替えて、現行のユーザー ID を取得する

```
<?php
$database = "SAMPLE";
$hostname = "localhost";
$port = 50000;
$authID = "db2inst1";
$auth_pass = "ibmdb2";

$tc_user = "tcuser";
$tc_pass = "tcpassword";

$dsn = "DATABASE=$database;HOSTNAME=$hostname;PORT=$port;PROTOCOL=TCPIP;UID=$authID;PWD=$auth_pass";
$options = array ("trustedcontext" => DB2_TRUSTED_CONTEXT_ENABLE);

$tc_conn = db2_connect($dsn, "", "", $options);
if($tc_conn) {
    echo "Explicit Trusted Connection succeeded.\n";

    if(db2_get_option($tc_conn, "trustedcontext")) {
        $userBefore = db2_get_option($tc_conn, "trusted_user");

        //Do some work as user 1.

        //Switching to trusted user.
        $parameters = array("trusted_user" => $tc_user, "trusted_password" => $tcuser_pass);
        $res = db2_set_option ($tc_conn, $parameters, 1);

        $userAfter = db2_get_option($tc_conn, "trusted_user");
        //Do more work as trusted user.

        if($userBefore != $userAfter) {
            echo "User has been switched." . "\n";
        }
    }

    db2_close($tc_conn);
}
else {
    echo "Explicit Trusted Connection failed.\n";
}
?>
```

## PHP での SQL ステートメントの実行 (ibm\_db2)

データベースに接続した後、ibm\_db2 API で使用可能な関数を使用して、SQL ステートメントを準備および実行します。SQL ステートメントには、静的テキスト、XQuery 式、または変数入力を表すパラメーター・マーカーを含めることができます。

### PHP での単一 SQL ステートメントの実行 (ibm\_db2):

入力パラメーターを受け入れない単一の SQL ステートメントを準備および実行するには、db2\_exec 関数を使用します。通常、db2\_exec 関数は共通のインクルード・ファイルまたは基本クラスで使用されるアプリケーションのデフォルトのスキーマを設定する際に使用します。

SQL インジェクション・アタックによるセキュリティ上の脅威を防ぐには、静的ストリングから成る SQL ステートメントを実行する場合にのみ、db2\_exec 関数を



使用してください。SQL ステートメントへユーザーから入力された PHP 変数を埋め込むことが原因で、アプリケーションが SQL インジェクション・アタックの危険にさらされることがあります。

ibm\_db2 API の接続関数の 1 つを呼び出して、接続リソースを取得します。

単一の SQL ステートメントを準備および実行するには、db2\_exec 関数を呼び出して、以下の引数を渡します。

#### *connection*

db2\_connect または db2\_pconnect 関数から戻される有効なデータベース接続リソース。

#### *statement*

SQL ステートメントを含むストリング。このストリングには、XMLQUERY 関数によって呼び出される XQuery 式を含めることができます。

#### *options*

オプション: 次のようなステートメント・オプションを指定する連想配列。

#### **DB2\_ATTR\_CASE**

SQL 標準に準拠しないデータベース・システムとの互換性を保つため、このオプションは列名を大/小文字のどちらでアプリケーションに戻すかを設定します。デフォルトでは、DB2\_CASE\_NATURAL に設定され、データベースから戻される列名がそのまま戻されます。このパラメーターを、DB2\_CASE\_LOWER に設定して列名を強制的に小文字に変更したり、DB2\_CASE\_UPPER に設定して列名を強制的に大文字に変更することもできます。

#### **DB2\_ATTR\_CURSOR**

このオプションは、ibm\_db2 が結果セットに戻すカーソルの型を設定します。デフォルトでは、ibm\_db2 は前方スクロール・カーソル (DB2\_FORWARD\_ONLY) を戻します。これにより、db2\_fetch\_array、db2\_fetch\_assoc、db2\_fetch\_both、db2\_fetch\_object、または db2\_fetch\_row の各呼び出しに対する、結果セット内の次の行が戻されます。このパラメーターを DB2\_SCROLLABLE に設定して両方向スクロール・カーソルを要求することもできます。このとき、ibm\_db2 のフェッチ関数は、アクセスする結果セット内の行の絶対位置を指定する 2 番目の引数を受け入れるようになります。

関数呼び出しが成功した場合、ステートメント・リソースが戻され、この照会に関連した後続の関数呼び出しでこれを使用することができます。

関数呼び出しが失敗した場合 (False が戻された場合)、db2\_stmt\_error または db2\_stmt\_errormsg 関数を使用して、エラーに関する診断情報を取得することができます。

ibm\_db2 API については詳しくは、<http://www.php.net/docs.php> を参照してください。

例 1: 単一の SQL ステートメントの実行。

```
<?php
$conn = db2_connect("sample", "db2inst1", "");
$sql = "SELECT * FROM DEPT";
$stmt = db2_exec($conn, $sql);
db2_close($conn);
?>
```

例 2: XQuery 式の実行。

```
<?php
$xmlquery = '$doc/customerinfo/phone';
$stmt = db2_exec($conn, "select xmlquery('$xmlquery'
PASSING INFO AS ¥'doc¥') from customer");?>
```

SQL ステートメントが両方向スクロール・カーソルを使用してすでに行を選択しているか、または行を挿入、更新、削除している場合には、`db2_num_rows` 関数を呼び出してステートメントによって戻された、またはその影響を受けた行数を戻すことができます。SQL ステートメントが結果セットを戻している場合は、行のフェッチを開始できます。

### PHP での変数入力を含む SQL ステートメントの準備と実行 (ibm\_db2):

変数入力を含む SQL ステートメントを準備および実行するには、`db2_prepare`、`db2_bind_param`、および `db2_execute` 関数を使用します。ステートメントを準備することで、データ取り出し用に最適化されたアクセス・プランがデータベース・サーバーによって作成され、ステートメントの再実行時にこれを再利用できるため、パフォーマンスが向上します。

`ibm_db2` API の接続関数の 1 つを呼び出して、接続リソースを取得します。

パラメーター・マーカを含む SQL ステートメントを準備および実行するには、次のようにします。

1. `db2_prepare` 関数を呼び出して、以下の引数を渡します。

#### *connection*

`db2_connect` または `db2_pconnect` 関数から戻される有効なデータベース接続リソース。

#### *statement*

SQL ステートメントを含む文字列 (変数入力が必要とする列値または述部値のパラメーター・マーカとして疑問符 (?) を含む)。この文字列には、`XMLQUERY` 関数によって呼び出される XQuery 式を含めることができます。パラメーター・マーカは、列の値または述部の値のプレースホルダーとしてのみ使用できます。列名、表名、その他の SQL ID の代わりにパラメーター・マーカを使用するステートメントについては、SQL コンパイラーでアクセス・プランを作成することはできません。

#### *options*

オプション: 次のようなステートメント・オプションを指定する連想配列。

#### **DB2\_ATTR\_CASE**

SQL 標準に準拠しないデータベース・システムとの互換性を保つため、このオプションは列名を大/小文字のどちらでアプリケーションに戻すかを設定します。デフォルトでは、`DB2_CASE_NATURAL` に設定され、データベースから戻される列名がそのまま戻されます。このパラメーターを、`DB2_CASE_LOWER` に設定して列名を強制的に小文字に変更したり、`DB2_CASE_UPPER` に設定して列名を強制的に大文字に変更することもできます。

#### **DB2\_ATTR\_CURSOR**

このオプションは、`ibm_db2` が結果セットに戻すカーソルの型を設定し

ます。デフォルトでは、`ibm_db2` は前方スクロール・カーソル (`DB2_FORWARD_ONLY`) を戻します。これにより、`db2_fetch_array`、`db2_fetch_assoc`、`db2_fetch_both`、`db2_fetch_object`、または `db2_fetch_row` の各呼び出しに対する、結果セット内の次の行が戻されます。このパラメーターを `DB2_SCROLLABLE` に設定して両方向スクロール・カーソルを要求することもできます。このとき、`ibm_db2` のフェッチ関数は、アクセスする結果セット内の行の絶対位置を指定する 2 番目の引数を受け入れるようになります。

関数呼び出しが成功した場合、ステートメント・ハンドル・リソースが戻され、この照会に関連した後続の関数呼び出しでこれを使用することができます。

関数呼び出しが失敗した場合 (`False` が戻された場合)、`db2_stmt_error` または `db2_stmt_errormsg` 関数を使用して、エラーに関する診断情報を取得することができます。

2. オプション: SQL ストリング内のパラメーター・マーカーごとに `db2_bind_param` 関数を呼び出して、以下の引数を渡します。入力値をパラメーター・マーカーにバインドすると、それぞれの入力値が単一のパラメーターとして扱われるため、アプリケーションに対する SQL インジェクション・アタックを防止できます。

*stmt*

`db2_prepare` 関数の呼び出しによって戻される準備済みステートメント。

*parameter-number*

SQL ステートメント内でのパラメーター・マーカーの位置を表す整数。

*variable-name*

*parameter-number* で指定されるパラメーターにバインドする PHP 変数の名前を指定するストリング。

3. `db2_execute` 関数を呼び出して、以下の引数を渡します。

*stmt*

`db2_prepare` 関数によって戻される準備済みステートメント。

*parameters*

オプション: パラメーター・マーカーの代わりに使用する値を順番に含む配列。

`ibm_db2` API について詳しくは、<http://www.php.net/docs.php> を参照してください。

変数入力を含むステートメントを準備および実行します。

```
$sql = "SELECT firstme, lastname FROM employee WHERE bonus > ? AND bonus < ?";
$stmt = db2_prepare($conn, $sql);
if (!$stmt) {
    // Handle errors
}

// Explicitly bind parameters
db2_bind_param($stmt, 1, $_POST['lower']);
db2_bind_param($stmt, 2, $_POST['upper']);

db2_execute($stmt);
// Process results
```

```
// Invoke prepared statement again using dynamically bound parameters
db2_execute($stmt, array($_POST['lower'], $_POST['upper']));
```

SQL ステートメントが 1 つ以上の結果セットを戻す場合、ステートメント・リソースからの行のフェッチを開始できます。

### PHP でのラージ・オブジェクトの挿入 (ibm\_db2):

ラージ・オブジェクトをデータベースに挿入する場合に、ラージ・オブジェクトのすべてのデータを PHP スtring にロードし、INSERT ステートメントでそれを IBM データ・サーバー・データベースに渡す代わりに、PHP サーバー上のファイルから直接ラージ・オブジェクトを挿入できます。

ibm\_db2 API の接続関数の 1 つを呼び出して、接続リソースを取得します。

ラージ・オブジェクトをファイルから直接データベースに挿入するには、次のようにしてください。

1. db2\_prepare 関数を呼び出して、ラージ・オブジェクト列を表すパラメーター・マーカータを持つ INSERT ステートメントを準備します。
2. PHP 変数の値を、ラージ・オブジェクトのデータを含むファイルのパスおよび名前に設定します。パスは相対パスでも絶対パスでも構いません。また、PHP 実行可能プログラムからアクセス可能であることが必要です。
3. db2\_bind\_param 関数を呼び出して、パラメーター・マーカータを変数にバインドします。この関数の 3 番目の引数は、ファイルのパスと名前を保持する PHP 変数の名前を表す String です。4 番目の引数は DB2\_PARAM\_FILE です。これは、データをファイルから取得するように ibm\_db2 拡張モジュールに通知します。
4. db2\_execute 関数を呼び出して、INSERT ステートメントを発行します。

ラージ・オブジェクトをデータベースに挿入するには、次のようにしてください。

```
$stmt = db2_prepare($conn, "INSERT INTO animal_pictures(picture) VALUES (?");

$picture = "/opt/albums/spook/grooming.jpg";
$rc = db2_bind_param($stmt, 1, "picture", DB2_PARAM_FILE);
$rc = db2_execute($stmt);
```

## 照会結果セットの読み取り

### PHP での結果セットからの行または列のフェッチ (ibm\_db2):

1 つ以上の結果セットを戻すステートメントを実行した後、ibm\_db2 API で使用可能ないずれかの関数を使用して、それぞれの結果セット上で戻された行を繰り返して取り出します。非常に大きいデータを格納する列が結果セットに含まれる場合、過剰なメモリー使用を避けるために、列ごとにデータを取得することができます。

関連する 1 つ以上の結果セットを持つ db2\_exec または db2\_execute 関数のいずれかによって戻されるステートメント・リソースが必要です。

結果セットからデータをフェッチするには、次のようにします。

1. いずれかのフェッチ関数を呼び出すことにより、結果セットからデータをフェッチします。

表 4. *ibm\_db2* フェッチ関数

関数	説明
<code>db2_fetch_array</code>	列位置によって索引付けされた、結果セット内の 1 行を表す配列を戻します。列には 0 から順に索引が付けられます。
<code>db2_fetch_assoc</code>	列名によって索引付けされた、結果セット内の 1 行を表す配列を戻します。
<code>db2_fetch_both</code>	列の名前と位置によって索引付けされた、結果セット内の 1 行を表す配列を戻します。
<code>db2_fetch_row</code>	次の行または要求された行への結果セット・ポインターを設定します。この関数を使用して、結果セットを繰り返して取り出します。
<code>db2_fetch_object</code>	フェッチされた行の中の列を表すプロパティを持つオブジェクトを戻します。オブジェクトのプロパティは、結果セット内の列の名前にマップされます。

これらの関数は、以下の引数を受け入れます。

*stmt*

有効なステートメント・リソース。

*row\_number*

結果セットから取り出す行の番号。行番号は 1 から始まります。 `db2_exec` または `db2_prepare` 関数の呼び出し時に両方向スクロール・カーソルを要求した場合には、このオプション・パラメーターの値を指定してください。デフォルトの前方スクロール・カーソルを使用すると、フェッチ・メソッドを呼び出すたびに、結果セット内の次の行が戻されます。

- オプション: `db2_fetch_row` 関数を呼び出した場合は、結果セットから繰り返して取り出すたびに、`db2_result` 関数を呼び出すことにより、指定された列から値を取得します。行の中の列の位置を表す整数 (0 で始まる)、または列の名前を表すストリングのどちらかを渡すことによって、列を指定できます。
- 結果セットの終わりに達したことを示す `False` がフェッチ関数から戻されるまで、行のフェッチを継続します。

`ibm_db2` API について詳しくは、<http://www.php.net/docs.php> を参照してください。

例 1: `db2_fetch_object` 関数を呼び出すことにより、結果セットから行をフェッチします

```
<?php
$conn = db2_connect("sample", "db2inst1", "");
$sql = 'SELECT FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EMPNO = ?';
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, array('000010'));
while ($row = db2_fetch_object($stmt)) {
    print "Name:
    {$row->FIRSTNAME} {$row->LASTNAME}";
}
```

```

";
}
db2_close($conn);
?>

```

例 2: db2\_fetch\_row 関数を呼び出すことにより、結果セットから行をフェッチします

```

<?php
$conn = db2_connect("sample", "db2inst1", "");
$sql = 'SELECT FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EMPNO = ?';
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, array('000010'));
while (db2_fetch_row($stmt)) {
    $fname = db2_result($stmt, 0);
    $lname = db2_result($stmt, 'LASTNAME');
    print "
    Name: $fname $lname

";
}
db2_close($conn);
?>

```

例 3: db2\_fetch\_both 関数を呼び出すことにより、結果セットから行をフェッチします

```

<?php
$conn = db2_connect("sample", "db2inst1", "");
$sql = 'SELECT FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EMPNO = ?';
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, array('000010'));
while ($row = db2_fetch_both($stmt)) {
    print "
    NAME: $row[0] $row[1]

";
    print "
    NAME: " . $row['FIRSTNAME'] . " " . $row['LASTNAME'] . "

";
}
db2_close($conn);
?>

```

データベースへの接続を閉じてもよい状態になったら、db2\_close 関数を呼び出します。db2\_pconnect を使って作成された持続的な接続を閉じようと試行した場合、クローズ要求は TRUE を戻しますが、次の呼び出し元は IBM データ・サーバー・クライアント接続を引き続き使用できます。

### PHP でのラージ・オブジェクトのフェッチ (ibm\_db2):

ラージ・オブジェクトを結果セットからフェッチする場合、ラージ・オブジェクトを PHP ストリングとして処理する代わりに、ラージ・オブジェクトを PHP サーバー上のファイルに直接フェッチすることにより、システム・リソースを節約することができます。

ibm\_db2 API の接続関数の 1 つを呼び出して、接続リソースを取得します。

ラージ・オブジェクトをデータベースからファイルに直接フェッチするには、次のようにしてください。



1. ストリームを表す PHP 変数を作成します。例えば、fopen 関数への呼び出しからの戻り値を変数に割り当てます。
2. db2\_prepare 関数を呼び出すことにより、SELECT ステートメントを作成します。
3. db2\_bind\_param 関数を呼び出すことにより、ラージ・オブジェクトの出力列を、ストリームを表す PHP 変数にバインドします。この関数の 3 番目の引数は、ファイルのパスと名前を保持する PHP 変数の名前を表す文字列です。4 番目の引数は DB2\_PARAM\_FILE です。これは、データをファイルに書き込むように ibm\_db2 拡張モジュールに通知します。
4. db2\_execute 関数を呼び出すことにより、SQL ステートメントを発行します。
5. ibm\_db2\_fetch 関数を呼び出すことにより、結果セット内の次の行を取得します (例えば、db2\_fetch\_object)。

ibm\_db2 API について詳しくは、<http://www.php.net/docs.php> を参照してください。

ラージ・オブジェクトをデータベースからフェッチします。

```
$stmt = db2_prepare($conn, "SELECT name, picture FROM animal_pictures");
$picture = fopen("/opt/albums/spook/grooming.jpg", "wb");
$rc = db2_bind_param($stmt, 1, "nickname", DB2_CHAR, 32);
$rc = db2_bind_param($stmt, 2, "picture", DB2_PARAM_FILE);
$rc = db2_execute($stmt);
$rc = db2_fetch_object($stmt);
```

## PHP でのストアード・プロシージャの呼び出し (ibm\_db2)

PHP アプリケーションからストアード・プロシージャを呼び出すには、SQL CALL ステートメントを準備および実行します。呼び出されるプロシージャには、入力パラメーター (IN)、出力パラメーター (OUT)、および入出力パラメーター (INOUT) を含めることができます。

ibm\_db2 API の接続関数の 1 つを呼び出して、接続リソースを取得します。

ストアード・プロシージャを呼び出すには、次のようにしてください。

1. db2\_prepare 関数を呼び出して、以下の引数を渡します。

### *connection*

db2\_connect または db2\_pconnect から戻される有効なデータベース接続リソース。

### *statement*

SQL CALL ステートメントを含む文字列 (任意の入力パラメーターまたは出力パラメーター用のパラメーター・マーカー (?) を含む)

### *options*

オプション: 結果セットに対して戻されるカーソルのタイプを指定する連想配列。このパラメーターを使用すると、このカーソル・タイプをサポートするデータベース・サーバーで両方向スクロール・カーソルを要求することができます。デフォルトでは、前方スクロール・カーソルが戻されます。

2. CALL ステートメント内のパラメーター・マーカーごとに db2\_bind\_param 関数を呼び出して、以下の引数を渡します。

*stmt*

db2\_prepare 関数の呼び出しによって戻される準備済みステートメント。

*parameter-number*

SQL ステートメント内でのパラメーター・マーカの位置を表す整数。

*variable-name*

*parameter-number* で指定されるパラメーターにバインドする PHP 変数の名前。

*parameter-type*

入力パラメーター (DB2\_PARAM\_INPUT)、出力パラメーター (DB2\_PARAM\_OUTPUT)、入力を受け入れて出力を戻すパラメーター (DB2\_PARAM\_INPUT\_OUTPUT) のうち、どのパラメーターとして PHP 変数を SQL パラメーターにバインドするかを指定する定数。

この手順では、出力を保持する PHP 変数の名前にそれぞれのパラメーター・マーカをバインドします。

3. db2\_execute 関数を呼び出して、準備済みステートメントを引数として渡します。

ibm\_db2 API について詳しくは、<http://www.php.net/docs.php> を参照してください。

SQL CALL ステートメントを準備して実行する。

```
$sql = 'CALL match_animal(?, ?)';
$stmt = db2_prepare($conn, $sql);

$second_name = "Rickety Ride";
$weight = 0;

db2_bind_param($stmt, 1, "second_name", DB2_PARAM_INOUT);
db2_bind_param($stmt, 2, "weight", DB2_PARAM_OUT);

print "Values of bound parameters _before_ CALL:\n";
print " 1: {$second_name} 2: {$weight}\n";

db2_execute($stmt);

print "Values of bound parameters _after_ CALL:\n";
print " 1: {$second_name} 2: {$weight}\n";
```

プロシージャー呼び出しが 1 つ以上の結果セットを戻す場合、ステートメント・リソースからの行のフェッチを開始できます。

#### PHP でのストアード・プロシージャーの複数の結果セットの取得 (ibm\_db2):

1 回のストアード・プロシージャー呼び出しによって複数の結果セットが戻される場合、ibm\_db2 API の db2\_next\_result 関数を使用して結果セットを取得することができます。

複数の結果セットを持つ db2\_exec または db2\_execute 関数によって戻されるステートメント・リソースが必要です。

複数の結果セットを取得するには、次のようにします。



- 以下のいずれかの `ibm_db2` フェッチ関数を呼び出すことによって、プロシージャから戻される最初の結果セットから行をフェッチします。その際、ステートメント・リソースを引数として渡します。(プロシージャから戻される最初の結果セットはステートメント・リソースに関連付けられます。)

表 5. `ibm_db2` フェッチ関数

関数	説明
<code>db2_fetch_array</code>	列位置によって索引付けされた、結果セット内の 1 行を表す配列を戻します。列には 0 から順に索引が付けられます。
<code>db2_fetch_assoc</code>	列名によって索引付けされた、結果セット内の 1 行を表す配列を戻します。
<code>db2_fetch_both</code>	列の名前と位置によって索引付けされた、結果セット内の 1 行を表す配列を戻します。
<code>db2_fetch_row</code>	次の行または要求された行への結果セット・ポインターを設定します。この関数を使用して、結果セットを繰り返して取り出します。
<code>db2_fetch_object</code>	フェッチされた行の中の列を表すプロパティを持つオブジェクトを戻します。オブジェクトのプロパティは、結果セット内の列の名前にマップされます。

- 最初のステートメント・リソースを `db2_next_result` 関数の最初の引数として渡すことにより、後続の結果セットを取得します。結果セットからフェッチできる行がなくなるまで、ステートメント・リソースから行をフェッチすることができます。

取得できる結果セットがなくなった場合、またはプロシージャが結果セットを戻さない場合に、`db2_next_result` 関数は `False` を戻します。

`ibm_db2` API については、<http://www.php.net/docs.php> を参照してください。

ストアード・プロシージャの複数の結果セットの取得

```
$stmt = db2_exec($conn, 'CALL multiResults()');

print "Fetching first result set\n";
while ($row = db2_fetch_array($stmt)) {
    // work with row
}

print "\nFetching second result set\n";
$result_2 = db2_next_result($stmt);
if ($result_2) {
    while ($row = db2_fetch_array($result_2)) {
        // work with row
    }
}

print "\nFetching third result set\n";
$result_3 = db2_next_result($stmt);
if ($result_3) {
```

```

while ($row = db2_fetch_array($result_3)) {
    // work with row
}
}

```

データベースへの接続を閉じてもよい状態になったら、`db2_close` 関数を呼び出します。 `db2_pconnect` を使って作成された持続的な接続を閉じようと試行した場合、クローズ要求は `TRUE` を戻しますが、次の呼び出し元は持続的な IBM データ・サーバー・クライアント接続を引き続き使用できます。

## PHP アプリケーションでのコミット・モード (`ibm_db2`)

接続リソースのコミット・モードを指定することにより、SQL ステートメントのグループがコミットされる方法を制御できます。 `ibm_db2` 拡張モジュールは、自動コミットと手動コミットの 2 つのコミット・モードをサポートします。

PHP でデータベース・トランザクションを制御するには、`db2_connect` 関数によって戻される通常の接続リソースを使用する必要があります。持続的な接続では、常に自動コミット・モードが使用されるからです。

### 自動コミット・モード

自動コミット・モードでは、各 SQL ステートメントはそれだけで完結するトランザクションであり、自動的にコミットされます。自動コミット・モードは、スケーラビリティが必要な Web アプリケーションのパフォーマンスを阻害するロック・エスカレーションの問題を防ぐのに役立ちます。 `ibm_db2` 拡張モジュールは、デフォルトで、すべての接続を自動コミット・モードで開きます。

`db2_autocommit($conn, DB2_AUTOCOMMIT_ON)` (ここで `conn` は有効な接続リソース) を呼び出すことにより、一旦使用不可になった自動コミット・モードをオンにすることができます。

`db2_autocommit` 関数を呼び出すと、PHP とデータベース管理システムとの追加の通信が必要となるため、PHP スクリプトのパフォーマンスに影響を受ける可能性があります。

### 手動コミット・モード

手動コミット・モードでは、`db2_commit` または `db2_rollback` 関数を呼び出したときにトランザクションが終了します。つまり、トランザクションの開始から `commit` または `rollback` 関数の呼び出しまでの間に同じ接続に対して実行されるすべてのステートメントは、単一のトランザクションとして扱われます。

手動コミット・モードは、1 つ以上の SQL ステートメントを含むトランザクションをロールバックする必要がある場合に役立ちます。複数の SQL ステートメントを 1 つのトランザクション内で発行し、明示的にトランザクションをコミットまたはロールバックせずにスクリプトが終了する場合には、`ibm_db2` 拡張モジュールはトランザクション内で実行されたすべての作業を自動的にロールバックします。

`db2_connect` オプション配列で "AUTOCOMMIT" =>

`DB2_AUTOCOMMIT_OFF` 設定を使用することにより、データベース接続の作成時に自動コミット・モードをオフにすることができます。また、

db2\_autocommit(\$conn, DB2\_AUTOCOMMIT\_OFF) (ここで *conn* は有効な接続リソース) を呼び出すことにより、既存の接続リソースの自動コミット・モードをオフにすることができます。

ibm\_db2 API について詳しくは、<http://www.php.net/docs.php> を参照してください。

## 例

db2\_commit または db2\_rollback が呼び出されたときにトランザクションを終了します。

```
$conn = db2_connect('SAMPLE', 'db2inst1', 'ibmdb2', array(
    'AUTOCOMMIT' => DB2_AUTOCOMMIT_OFF));

// Issue one or more SQL statements within the transaction
$result = db2_exec($conn, 'DELETE FROM TABLE employee');
if ($result === FALSE) {
    print '<p>Unable to complete transaction!</p>';
    db2_rollback($conn);
}
else {
    print '<p>Successfully completed transaction!</p>';
    db2_commit($conn);
}
```

## PHP アプリケーションでのエラー処理関数 (ibm\_db2)

データベースに接続しようとしたり、SQL ステートメントを発行しようとしたらすると、エラーが発生することがあります。ユーザー名またはパスワードが正しくない、表または列の名前のつづりが誤っている、SQL ステートメントが無効である、などの可能性があります。ibm\_db2 API は、こうした状態から正常に回復するのに役立つエラー処理関数を提供します。

### 接続エラー

接続試行が失敗した場合は、以下のいずれかの関数を使用して診断情報を取得します。

表 6. 接続エラーを処理するための *ibm\_db2* 関数

関数	説明
db2_conn_error	最後の接続試行で戻された SQLSTATE を取得します
db2_conn_errormsg	アプリケーション・エラー・ログに該当する記述エラー・メッセージを検索します。

### SQL エラー

SQL ステートメントを準備または実行しようとして失敗した場合、あるいは結果セットから結果をフェッチしようとして失敗した場合には、以下のいずれかの関数を使用して診断情報を取得します。

表 7. SQL エラーを処理するための *ibm\_db2* 関数

関数	説明
db2_stmt_error	SQL ステートメントを準備または実行する最後の試行、あるいは結果セットから結果をフェッチする最後の試行によって戻された SQLSTATE を取得します
db2_stmt_errormsg	アプリケーション・エラー・ログに該当する記述エラー・メッセージを検索します。

ibm\_db2 API について詳しくは、<http://www.php.net/docs.php> を参照してください。

**ヒント:** データベースから戻された SQLSTATE を直接表示することによって生じるセキュリティの脆弱性を防ぎ、Web アプリケーションでの全体的なユーザー・エクスペリエンスを向上させるために、switch 文を使用して、既知のエラー状態からの回復もしくは、カスタム・エラー・メッセージを戻します。以下に例を示します。

```
switch($this->state):
    case '22001':
        // More data than allowed for the defined column
        $message = "You entered too many characters for this value.";
        break;
```

## 例

### 例 1: 接続エラーの処理

```
$connection = db2_connect($database, $user, $password);
if (!$connection) {
    $this->state = db2_conn_error();
    return false;
}
```

### 例 2: SQL エラーの処理

```
$stmt = db2_prepare($connection, "DELETE FROM employee
WHERE firstnme = ?");
if (!$stmt) {
    $this->state = db2_stmt_error();
    return false;
}
```

### 例 3: 準備されたステートメントを実行した結果として生じる SQL エラーの処理

```
$success = db2_execute($stmt, array('Dan'));
if (!$success) {
    $this->state = db2_stmt_error($stmt);
    return $false;
}
```

## PHP でのデータベース・メタデータ取得関数 (ibm\_db2)

ibm\_db2 API に含まれる関数を使用すると、DB2 Database for Linux, UNIX, and Windows、IBM Cloudscape によって提供されるデータベースや、DB2 Connect™、DB2 for z/OS および DB2 for i を介して提供されるデータベースのメタデータを取得できます。

アプリケーションのクラス (管理インターフェースなど) によっては、任意のデータベースに含まれる構造および SQL オブジェクトを動的に反映する必要があるものがあります。データベースについてのメタデータを取得する 1 つの方法は、システム・カタログ表に対して SELECT ステートメントを直接発行することです。しかし、システム・カタログ表のスキーマは DB2 のバージョン間で異なることがあります。あるいは、DB2 Database for Linux, UNIX, and Windows 上のシステム・カタログ表のスキーマが DB2 for z/OS 上のシステム・カタログ表のスキーマと異なることがあります。これらの相違点をアプリケーション・コードで保守することに多大の労力を費やす代わりに、ibm\_db2 拡張モジュールに含まれる PHP 関数を使用して、データベース・メタデータを取得することができます。

これらの関数を呼び出す前に、PHP 環境をセットアップする必要があり、db2\_connect または db2\_pconnect 関数によって戻される接続リソースを持っている必要があります。

**重要:** メタデータ関数を呼び出すと、かなりの量のスペースが使用されます。可能であれば、後続の呼び出しで使用するために、呼び出しの結果をキャッシュに入れてください。

表 8. *ibm\_db2* メタデータ取得関数

関数	説明
db2_client_info	IBM データ・サーバー・クライアントに関する情報を含む読み取り専用オブジェクトを戻します
db2_column_privileges	表の列および関連する特権をリストする結果セットを戻します
db2_columns	表の列および関連するメタデータをリストする結果セットを戻します
db2_foreign_keys	表の外部キーをリストする結果セットを戻します
db2_primary_keys	表の主キーをリストする結果セットを戻します
db2_procedure_columns	1 つ以上のストアド・プロシージャのパラメーターをリストする結果セットを戻します
db2_procedures	データベースに登録されたストアド・プロシージャをリストする結果セットを戻します
db2_server_info	データベース管理システム・ソフトウェアと構成に関する情報を含む読み取り専用オブジェクトを戻します
db2_special_columns	表の固有の行 ID をリストする結果セットを戻します
db2_statistics	表の索引と統計をリストする結果セットを戻します
db2_table_privileges	データベース内の表および関連する特権をリストする結果セットを戻します

ほとんどの `ibm_db2` データベース・メタデータ取得関数によって戻される結果セットには、関数ごとに定義された列が含まれています。結果セットから行を取得するには、この目的のための `ibm_db2` 関数を使用してください。

`db2_client_info` および `db2_server_info` 関数は、読み取り専用プロパティを持つ単一のオブジェクトを直接戻します。これらのオブジェクトのプロパティを使用して、接続先のデータベース管理システムに応じて様々な仕方で動作するアプリケーションを作成できます。例えば、`ibm_db2` 拡張モジュールで構築された Web ベースのデータベース管理アプリケーションは、考えられるすべてのデータベース管理システムについて最小公分母の限度をエンコードするのではなく、

`db2_server_info()->MAX_COL_NAME_LEN` プロパティを使用して、接続先のデータベース管理システムでの列名の最大長に対応する最大長を持つ命名列のテキスト・フィールドを動的に表示できます。

`ibm_db2` API について詳しくは、<http://www.php.net/docs.php> を参照してください。

## 例

例 1: 表の列および関連する特権のリストを表示する

```
<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if ($conn) {
    $stmt = db2_column_privileges($conn, NULL, NULL, 'DEPARTMENT');
    $row = db2_fetch_array($stmt);
    print $row[2] . "\n";
    print $row[3] . "\n";
    print $row[7];
    db2_close($conn);
}
else {
    echo db2_conn_errormsg();
    printf("Connection failed\n\n");
}
?>
```

例 2: 表の主キーのリストを表示する

```
<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if ($conn) {
    $stmt = db2_primary_keys($conn, NULL, NULL, 'DEPARTMENT');
    while ($row = db2_fetch_array($stmt)) {
        echo "TABLE_NAME:¥t" . $row[2] . "\n";
        echo "COLUMN_NAME:¥t" . $row[3] . "\n";
        echo "KEY_SEQ:¥t" . $row[4] . "\n";
    }

    db2_close($conn);
}
else {
    echo db2_conn_errormsg();
    printf("Connection failed\n\n");
}
?>
```

例 3: 1 つ以上のストアド・プロシージャのパラメーターのリストを表示する

```

<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if ($conn) {
    $stmt = db2_procedures($conn, NULL, 'SYS%', '%');

    $row = db2_fetch_assoc($stmt);
    var_dump($row);

    db2_close($conn);
}
else {
    echo "Connection failed.\n";
}
?>

```

例 4: 表の索引と統計のリストを表示する

```

<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if ($conn) {
    echo "Test DEPARTMENT table:\n";
    $result = db2_statistics($conn, NULL, NULL, "EMPLOYEE", 1);
    while ($row = db2_fetch_assoc($result)) {
        var_dump($row);
    }

    echo "Test non-existent table:\n";
    $result = db2_statistics($conn, NULL, NULL, "NON_EXISTENT_TABLE", 1);
    $row = db2_fetch_array($result);
    if ($row) {
        echo "Non-Empty\n";
    } else {
        echo "Empty\n";
    }

    db2_close($conn);
}
else {
    echo 'no connection: ' . db2_conn_errormsg();
}
?>

```

例 5: データベース内の表および関連する特権のリストを表示する

```

<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if ($conn) {
    $stmt = db2_table_privileges($conn, NULL, "%%", "DEPARTMENT");
    while ($row = db2_fetch_assoc($stmt)) {
        var_dump($row);
    }
    db2_close($conn);
}
else {
    echo db2_conn_errormsg();
    printf("Connection failed\n\n");
}
?>

```

## PHP でのアプリケーション開発 (PDO)

PDO\_IBM 拡張モジュールは、PHP 5.1 で導入された標準オブジェクト指向データベース・インターフェースによってデータにアクセスし、それを操作するために役



立つさまざまな PHP 関数を提供します。この拡張モジュールには、データベースへの接続、SQL ステートメントの実行と準備、結果セットからの行のフェッチ、トランザクションの管理、ストアド・プロシージャの呼び出し、エラーの処理、およびメタデータの取得を行うための関数が含まれています。

## PHP による IBM データ・サーバー・データベースへの接続 (PDO)

SQL ステートメントを発行してデータの作成、更新、削除、または検索を行うには、その前にデータベースに接続する必要があります。PHP 用の PHP Data Objects (PDO) インターフェースを使用すると、カタログ式接続または直接 TCP/IP 接続のいずれかによって IBM データ・サーバー・データベースに接続できます。パフォーマンスを改善するために、持続的な接続を作成することもできます。

システム上に PHP 5.1 以降の環境をセットアップし、PDO および PDO\_IBM 拡張モジュールを使用可能にする必要があります。

このプロシージャは、IBM データ・サーバー・データベースへの接続オブジェクトを戻します。この接続は、PDO オブジェクトを NULL に設定するか、PHP スクリプトが完了するまで開いたままです。

IBM データ・サーバー・データベースへ接続するには、以下を行います。

1. `try{}` ブロック内で PDO コンストラクターを呼び出して、データベースへの接続を作成します。DSN 値は PDO\_IBM 拡張モジュール用のものを使用します。まず `ibm:` から始め、その次にカタログしたデータベース名または TCP/IP 直接接続に使用する完全なデータベース接続ストリングのどちらかを指定します。
  - (Windows): デフォルトで、PDO\_IBM 拡張モジュールは接続プールを使用し、接続リソースをできるだけ少なくして、接続パフォーマンスを向上させます。
  - (Linux および UNIX): 持続的な接続を作成するには、`driver_options` 引数 (4 番目の引数) として `array(PDO::ATTR_PERSISTENT => TRUE)` を PDO コンストラクターに渡します。
2. オプション: PDO コンストラクターに渡す 4 番目の引数に、PDO 接続のエラー処理オプションを以下のように設定します。
  - PDO は、エラー発生時に `PDO::errorInfo()` で取得されるエラー・メッセージと `PDO::errorCode()` で取得される SQLCODE をデフォルトで設定します。このモードを明示的に要求するには、`PDO::ATTR_ERRMODE => PDO::ERRMODE_SILENT` を設定します。
  - エラーが発生したときに PHP の `E_WARNING` が出されるようにするには、エラー・メッセージと SQLCODE の設定に加えて、`PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING` を設定します。
  - エラー発生時に PHP 例外がスローされるようにするには、`PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION` を設定します。
3. `try{}` ブロックでスローされた例外を、対応する `catch {}` ブロックでキャッチします。

PDO API について詳しくは、<http://php.net/manual/en/book.pdo.php> を参照してください。

持続的な接続で IBM データ・サーバー・データベースへ接続します。



```

try {
    $connection = new PDO("ibm:SAMPLE", "db2inst1", "ibmdb2", array(
        PDO::ATTR_PERSISTENT => TRUE,
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)
    );
}
catch (Exception $e) {
    echo($e->getMessage());
}

```

次に、SQL ステートメントを準備および実行します。

## PHP での SQL ステートメントの実行 (PDO)

データベースに接続した後、PDO API で使用可能なメソッドを使用して、SQL ステートメントを準備および実行します。SQL ステートメントには、静的テキスト、または変数入力を表すパラメーター・マーカーを含めることができます。

### PHP での単一 SQL ステートメントの実行 (PDO):

入力パラメーターを受け入れない単一の SQL ステートメントを準備および実行するには、PDO::exec または PDO::query メソッドを使用します。結果セットを戻さないステートメントを実行するには、PDO::exec メソッドを使用します。1 つ以上の結果セットを戻すステートメントを実行するには、PDO::query メソッドを使用します。

**重要:** SQL インジェクション・アタックによるセキュリティー上の脅威を防ぐには、静的ストリングから成る SQL ステートメントを実行する場合にのみ、PDO::exec または PDO::query メソッドを使用してください。SQL ステートメントへユーザーから入力された PHP 変数を埋め込むことが原因で、アプリケーションが SQL インジェクション・アタックの危険にさらされることがあります。

PDO コンストラクターを呼び出して、接続オブジェクトを取得します。

入力パラメーターを受け入れない単一の SQL ステートメントを準備および実行するには、以下のいずれかのメソッドを呼び出します。

- 結果セットを戻さない SQL ステートメントを実行するには、PDO 接続オブジェクトに対して PDO::exec メソッドを呼び出します。その際、SQL ステートメントを含むストリングを渡します。例えば、PDO::exec の典型的な使用法は、共通のインクルード・ファイルまたは基本クラスでアプリケーションのデフォルト・スキーマを設定するというものです。

SQL ステートメントが成功した場合 (行を正常に挿入、変更、または削除できた場合)、PDO::exec メソッドは挿入、変更、または削除された行数を表す整数値を返します。

PDO::exec メソッドが失敗したかどうか (FALSE または 0 を戻したかどうか) を判別するには、=== 演算子を使用することで、FALSE に関して戻り値を厳密にテストしてください。

- 1 つ以上の結果セットを戻す SQL ステートメントを実行するには、PDO 接続オブジェクトに対して PDO::query メソッドを呼び出します。その際、SQL ステートメントを含むストリングを渡します。例えば、静的 SELECT ステートメントを実行するためにこのメソッドを呼び出すことができます。

メソッドの呼び出しが成功すると PDOStatement リソースが戻され、後続のメソッド呼び出しでこれを使用することができます。

メソッド呼び出しが失敗した場合 (FALSE が戻された場合)、PDO::errorCode および PDO::errorInfo メソッドを使用して、エラーに関する診断情報を取得することができます。

PDO API について詳しくは、<http://php.net/manual/en/book.pdo.php> を参照してください。

**例 1:** アプリケーションのデフォルト・スキーマを設定するために PDO::exec メソッドを呼び出します

```
$conn = new PDO('ibm:SAMPLE', 'db2inst1', 'ibmdb2');
$result = $conn->exec('SET SCHEMA myapp');
if ($result === FALSE) {
    print "Failed to set schema: " . $conn->errorMsg();
}
```

**例 2:** SQL SELECT ステートメントを発行するために PDO::query メソッドを呼び出します

```
$conn = new PDO('ibm:SAMPLE', 'db2inst1', 'ibmdb2');
$result = $conn->query('SELECT firstnme, lastname FROM employee');
if (!$result) {
    print "<p>Could not retrieve employee list: " . $conn->errorMsg(). "</p>";
}
while ($row = $conn->fetch()) {
    print "<p>Name: {$row[0] $row[1]}</p>";
}
```

PDOStatement オブジェクトを作成するために PDO::query メソッドを呼び出した場合、PDOStatement::fetch または PDOStatement::fetchAll メソッドを呼び出すことで、オブジェクトからの行の取得を始めることができます。

### PHP での SQL ステートメントの準備と実行 (PDO):

変数入力を含む SQL ステートメントを準備および実行するには、PDO::prepare、PDOStatement::bindParam、および PDOStatement::execute メソッドを使用します。ステートメントを準備することで、データ取り出し用に最適化されたアクセス・プランがデータベース・サーバーによって作成され、ステートメントの再実行時にこれを再利用できるため、パフォーマンスが向上します。

PDO コンストラクターを呼び出して、接続オブジェクトを取得します。

パラメーター・マーカーを含む SQL ステートメントを準備および実行するには、次のようにします。

1. PDO::prepare メソッドを呼び出して、以下の引数を渡します。

#### *statement*

SQL ステートメントを含むストリング (変数入力が必要とする列値または述部値のパラメーター・マーカーとして疑問符 (?) または名前付き変数 (:name) を含む)。パラメーター・マーカーは、列の値または述部の値のプレースホルダーとしてのみ使用できます。列名、表名、その他の SQL ID の代わりにパラメーター・マーカーを使用するステートメントについては、SQL コンパイラーでアクセス・プランを作成することはできません。疑問符 (?)

パラメーター・マーカと名前付きパラメーター・マーカ (:name) の両方を同じ SQL ステートメントで使用することはできません。

#### *driver\_options*

オプション: 次のようなステートメント・オプションを格納する配列。

#### **PDO::ATTR\_CURSOR**

このオプションは、PDO が結果セットに戻すカーソルの型を設定します。デフォルトでは、PDO は前方スクロール・カーソル (PDO::CURSOR\_FWDONLY) を戻します。これにより、PDOStatement::fetch() の各呼び出しに対する、結果セット内の次の行が戻されます。両方向スクロール・カーソルを要求するには、このパラメーターを PDO::CURSOR\_SCROLL に設定できます。

関数呼び出しが成功した場合、PDOStatement オブジェクトが戻され、この照会に関連した後続のメソッド呼び出しでこれを使用することができます。

関数呼び出しが失敗した場合 (False が戻された場合)、PDO::errorCode または PDO::errorInfo メソッドを使用して、エラーに関する診断情報を取得することができます。

2. オプション: SQL ストリング内のパラメーター・マーカごとに PDOStatement::bindParam メソッドを呼び出して、以下の引数を渡します。入力値をパラメーター・マーカにバインドすると、それぞれの入力値が単一のパラメーターとして扱われるため、アプリケーションに対する SQL インジェクション・アタックを防止できます。

#### *parameter*

パラメーターの ID。疑問符パラメーター・マーカ (?) の場合、これは SQL ステートメント内のパラメーターの 1 から順に番号付けされた位置を表す整数です。名前付きパラメーター・マーカ (:name) の場合、これはパラメーター名を表すストリングです。

#### *variable*

パラメーター・マーカの代わりに使用する値

3. PDOStatement::execute メソッドを呼び出します。その際、オプションで、パラメーター・マーカの代わりに使用する値を含む配列を渡すことができます (疑問符パラメーター・マーカの場合は順番に、または名前付きパラメーター・マーカの場合は :name => value 連想配列として)。

PDO API について詳しくは、<http://php.net/manual/en/book.pdo.php> を参照してください。

変数入力を含むステートメントを準備および実行します。

```
$sql = "SELECT firstnme, lastname FROM employee WHERE bonus > ? AND bonus < ?";
$stmt = $conn->prepare($sql);
if (!$stmt) {
    // Handle errors
}

// Explicitly bind parameters
$stmt->bindParam(1, $_POST['lower']);
$stmt->bindParam(2, $_POST['upper']);

$stmt->execute($stmt);
```

```
// Invoke statement again using dynamically bound parameters
$stmt->execute($stmt, array($_POST['lower'], $_POST['upper']));
```

SQL ステートメントが 1 つ以上の結果セットを戻す場合、PDOStatement::fetch または PDOStatement::fetchAll メソッドを呼び出すことにより、ステートメント・リソースからの行のフェッチを開始できます。

### PHP でのラージ・オブジェクトの挿入 (PDO):

ラージ・オブジェクトをデータベースに挿入する場合に、ラージ・オブジェクトのすべてのデータを PHP スtring にロードし、INSERT ステートメントでそれを IBM データ・サーバー・データベースに渡す代わりに、PHP サーバー上のファイルから直接ラージ・オブジェクトを挿入できます。

PDO コンストラクターを呼び出して、接続オブジェクトを取得します。

ラージ・オブジェクトをファイルから直接データベースに挿入するには、次のようにしてください。

1. PDO::prepare メソッドを呼び出して、ラージ・オブジェクト列を表すパラメーター・マーカーを持つ INSERT ステートメントから PDOStatement オブジェクトを作成します。
2. ストリームを表す PHP 変数を作成します (例えば、fopen 関数によって戻される値を変数に割り当てることにより)。
3. PDOStatement::bindParam メソッドを呼び出して、以下の引数を渡し、ラージ・オブジェクトのデータのストリームを表す PHP 変数に、パラメーター・マーカーをバインドします。

#### *parameter*

パラメーターの ID。疑問符パラメーター・マーカー (?) の場合、これは SQL ステートメント内のパラメーターの 1 から順に番号付けされた位置を表す整数です。名前付きパラメーター・マーカー (:name) の場合、これはパラメーター名を表す String です。

#### *variable*

パラメーター・マーカーの代わりに使用する値

#### *data\_type*

PHP 定数、PDO::PARAM\_LOBは、データをファイルから取り出すように PDO 拡張モジュールに通知します。

4. PDOStatement::execute メソッドを呼び出して、INSERT ステートメントを発行します。

ラージ・オブジェクトをデータベースに挿入するには、次のようにしてください。

```
$stmt = $conn->prepare("INSERT INTO animal_pictures(picture) VALUES (?");
$picture = fopen("/opt/albums/spook/grooming.jpg", "rb");
$stmt->bindParam(1, $picture, PDO::PARAM_LOB);
$stmt->execute();
```

## 照会結果セットの読み取り

### PHP での結果セットからの行または列のフェッチ (PDO):

1 つ以上の結果セットを戻すステートメントを実行した後、PDO API で使用可能ないずれかのメソッドを使って、戻された行を繰り返して取り出します。また、PDO API に含まれるメソッドを使用して、結果セット内の 1 つ以上の行から 1 つの列をフェッチすることもできます。

関連する 1 つ以上の結果セットを持つ PDO::query または PDOStatement::execute メソッドのいずれかによって戻されるステートメント・リソースが必要です。

結果セットからデータをフェッチするには、次のようにします。

1. 以下のいずれかのフェッチ・メソッドを呼び出すことにより、結果セットからデータをフェッチします。

- 結果セットから単一の行を配列またはオブジェクトとして戻すには、PDOStatement::fetch メソッドを呼び出します。
- 結果セットからすべての行を配列の配列またはオブジェクトの配列として戻すには、PDOStatement::fetchAll メソッドを呼び出します。

デフォルトで、PDO は各行を、列名と行内の列位置によって 0 から順に番号付けされた配列として戻します。異なる様式で戻すよう要求するには、PDOStatement::fetch メソッドの呼び出し時に以下のいずれかの定数を最初のパラメーターとして指定します。

#### **PDO::FETCH\_ASSOC**

結果セット内に戻される際の列名によってインデックス付けされた配列を戻します。

#### **PDO::FETCH\_BOTH (デフォルト)**

結果セット内に戻される際の列名と 0 から順に番号付けされた列番号の両方によってインデックス付けされた配列を戻します。

#### **PDO::FETCH\_BOUND**

TRUE を返し、結果セット内の列の値を、PDOStatement::bindParam メソッドによってバインドされた PHP 変数に設定します。

#### **PDO::FETCH\_CLASS**

要求されたクラスの新規インスタンスを返し、結果セットの列をそのクラスの名前付きプロパティにマップします。

#### **PDO::FETCH\_INTO**

要求されたクラスの既存のインスタンスを更新し、結果セットの列をそのクラスの名前付きプロパティにマップします。

#### **PDO::FETCH\_LAZY**

PDO::FETCH\_BOTH と PDO::FETCH\_OBJ を結合し、アクセス時のオブジェクト変数名を作成します。

#### **PDO::FETCH\_NUM**

結果セット内に戻される際の列番号 (列 0 から始まる) によって順に番号付けされた配列を戻します。

#### **PDO::FETCH\_OBJ**

結果セット内に戻される列名に対応するプロパティ名を有する無名オブジェクトを戻します。

PDO::query または PDOStatement::execute メソッドの呼び出し時に両方向スクロール・カーソルを要求した場合は、どの行が呼び出し元に戻されるかを制御する次のようなオプション・パラメーターを渡すことができます。

- フェッチ要求のフェッチ・オリエンテーションを表す、以下のいずれかの定数：

**PDO::FETCH\_ORI\_NEXT (デフォルト)**

結果セット内の次の行をフェッチします。

**PDO::FETCH\_ORI\_PRIOR**

結果セット内の直前の行をフェッチします。

**PDO::FETCH\_ORI\_FIRST**

結果セット内の最初の行をフェッチします。

**PDO::FETCH\_ORI\_LAST**

結果セット内の最後の行をフェッチします。

**PDO::FETCH\_ORI\_ABS**

結果セット内の絶対行をフェッチします。PDOStatement::fetch メソッドの 3 番目の引数として正の整数が必要です。

**PDO::FETCH\_ORI\_REL**

結果セット内の相対行をフェッチします。PDOStatement::fetch メソッドの 3 番目の引数として正または負の整数が必要です。

- 結果セット内の絶対または相対行を要求する整数。PDOStatement::fetch メソッドの 2 番目の引数で要求されたフェッチ・オリエンテーションに対応します。
2. オプション: 以下のいずれかのメソッドを呼び出すことにより、結果セット内の 1 つ以上の行から 1 つの列をフェッチします。
- 結果セット内の単一行から単一列を戻すには、次のようにしてください。

取り出す列をメソッドの最初の引数で指定して、PDOStatement::fetchColumn メソッドを呼び出します。列の番号は 0 で始まります。列を指定しない場合、PDOStatement::fetchColumn は行の中の最初の列を戻します。

- 結果セット内にある残りの行のすべてから単一の列を含む配列を戻すには、次のようにしてください。

PDOStatement::fetchAll メソッドを呼び出します。その際、PDO::FETCH\_COLUMN 定数を最初の引数として、検索対象の列を 2 番目の引数として渡します。列の番号は 0 で始まります。列を指定しない場合、PDOStatement::fetchAll(PDO::FETCH\_COLUMN) 呼び出しによって行の中の最初の列が戻されます。

PDO API について詳しくは、<http://php.net/manual/en/book.pdo.php> を参照してください。

列番号によって索引付けされた配列を戻します。

```
$stmt = $conn->query("SELECT firstme, lastname FROM employee");
while ($row = $stmt->fetch(PDO::FETCH_NUM)) {
    print "Name: <p>{$row[0]} {$row[1]}</p>";
}
```



データベースへの接続を閉じてもよい状態になったら、PDO オブジェクトを NULL に設定します。PHP スクリプトの完了時に接続が自動的に閉じられます。

### PHP でのラージ・オブジェクトのフェッチ (PDO):

ラージ・オブジェクトを結果セットからフェッチする場合、ラージ・オブジェクトを PHP スtringとして処理する代わりに、ラージ・オブジェクトを PHP サーバー上のファイルに直接フェッチすることにより、システム・リソースを節約することができます。

PDO コンストラクターを呼び出して、接続オブジェクトを取得します。

ラージ・オブジェクトをデータベースからファイルに直接フェッチするには、次のようにしてください。

1. ストリームを表す PHP 変数を作成します。例えば、fopen 関数への呼び出しからの戻り値を変数に割り当てます。
2. PDO::prepare メソッドを呼び出すことにより、SQL ステートメントから PDOStatement オブジェクトを作成します。
3. PDOStatement::bindColumn メソッドを呼び出すことにより、ラージ・オブジェクトの出力列を、ストリームを表す PHP 変数にバインドします。2 番目の引数は、ファイルのパスと名前を保持する PHP 変数の名前を表す String です。3 番目の引数は PHP 定数、PDO::PARAM\_LOB です。これは、データをファイルに書き込むように PDO 拡張モジュールに通知します。  
PDOStatement::bindColumn メソッドを呼び出して、結果セット内の各列に異なる PHP 変数を割り当てる必要があります。
4. PDOStatement::execute メソッドを呼び出すことにより、SQL ステートメントを発行します。
5. PDOStatement::fetch(PDO::FETCH\_BOUND) を呼び出して結果セット内の次の行を取り出し、PDOStatement::bindColumn メソッドの呼び出し時に関連付けた PHP 変数にその列の出力をバインドします。

ラージ・オブジェクトをデータベースからファイルに直接フェッチするには、次のようにしてください。

```
$stmt = $conn->prepare("SELECT name, picture FROM animal_pictures");
$picture = fopen("/opt/albums/spook/grooming.jpg", "wb");
$stmt->bindColumn('NAME', $nickname, PDO::PARAM_STR, 32);
$stmt->bindColumn('PICTURE', $picture, PDO::PARAM_LOB);
$stmt->execute();
$stmt->fetch(PDO::FETCH_BOUND);
```

### PHP でのストアド・プロシージャの呼び出し (PDO)

PHP アプリケーションからストアド・プロシージャを呼び出すには、SQL CALL ステートメントを実行します。呼び出されるプロシージャには、入力パラメーター (IN)、出力パラメーター (OUT)、および入出力パラメーター (INOUT) を含めることができます。

PDO コンストラクターを呼び出して、接続オブジェクトを取得します。

このプロシージャーは、SQL CALL ステートメントを準備して実行します。詳細については、SQL ステートメントの準備および実行に関するトピックも参照してください。

ストアード・プロシージャーを呼び出すには、次のようにしてください。

1. PDO::prepare メソッドを呼び出して、OUT および INOUT パラメーターを表すパラメーター・マーカを持つ CALL ステートメントを準備します。
2. CALL ステートメントのパラメーター・マーカごとに、PDOStatement::bindParam メソッドを呼び出して、CALL ステートメントが発行された後にパラメーターの出力値を保持する PHP 変数の名前にそれぞれのパラメーター・マーカをバインドします。INOUT パラメーターの場合、PHP 変数の値は、CALL ステートメントが発行されるときにパラメーターの入力値として渡されます。
  - a. 3 番目のパラメーター *data\_type* を、バインドするデータのタイプを指定する以下の PDO::PARAM\_\* 定数のいずれかに設定します。

#### **PDO::PARAM\_NULL**

SQL NULL データ・タイプを表します。

#### **PDO::PARAM\_INT**

SQL 整数タイプを表します。

#### **PDO::PARAM\_LOB**

SQL ラージ・オブジェクト・タイプを表します。

#### **PDO::PARAM\_STR**

SQL 文字データ・タイプを表します。

INOUT パラメーターの場合、ビット単位 OR 演算子を使用して、PDO::PARAM\_INPUT\_OUTPUT を、バインドする予定のデータのタイプに付加します。

- b. 4 番目のパラメーター *length* を、出力値の期待される最大長に設定します。
3. PDOStatement::execute メソッドを呼び出して、準備済みステートメントを引数として渡します。

PDO API について詳しくは、<http://php.net/manual/en/book.pdo.php> を参照してください。

SQL CALL ステートメントを準備して実行する。

```
$sql = 'CALL match_animal(?, ?)';
$stmt = $conn->prepare($sql);

$second_name = "Rickety Ride";
$weight = 0;

$stmt->bindParam(1, $second_name, PDO::PARAM_STR|PDO::PARAM_INPUT_OUTPUT, 32);
$stmt->bindParam(2, $weight, PDO::PARAM_INT, 10);

print "Values of bound parameters _before_ CALL:¥n";
print " 1: {$second_name} 2: {$weight}¥n";

$stmt->execute();

print "Values of bound parameters _after_ CALL:¥n";
print " 1: {$second_name} 2: {$weight}¥n";
```



## PHP でのストアード・プロシージャからの複数の結果セットの取得 (PDO):

1 回のストアード・プロシージャ呼び出しによって複数の結果セットが戻される場合、PDO API の PDOStatement::nextRow メソッドを使用して結果セットを取得することができます。

PDO::query または PDOStatement::execute メソッドを使ってストアード・プロシージャを呼び出すことによって戻される PDOStatement オブジェクトが必要です。

複数の結果セットを取得するには、次のようにします。

1. 以下のいずれかの PDO フェッチ・メソッドを呼び出すことにより、プロシージャから戻された最初の結果セットから行をフェッチします。(プロシージャから戻される最初の結果セットは、CALL ステートメントによって戻される PDOStatement オブジェクトに関連付けられます。)

- 結果セットから単一の行を配列またはオブジェクトとして戻すには、PDOStatement::fetch メソッドを呼び出します。
- 結果セットからすべての行を配列の配列またはオブジェクトの配列として戻すには、PDOStatement::fetchAll メソッドを呼び出します。

最初の結果セットからフェッチできる行がなくなるまで、PDOStatement オブジェクトから行をフェッチします。

2. PDOStatement::nextRowset メソッドを呼び出して次の結果セットを戻すことにより、後続の結果セットを取得します。結果セットからフェッチできる行がなくなるまで、PDOStatement オブジェクトから行をフェッチすることができます。

取得できる結果セットがなくなった場合、またはプロシージャが結果セットを戻さなかった場合に、PDOStatement::nextRowset メソッドは False を戻します。

PDO API について詳しくは、<http://php.net/manual/en/book.pdo.php> を参照してください。

### ストアード・プロシージャの複数の結果セットの取得

```
$sql = 'CALL multiple_results()';
$stmt = $conn->query($sql);
do {
    $rows = $stmt->fetchAll(PDO::FETCH_NUM);
    if ($rows) {
        print_r($rows);
    }
} while ($stmt->nextRowset());
```

データベースへの接続を閉じてもよい状態になったら、PDO オブジェクトを NULL に設定します。PHP スクリプトの完了時に接続が自動的に閉じられます。

## PHP でのコミット・モード (PDO)

接続リソースのコミット・モードを指定することにより、SQL ステートメントのグループがコミットされる方法を制御できます。PDO 拡張モジュールは、自動コミットと手動コミットの 2 つのコミット・モードをサポートします。

### 自動コミット・モード

自動コミット・モードでは、各 SQL ステートメントはそれだけで完結するトランザクションであり、自動的にコミットされます。自動コミット・モー

ドは、スケーラビリティが必要な Web アプリケーションのパフォーマンスを阻害するロック・エスカレーションの問題を防ぐのに役立ちます。PDO 拡張モジュールは、デフォルトで、すべての接続を自動コミット・モードで開きます。

### 手動コミット・モード

手動コミット・モードでは、PDO::beginTransaction メソッドを呼び出したときにトランザクションが開始し、PDO::commit か PDO::rollBack メソッドのいずれかを呼び出したときにトランザクションが終了します。これは、トランザクションの開始から、commit または rollback メソッドの呼び出しまでの間に (同じ接続で) 実行されるステートメントは、単一のトランザクションとして扱われるという意味です。

手動コミット・モードは、1 つ以上の SQL ステートメントを含むトランザクションをロールバックする必要がある場合に役立ちます。複数の SQL ステートメントを 1 つのトランザクション内で発行し、明示的にトランザクションをコミットまたはロールバックせずにスクリプトが終了する場合、PDO はトランザクション内で実行されたすべての作業を自動的にロールバックします。

トランザクションをコミットまたはロールバックした後に、PDO はデータベース接続を自動コミット・モードに自動的にリセットします。

PDO API について詳しくは、<http://php.net/manual/en/book.pdo.php> を参照してください。

### 例

トランザクションは PDO::commit または PDO::rollBack が呼び出されたときに終了します。

```
$conn = new PDO('ibm:SAMPLE', 'db2inst1', 'ibmdb2', array(
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
// PDO::ERRMODE_EXCEPTION means an SQL error throws an exception
try {
    // Issue these SQL statements in a transaction within a try{} block
    $conn->beginTransaction();

    // One or more SQL statements

    $conn->commit();
}
catch (Exception $e) {
    // If something raised an exception in our transaction block of statements,
    // roll back any work performed in the transaction
    print '<p>Unable to complete transaction!</p>';
    $conn->rollBack();
}
```

### PHP でのエラーおよび警告の処理 (PDO)

データベースに接続しようとしたり、SQL ステートメントを発行しようとしたらすると、エラーが発生することがあります。接続用のパスワードが正しくないか、SELECT ステートメントで参照した表が存在しないか、または SQL ステートメントが無効である可能性があります。PDO は、こうした状態から正常に回復するのに役立つエラー処理メソッドを提供します。

システム上に PHP 環境をセットアップし、PDO および PDO\_IBM 拡張モジュールを使用可能にする必要があります。

PDO は、エラーを警告、エラー、または例外として扱うオプションを提供します。ただし、ユーザーが新規の PDO 接続オブジェクトを作成すると、PDO は、エラー発生時に必ず PDOException オブジェクトをスローします。この例外がキャッチされないと、PHP はエラー情報のバックトレースをプリントするので、それによって、ユーザー名やパスワードといったデータベース接続の信用証明情報が露出してしまう恐れがあります。

このプロシージャは、PDOException オブジェクトをキャッチし、関連するエラーを処理します。

1. PDOException オブジェクトをキャッチし、関連したエラーを処理するには以下のようにします。
  - a. PDO コンストラクターへの呼び出しを try ブロックに入れます。
  - b. try ブロックの後に、PDOException オブジェクトをキャッチする catch ブロックを含めます。
  - c. エラーに関連したエラー・メッセージを、PDOException オブジェクトの `Exception::getMessage` メソッドを呼び出すことによって取得します。
2. PDO または PDOStatement オブジェクトに関連した SQLSTATE を取得するには、そのオブジェクトの `errorCode` メソッドを呼び出します。
3. PDO または PDOStatement オブジェクトに関連したエラー情報の配列を取得するには、そのオブジェクトの `errorInfo` メソッドを呼び出します。配列には、最初の要素として SQLSTATE を表す文字列が、2 つ目の要素として SQL または CLI エラー・コードを表す整数が、そして 3 つ目の要素としてフルテキストのエラー・メッセージの入った文字列が含まれます。

PDO API について詳しくは、<http://php.net/manual/en/book.pdo.php> を参照してください。



---

## 第 3 章 Python アプリケーションの開発

---

### IBM データ・サーバーのための Python、SQLAlchemy、および Django Framework アプリケーション開発

Python は、迅速なアプリケーション開発に最適な、汎用の高水準スクリプト言語です。Python はコードの読みやすさを重視しており、プロシージャー、オブジェクト指向、アスペクト指向、メタプログラミング、および機能プログラミングなどのさまざまなプログラミング・パラダイムをサポートしています。Python 言語は、Python Software Foundation によって管理されています。

Python アプリケーションから IBM データ・サーバー・データベースにアクセスするために、以下の拡張モジュールが使用可能です。

#### **ibm\_db**

この API は IBM によって定義されており、高度なフィーチャーを最適な形でサポートします。SQL 照会の発行、ストアド・プロシージャーの呼び出し、pureXML の使用に加えて、メタデータ情報にもアクセスできます。

#### **ibm\_db\_dbi**

この API は、Python Database API Specification v2.0 をインプリメントします。ibm\_db\_dbi API はこの仕様に準拠しているため、ibm\_db API でサポートされるいくつかの拡張フィーチャーは提供されません。Python Database API Specification v2.0 をサポートするドライバーを備えたアプリケーションが既に存在する場合、簡単に ibm\_db に切り替えることができます。ibm\_db および ibm\_db\_dbi API は一緒にパッケージ化されています。

#### **ibm\_db\_sa**

このアダプターは、IBM データ・サーバーへの柔軟なアクセス方式を提供する SQLAlchemy をサポートします。SQLAlchemy は普及しているオープン・ソースの Python SQL ツールキットであり、オブジェクト・リレーショナル・マッパー (ORM) です。

#### **ibm\_db\_django**

このアダプターは、Django から IBM データ・サーバーへのアクセスを有効にします。Django は、高パフォーマンスの洗練された Web アプリケーションを短時間でビルドするのに使用できる、Web フレームワークとして広く使用されています。

Python アプリケーションは、以下の IBM データ・サーバーのデータベースにアクセスすることができます。

- IBM DB2 バージョン 9.1 for Linux, UNIX, and Windows、Fix Pack 2 以降
- IBM DB2 Universal Database (DB2 UDB) バージョン 8 for Linux, UNIX, and Windows、Fixpak 15 以降
- i5/OS V5R3 (PTF SI27358 を適用 (SI27250 を含む)) での IBM DB2 Universal Database へのリモート接続

- IBM i 5.4 以降 (PTF SI27256 を適用) での IBM DB2 へのリモート接続
- IBM DB2 for z/OS バージョン 8 およびバージョン 9 へのリモート接続
- IBM Informix® Dynamic Server v11.10 以降

## Python ダウンロードおよび関連リソース

IBM データ・サーバー用の Python アプリケーションを開発するために入手可能な多くのリソースがあります。

表9. Python ダウンロードおよび関連リソース

ダウンロード	
Python (Windows バイナリーを含みます。ほとんどの Linux ディストリビューションは、既にプリコンパイルされた Python に付属しています。)	<a href="http://www.python.org/download/">http://www.python.org/download/</a>
SQLAlchemy	<a href="http://www.sqlalchemy.org/download.html">http://www.sqlalchemy.org/download.html</a>
Django	<a href="http://www.djangoproject.com/download/">http://www.djangoproject.com/download/</a>
ibm_db および ibm_db_dbi 拡張機能 (ソース・コードを含む)	<a href="http://pypi.python.org/pypi/ibm_db/">http://pypi.python.org/pypi/ibm_db/</a>
	<a href="http://code.google.com/p/ibm-db/downloads/list">http://code.google.com/p/ibm-db/downloads/list</a>
SQLAlchemy 0.4 用の ibm_db_sa アダプター	<a href="http://code.google.com/p/ibm-db/downloads/list">http://code.google.com/p/ibm-db/downloads/list</a>
	<a href="http://pypi.python.org/pypi/ibm_db_sa">http://pypi.python.org/pypi/ibm_db_sa</a>
Django 1.0.x および 1.1 用 ibm_db_django アダプター	<a href="http://code.google.com/p/ibm-db/downloads/list">http://code.google.com/p/ibm-db/downloads/list</a>
	<a href="http://pypi.python.org/pypi/ibm_db_django">http://pypi.python.org/pypi/ibm_db_django</a>
setuptools プログラム	<a href="http://pypi.python.org/pypi/setuptools">http://pypi.python.org/pypi/setuptools</a>
IBM Data Server Driver Package (DS Driver)	<a href="http://www.ibm.com/software/data/support/data-server-clients/index.html">http://www.ibm.com/software/data/support/data-server-clients/index.html</a>
API 資料	
ibm_db API 資料	<a href="http://code.google.com/p/ibm-db/wiki/APIs">http://code.google.com/p/ibm-db/wiki/APIs</a>
<i>Python Database API Specification v2.0</i>	<a href="http://www.python.org/dev/peps/pep-0249/">http://www.python.org/dev/peps/pep-0249/</a>
SQLAlchemy 資料	
<i>Quick Getting Started Steps for ibm_db_sa</i>	<a href="http://code.google.com/p/ibm-db/wiki/README">http://code.google.com/p/ibm-db/wiki/README</a>
<i>SQLAlchemy 0.4 Documentation</i>	<a href="http://www.sqlalchemy.org/docs/04/index.html">http://www.sqlalchemy.org/docs/04/index.html</a>
Django のドキュメンテーション	
<i>Getting Started steps for ibm_db_django</i>	<a href="http://code.google.com/p/ibm-db/wiki/ibm_db_django_README">http://code.google.com/p/ibm-db/wiki/ibm_db_django_README</a>
<i>Django のドキュメンテーション</i>	<a href="http://www.djangoproject.com">http://www.djangoproject.com</a>
追加リソース	
Python Programming Language Web サイト	<a href="http://www.python.org/">http://www.python.org/</a>
The Python SQL Toolkit and Object Relational Mapper Web サイト	<a href="http://www.sqlalchemy.org/">http://www.sqlalchemy.org/</a>

## IBM データ・サーバー用の Python 環境のセットアップ

IBM データ・サーバーに接続し、SQL ステートメントを実行するには、その前に ibm\_db (Python)、およびオプションで ibm\_db\_sa (SQLAlchemy) または

ibm\_db\_django(Django) のパッケージをシステムにインストールすることによって、Python 環境をセットアップする必要があります。

以下のソフトウェアがシステムにインストールされている必要があります。

- Python 2.5 以降。Linux オペレーティング・システムの場合、python2.5-dev パッケージも必要になります。
- setuptools。これは <http://pypi.python.org/pypi/setuptools> で使用可能なプログラムです。このプログラムを使用して、Python パッケージのダウンロード、ビルド、インストール、アップグレード、およびアンインストールを行うことができます。
- 次に示すいずれかのクライアント・タイプ (バージョン 8 フィックスパック 14 またはバージョン 9 フィックスパック 2 以降): IBM Data Server Driver Package、IBM Data Server Client、または IBM Data Server Runtime Client。

IBM Informix Dynamic Server では、バージョン 9.5 以降のクライアントが必要です。

Python 環境をセットアップするには、以下のようにします。

1. 以下のいずれかの方法を使用して、Linux または Windows 環境をセットアップします。
  - インターネット・アクセスがある場合、以下のいずれかのコマンドを発行します。
    - ibm\_db をインストールする場合: `easy_install ibm_db`
    - ibm\_db\_sa と ibm\_db の両方をインストールする場合: `easy_install ibm_db_sa`
    - ibm\_db\_django をインストールする場合: `easy_install ibm_db_django`この手順により、setuptools がインストールされる site-packages ディレクトリの下に egg がインストールされます。
  - インターネット・アクセスがない場合、システムに適した egg ファイルを <http://code.google.com/p/ibm-db/downloads/list> からコピーし、次のコマンドを発行します。  
`easy_install egg_file_name`  
ここで、`egg_file_name` は egg ファイルへのパスです。例えば、以下のコマンドを発行します。  
`easy_install /home/user/ibm_db-xx-py2.5-linux-i386.egg`
  - [http://pypi.python.org/pypi/ibm\\_db/](http://pypi.python.org/pypi/ibm_db/) からソース・コードをダウンロードして、ドライバーをビルドし、それをインストールします。ドライバーのビルドおよびインストールに関する指示は、ドライバー・ソース・コードに含まれている README ファイルにあります。
2. **PYTHONPATH** という環境変数を作成し、以下の例に示すように `ibm_db egg` をインストールしたパスに設定します。
  - Windows オペレーティング・システムの場合:  
`PYTHONPATH=setuptools_install_path%site-packages%ibm_db-xx.egg`
  - Linux (BASH シェル) の場合: `export PYTHONPATH=setuptools_install_path/site-packages/ibm_db-xx.egg`

3. コマンド・プロンプトから、python を入力して Python インタープリターを起動し、以下の例のようなコードを入力して、セットアップをテストします。

- `ibm_db` をテストするには、以下のようにします。

```
import ibm_db
ibm_db_conn = ibm_db.connect('dsn=database', 'user', 'password')
import ibm_db_dbi
conn = ibm_db_dbi.Connection(ibm_db_conn)
conn.tables('SYSCAT', '%')
```

- `ibm_db_sa` をテストするには、以下のようにします。

```
import sqlalchemy
from sqlalchemy import *
import ibm_db_sa.ibm_db_sa
db2 = sqlalchemy.create_engine('ibm_db_sa://user:password@host.name.com:50000/database')
metadata = MetaData()
users = Table('users', metadata,
              Column('user_id', Integer, primary_key = True),
              Column('user_name', String(16), nullable = False),
              Column('email_address', String(60), key='email'),
              Column('password', String(20), nullable = False)
              )
metadata.bind = db2
metadata.create_all()
users_table = Table('users', metadata, autoload=True, autoload_with=db2)
users_table
```

- `ibm_db_django` をテストするには、以下のようにします。

- a. 新しい Django プロジェクトを作成します。

```
django-admin.py startproject myproj
```

- b. `settings.py` ファイルを編集して、DB2 へのアクセスを構成します。システム上で利用できる任意のエディターを使用してください。Unix の場合の一例を以下に示します。

```
$ cd myproj
$ vi settings.py
```

```
DATABASE_ENGINE = 'ibm_db_django'
DATABASE_NAME = 'mydb'
DATABASE_USER = 'db2inst1'
DATABASE_PASSWORD = 'ibmdb2'
DATABASE_HOST = 'localhost'
DATABASE_PORT = '50000'
```

- c. `settings.py` ファイルのタプル `INSTALLED_APPS` セクションに、以下の行を追加します。

```
'django.contrib.flatpages',
'django.contrib.redirects',
'django.contrib.comments',
'django.contrib.admin',
```

- d. テスト・スイートを実行して、正しく構成されていることを確認します。

```
python manage.py test
```

これで、Python パッケージがシステムにインストールされ、使用する準備ができました。

データ・サーバーに接続し、SQL ステートメントの発行を開始します。



## ibm\_db を使用した Python でのアプリケーション開発

ibm\_db API は、IBM データ・サーバー・データベース内のデータにアクセスし、それを操作するために役立つさまざまな Python 関数 (データベースへの接続、SQL ステートメントの準備と発行、結果セットからの行のフェッチ、ストアード・プロシージャの呼び出し、トランザクションのコミットとロールバック、エラーの処理、およびメタデータの取得を行うための関数を含む) を提供します。

### Python による IBM データ・サーバー・データベースへの接続

SQL ステートメントを実行してデータの作成、更新、削除、または検索を行うには、その前にデータベースに接続する必要があります。ibm\_db API を使用すると、カタログ式接続またはアンカタログ式接続のいずれかによりデータベースに接続できます。パフォーマンスを改善するために、持続的な接続を作成することもできます。

- Python 環境のセットアップ。
- Python スクリプトから `import ibm_db` を発行します。

SQL ステートメントの呼び出しに使用できる接続リソースを戻すには、以下のいずれかの関数を呼び出します。

表 10. *ibm\_db* 接続関数

関数	説明
<code>ibm_db.connect</code>	非持続的な接続を作成します。
<code>ibm_db.pconnect</code>	持続的な接続を作成します。持続的な接続は最初の Python スクリプト要求後に開いたままになります。このため、資格情報のセットが同じであれば、後続の Python 要求で接続を再利用することができます。

これらの関数に引数として渡されるデータベース値には、カタログしたデータベース名、または直接 TCP/IP 接続用の完全なデータベース接続ストリングを指定できます。トランザクションをコミットするタイミング、戻される列名の大/小文字の区別、カーソル・タイプなどを制御するオプションの引数を指定できます。

接続の試行が失敗した場合、`ibm_db.conn_error` または `ibm_db.conn_errormsg` 関数を呼び出すことによって診断情報を取得できます。

ibm\_db API について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

例 1: ローカル・データベースまたはカタログしたデータベースに接続します

方法 1:

```
import ibm_db
conn = ibm_db.connect("dsn=name","username","password")
```

方法 2:

```
import ibm_db
conn = ibm_db.connect("name","username","password")
```

例 2: カタログしたデータベースまたはアンカタログしたデータベースに接続します

```
import ibm_db
ibm_db.connect("DATABASE=name;HOSTNAME=host;PORT=60000;PROTOCOL=TCPIP;UID=username;
              PWD=password;", "", "")
```

接続の試行が成功した場合は、SQL ステートメントを実行する `ibm_db` 関数の呼び出し時に接続リソースを使用できます。次に、SQL ステートメントを準備および実行します。

## Python での SQL ステートメントの実行

データベースに接続した後、`ibm_db` API で使用可能な関数を使用して、SQL ステートメントを準備および実行します。SQL ステートメントには、静的テキスト、XQuery 式、または変数入力を表すパラメーター・マーカーを含めることができます。

### Python での単一 SQL ステートメントの準備と実行:

単一の SQL ステートメントを準備および実行するには、`ibm_db.exec_immediate` 関数を使用します。SQL インジェクション・アタックによるセキュリティ上の脅威を防ぐには、静的ストリングから成る SQL ステートメントを実行する場合にのみ、`ibm_db.exec_immediate` 関数を使用してください。SQL ステートメントへユーザーから入力された Python 変数を埋め込むことが原因で、アプリケーションが SQL インジェクション・アタックの危険にさらされることがあります。

`ibm_db` API の接続関数の 1 つを呼び出して、接続リソースを取得します。

単一の SQL ステートメントを準備および実行するには、`ibm_db.exec_immediate` 関数を呼び出して、以下の引数を渡します。

#### *connection*

`ibm_db.connect` または `ibm_db.pconnect` 関数から戻される有効なデータベース接続リソース。

#### *statement*

SQL ステートメントを含むストリング。このストリングには、XMLQUERY 関数によって呼び出される XQuery 式を含めることができます。

#### *options*

オプション: 結果セットに対して戻されるカーソルのタイプを指定するディクショナリー。このパラメーターを使用すると、このカーソル・タイプをサポートするデータベース・サーバーに関して両方向スクロール・カーソルを要求することができます。デフォルトでは、前方スクロール・カーソルが戻されます。

関数呼び出しが失敗した場合 (`False` が戻された場合)、`ibm_db.stmt_error` または `ibm_db.stmt_errormsg` 関数を使用して、エラーに関する診断情報を取得することができます。

関数呼び出しが成功した場合、`ibm_db.num_rows` 関数を使用して、SQL ステートメントによって戻された (またはその影響を受けた) 行数を戻すことができます。SQL ステートメントが結果セットを戻す場合は、行のフェッチを開始できます。`ibm_db` API について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

例 1: 単一 SQL ステートメントの実行

```
import ibm_db
conn = ibm_db.connect("dsn=name","username","password")
stmt = ibm_db.exec_immediate(conn, "UPDATE employee SET bonus = '1000' WHERE job = 'MANAGER'")
print "Number of affected rows: ", ibm_db.num_rows(stmt)
```

## 例 2: XQuery 式の実行

```
import ibm_db
conn = ibm_db.connect("dsn=name","username","password")
if conn:
    sql = "SELECT XMLSERIALIZE(XMLQUERY('for $i in $t/address where $i/city = \"01athe\" return <zip>
        {$i/zip/text()}</zip>' passing c.xmlcol as \"t\") AS CLOB(32k)) FROM xml_test c WHERE id = 1"
    stmt = ibm_db.exec_immediate(conn, sql)
    result = ibm_db.fetch_both(stmt)
    while( result ):
        print "Result from XMLSerialize and XMLQuery:", result[0]
        result = ibm_db.fetch_both(stmt)
```

SQL ステートメントが 1 つ以上の結果セットを戻す場合、ステートメント・リソースからの行のフェッチを開始できます。

## Python での変数入力を含む SQL ステートメントの準備と実行:

変数入力を含む SQL ステートメントを準備および実行するには、`ibm_db.prepare`、`ibm_db.bind_param`、および `ibm_db.execute` 関数を使用します。ステートメントを準備することで、データ取り出し用に最適化されたアクセス・プランがデータベース・サーバーによって作成され、ステートメントの再実行時にこれを再利用できるため、パフォーマンスが向上します。

`ibm_db` API の接続関数の 1 つを呼び出して、接続リソースを取得します。

パラメーター・マーカを含む SQL ステートメントを準備および実行するには、次のようにします。

1. `ibm_db.prepare` 関数を呼び出して、以下の引数を渡します。

### *connection*

`ibm_db.connect` または `ibm_db.pconnect` 関数から戻される有効なデータベース接続リソース。

### *statement*

SQL ステートメントを含む文字列 (変数入力を必要とする列値または述部値のパラメーター・マーカとして疑問符 (?) を含む)。この文字列には、XMLQUERY 関数によって呼び出される XQuery 式を含めることができます。

### *options*

オプション: 結果セットに対して戻されるカーソルのタイプを指定するディクショナリー。このパラメーターを使用すると、このカーソル・タイプをサポートするデータベース・サーバーに関して両方向スクロール・カーソルを要求することができます。デフォルトでは、前方スクロール・カーソルが戻されます。

関数呼び出しが成功した場合、ステートメント・ハンドル・リソースが戻され、照会に関連した後続の関数呼び出しでこれを使用することができます。

関数呼び出しが失敗した場合 (False が戻された場合)、`ibm_db.stmt_error` または `ibm_db.stmt_errormsg` 関数を使用して、エラーに関する診断情報を取得することができます。

- オプション: SQL ストリング内のパラメーター・マーカーごとに `ibm_db.bind_param` 関数を呼び出して、以下の引数を渡します。入力値をパラメーター・マーカーにバインドすると、それぞれの入力値が単一のパラメーターとして扱われるため、SQL インジェクション・アタックを防止できます。

*stmt*

`ibm_db.prepare` 関数の呼び出しによって戻される準備済みステートメント。

*parameter-number*

SQL ステートメント内でのパラメーター・マーカーの位置を表す整数。

*variable*

パラメーター・マーカーの代わりに使用する値

- `ibm_db.execute` 関数を呼び出して、以下の引数を渡します。

*stmt*

`ibm_db.prepare` から戻される準備済みステートメント。

*parameters*

準備済みステートメントに含まれるパラメーター・マーカーに一致する入力パラメーターの組。

`ibm_db API` について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

変数入力を含むステートメントを準備および実行します。

```
import ibm_db
conn = ibm_db.connect("dsn=name","username","password")
sql = "SELECT EMPNO, LASTNAME FROM EMPLOYEE WHERE EMPNO > ? AND EMPNO < ?"
stmt = ibm_db.prepare(conn, sql)
max = 50
min = 0
# Explicitly bind parameters
ibm_db.bind_param(stmt, 1, min)
ibm_db.bind_param(stmt, 2, max)
ibm_db.execute(stmt)
# Process results

# Invoke prepared statement again using dynamically bound parameters
param = max, min,
ibm_db.execute(stmt, param)
```

SQL ステートメントが 1 つ以上の結果セットを戻す場合、ステートメント・リソースからの行のフェッチを開始できます。

## Python での結果セットからの行または列のフェッチ

1 つ以上の結果セットを戻すステートメントを実行した後、`ibm_db API` で使用可能ないずれかの関数を使って、戻された行を繰り返して取り出します。非常に大きいデータ (BLOB または CLOB データなど) を格納する列が結果セットに含まれる場合、過剰なメモリー使用を避けるために、列ごとにデータを取得することができます。

関連する 1 つ以上の結果セットを持つ `ibm_db.exec_immediate` または `ibm_db.execute` 関数のいずれかによって戻されるステートメント・リソースが必要です。

結果セットからデータをフェッチするには、次のようにします。

1. いずれかのフェッチ関数を呼び出すことにより、結果セットからデータをフェッチします。

表 11. *ibm\_db* フェッチ関数

関数	説明
<code>ibm_db.fetch_tuple</code>	列位置によって索引付けされた、結果セット内の 1 行を表す組を戻します。列には 0 から順に索引が付けられます。
<code>ibm_db.fetch_assoc</code>	列名によって索引付けされた、結果セット内の 1 行を表すディクショナリーを戻します。
<code>ibm_db.fetch_both</code>	列名と位置によって索引付けされた、結果セット内の 1 行を表すディクショナリーを戻します。
<code>ibm_db.fetch_row</code>	次の行または要求された行への結果セット・ポインターを設定します。この関数を使用して、結果セットを繰り返して取り出します。

これらの関数は、以下の引数を受け入れます。

*stmt*

有効なステートメント・リソース。

*row\_number*

結果セットから取り出す行の番号。 `ibm_db.exec_immediate` または `ibm_db.prepare` 関数の呼び出し時に両方向スクロール・カーソルを要求した場合には、このパラメーターの値を指定してください。デフォルトの前方スクロール・カーソルを使用すると、フェッチ・メソッドを呼び出すたびに、結果セット内の次の行が戻されます。

2. オプション: `ibm_db.fetch_row` 関数を呼び出した場合は、結果セットから繰り返して取り出すたびに、`ibm_db.result` 関数を呼び出すことにより、指定された列から値を取得します。行の中の列の位置を表す整数 (0 で始まる)、または列の名前を表すストリングのどちらかを渡すことによって、列を指定できます。
3. 結果セットの終わりに達したことを示す `False` がフェッチ・メソッドから戻されるまで、行のフェッチを継続します。

`ibm_db` API について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

例 1: `ibm_db.fetch_both` 関数を呼び出すことにより、結果セットから行をフェッチします

```
import ibm_db

conn = ibm_db.connect( "dsn=name", "username", "password" )
sql = "SELECT * FROM EMPLOYEE"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    print "The ID is : ", dictionary["EMPNO"]
    print "The Name is : ", dictionary[1]
    dictionary = ibm_db.fetch_both(stmt)
```

例 2: `ibm_db.fetch_tuple` 関数を呼び出すことにより、結果セットから行をフェッチします

```
import ibm_db

conn = ibm_db.connect( "dsn=name", "username", "password" )
sql = "SELECT * FROM EMPLOYEE"
stmt = ibm_db.exec_immediate(conn, sql)
tuple = ibm_db.fetch_tuple(stmt)
while tuple != False:
    print "The ID is : ", tuple[0]
    print "The name is : ", tuple[1]
    tuple = ibm_db.fetch_tuple(stmt)
```

例 3: `ibm_db.fetch_assoc` 関数を呼び出すことにより、結果セットから行をフェッチします

```
import ibm_db

conn = ibm_db.connect( "dsn=name", "username", "password" )
sql = "SELECT * FROM EMPLOYEE"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
    print "The ID is : ", dictionary["EMPNO"]
    print "The name is : ", dictionary["FIRSTNAME"]
    dictionary = ibm_db.fetch_assoc(stmt)
```

例 4: 結果セットから列をフェッチします

```
import ibm_db

conn = ibm_db.connect( "dsn=name", "username", "password" )
sql = "SELECT * FROM EMPLOYEE"
stmt = ibm_db.exec_immediate(conn, sql)
while ibm_db.fetch_row(stmt) != False:
    print "The Employee number is : ", ibm_db.result(stmt, 0)
    print "The Name is : ", ibm_db.result(stmt, "NAME")
```

データベースへの接続を閉じてもよい状態になったら、`ibm_db.close` 関数を呼び出します。`ibm_db.pconnect` を使って作成された持続的な接続を閉じようと試行した場合、クローズ要求は `True` を返しますが、次の呼び出し元は引き続き接続を使用できます。

## Python でのストアード・プロシージャの呼び出し

Python アプリケーションからストアード・プロシージャを呼び出すには、SQL CALL ステートメントを準備および実行します。呼び出されるプロシージャには、入力パラメーター (IN)、出力パラメーター (OUT)、および入出力パラメーター (INOUT) を含めることができます。

`ibm_db` API の接続関数の 1 つを呼び出して、接続リソースを取得します。

ストアード・プロシージャを呼び出すには、次のようにしてください。

1. `ibm_db.prepare` 関数を呼び出して、以下の引数を渡します。

*connection*

`ibm_db.connect` または `ibm_db.pconnect` から戻される有効なデータベース接続リソース。

*statement*

SQL CALL ステートメントを含むストリング (任意の入力パラメーターまたは出力パラメーター用のパラメーター・マーカー (?) を含む)。

*options*

オプション: 結果セットに対して戻されるカーソルのタイプを指定するディクショナリー。このパラメーターを使用すると、このカーソル・タイプをサポートするデータベース・サーバーに関して両方向スクロール・カーソルを要求することができます。デフォルトでは、前方スクロール・カーソルが戻されます。

- CALL ステートメント内のパラメーター・マーカーごとに `ibm_db.bind_param` 関数を呼び出して、以下の引数を渡します。

*stmt*

`ibm_db.prepare` 関数の呼び出しによって戻される準備済みステートメント。

*parameter-number*

SQL ステートメント内でのパラメーター・マーカーの位置を表す整数。

*variable*

出力を保持する Python 変数の名前。

*parameter-type*

入力パラメーター (SQL\_PARAM\_INPUT)、出力パラメーター (SQL\_PARAM\_OUTPUT)、入力を受け入れて出力を戻すパラメーター (SQL\_PARAM\_INPUT\_OUTPUT) のうち、どのパラメーターとして Python 変数を SQL パラメーターにバインドするかを指定する定数。

この手順では、出力を保持する Python 変数の名前にそれぞれのパラメーター・マーカーをバインドします。

- `ibm_db.execute` 関数を呼び出して、準備済みステートメントを引数として渡します。

`ibm_db` API について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

SQL CALL ステートメントを準備して実行する。

```
import ibm_db

conn = ibm_db.connect("dsn=sample","username","password")
if conn:
    sql = 'CALL match_animal(?, ?, ?)'
    stmt = ibm_db.prepare(conn, sql)

    name = "Peaches"
    second_name = "Rickety Ride"
    weight = 0
    ibm_db.bind_param(stmt, 1, name, ibm_db.SQL_PARAM_INPUT)
    ibm_db.bind_param(stmt, 2, second_name, ibm_db.SQL_PARAM_INPUT_OUTPUT)
    ibm_db.bind_param(stmt, 3, weight, ibm_db.SQL_PARAM_OUTPUT)

    print "Values of bound parameters _before_ CALL:"
    print " 1: %s 2: %s 3: %d\n" % (name, second_name, weight)

    if ibm_db.execute(stmt):
        print "Values of bound parameters _after_ CALL:"
        print " 1: %s 2: %s 3: %d\n" % (name, second_name, weight)
```



プロシージャー呼び出しが 1 つ以上の結果セットを戻す場合、ステートメント・リソースからの行のフェッチを開始できます。

## Python でのストアード・プロシージャーの複数の結果セットの取得 (ibm\_db2)

1 回のストアード・プロシージャー呼び出しによって複数の結果セットが戻される場合、ibm\_db API の `ibm_db.next_result` 関数を使用して結果セットを取得することができます。

複数の結果セットを持つ `ibm_db.exec_immediate` または `ibm_db.execute` 関数によって戻されるステートメント・リソースが必要です。

複数の結果セットを取得するには、次のようにします。

1. 以下のいずれかの `ibm_db` フェッチ関数を呼び出すことによって、プロシージャーから戻される最初の結果セットから行をフェッチします。その際、ステートメント・リソースを引数として渡します。(プロシージャーから戻される最初の結果セットはステートメント・リソースに関連付けられます。)

表 12. `ibm_db` フェッチ関数

関数	説明
<code>ibm_db.fetch_tuple</code>	列位置によって索引付けされた、結果セット内の 1 行を表す組を戻します。列には 0 から順に索引が付けられます。
<code>ibm_db.fetch_assoc</code>	列名によって索引付けされた、結果セット内の 1 行を表すディクショナリーを戻します。
<code>ibm_db.fetch_both</code>	列名と位置によって索引付けされた、結果セット内の 1 行を表すディクショナリーを戻します。
<code>ibm_db.fetch_row</code>	次の行または要求された行への結果セット・ポインターを設定します。この関数を使用して、結果セットを繰り返して取り出します。

2. 最初のステートメント・リソースを `ibm_db.next_result` 関数の最初の引数として渡すことにより、後続の結果セットを取得します。結果セットからフェッチできる行がなくなるまで、ステートメント・リソースから行をフェッチすることができます。

取得できる結果セットがなくなった場合、またはプロシージャーが結果セットを戻さない場合に、`ibm_db.next_result` 関数は `False` を戻します。

`ibm_db` API について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

ストアード・プロシージャーの複数の結果セットの取得

```
import ibm_db
conn = ibm_db.connect( "dsn=sample", "user", "password" )
if conn:
    sql = 'CALL sp_multi()'
    stmt = ibm_db.exec_immediate(conn, sql)
    row = ibm_db.fetch_assoc(stmt)
```



```

while row != False :
    print "The value returned : ", row
    row = ibm_db.fetch_assoc(stmt)

stmt1 = ibm_db.next_result(stmt)
while stmt1 != False:
    row = ibm_db.fetch_assoc(stmt1)
    while row != False :
        print "The value returned : ", row
        row = ibm_db.fetch_assoc(stmt1)
    stmt1 = ibm_db.next_result(stmt)

```

データベースへの接続を閉じてもよい状態になったら、`ibm_db.close` 関数を呼び出します。`ibm_db.pconnect` を使って作成された持続的な接続を閉じようと試行した場合、クローズ要求は `True` を戻しますが、次の呼び出し元は IBM データ・サーバー・クライアント接続を引き続き使用できます。

## Python アプリケーションでのコミット・モード

接続リソースのコミット・モードを指定することにより、SQL ステートメントのグループがコミットされる方法を制御できます。`ibm_db` API は自動コミットと手動コミットの 2 つのコミット・モードをサポートしています。

### 自動コミット・モード

自動コミット・モードでは、各 SQL ステートメントはそれだけで完結するトランザクションであり、自動的にコミットされます。自動コミット・モードは、スケーラビリティが必要な Web アプリケーションのパフォーマンスを阻害するロック・エスカレーションの問題を防ぐのに役立ちます。デフォルトでは、`ibm_db` API はそれぞれの接続を自動コミット・モードで開きます。

`ibm_db.autocommit(conn, ibm_db.SQL_AUTOCOMMIT_ON)` (ここで `conn` は有効な接続リソース) を呼び出すことにより、使用不可になった後の自動コミット・モードをオンにすることができます。

`ibm_db.autocommit` 関数を呼び出すと、Python とデータベース管理システムの間の追加の通信が必要となるため、Python スクリプトのパフォーマンスが影響を受ける可能性があります。

### 手動コミット・モード

手動コミット・モードでは、`ibm_db.commit` または `ibm_db.rollback` 関数を呼び出したときにトランザクションが終了します。つまり、トランザクションの開始から `commit` または `rollback` 関数の呼び出しまでの間に同じ接続に対して実行されるすべてのステートメントは、単一のトランザクションとして扱われます。

手動コミット・モードは、1 つ以上の SQL ステートメントを含むトランザクションをロールバックする必要がある場合に役立ちます。SQL ステートメントをトランザクション内で実行し、明示的にトランザクションをコミットまたはロールバックせずにスクリプトが終了する場合には、`ibm_db` 拡張モジュールはトランザクション内で実行されたすべての作業を自動的にロールバックします。

`ibm_db.connect` または `ibm_db.pconnect` オプション配列で `{ ibm_db.SQL_ATTR_AUTOCOMMIT: ibm_db.SQL_AUTOCOMMIT_OFF }` 設定を使用することにより、データベース接続の作成時に自動コミット・モードをオフ

にすることができます。また、`ibm_db.autocommit(conn, ibm_db.SQL_AUTOCOMMIT_OFF)` (ここで `conn` は有効な接続リソース) を呼び出すことにより、接続リソースの自動コミット・モードをオフにすることができます。

`ibm_db` API について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

## 例

自動コミット・モードをオフにして、`ibm_db.commit` または `ibm_db.rollback` が呼び出されたときにトランザクションを終了します。

```
import ibm_db

array = { ibm_db.SQL_ATTR_AUTOCOMMIT : ibm_db.SQL_AUTOCOMMIT_OFF }
conn = ibm_db.pconnect("dsn=SAMPLE", "user", "password", array)
sql = "DELETE FROM EMPLOYEE"
try:
    stmt = ibm_db.exec_immediate(conn, sql)
except:
    print "Transaction couldn't be completed."
    ibm_db.rollback(conn)
else:
    ibm_db.commit(conn)
    print "Transaction complete."
```

## Python でのエラー処理関数

データベースに接続しようとしたり、SQL ステートメントを発行しようとしたりすると、エラーが発生することがあります。ユーザー名またはパスワードが正しくない、表または列の名前のつづりが誤っている、SQL ステートメントが無効である、などの可能性があります。 `ibm_db` API は、こうした状態から正常に回復するのに役立つエラー処理関数を提供します。

## 接続エラー

接続試行が失敗した場合は、以下のいずれかの関数を使用して診断情報を取得します。

表 13. 接続エラーを処理するための `ibm_db` 関数

関数	説明
<code>ibm_db.conn_error</code>	最後の接続試行で戻された <code>SQLSTATE</code> を取得します
<code>ibm_db.conn_errormsg</code>	アプリケーション・エラー・ログに該当する記述エラー・メッセージを検索します。

## SQL エラー

SQL ステートメントを準備または実行しようとして失敗した場合、あるいは結果セットから結果をフェッチしようとして失敗した場合には、以下のいずれかの関数を使用して診断情報を取得します。

表 14. SQL エラーを処理するための *ibm\_db* 関数

関数	説明
<code>ibm_db.stmt_error</code>	SQL ステートメントを準備または実行する最後の試行、あるいは結果セットから結果をフェッチする最後の試行によって戻された <code>SQLSTATE</code> を取得します
<code>ibm_db.stmt_errormsg</code>	アプリケーション・エラー・ログに該当する記述エラー・メッセージを検索します。

`ibm_db` API について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

## 例

### 例 1: 接続エラーの処理

```
import ibm_db
try:
    conn = ibm_db.connect("dsn=sample","user","password")
except:
    print "no connection:", ibm_db.conn_errormsg()
else:
    print "The connection was successful"
```

### 例 2: SQL エラーの処理

```
import ibm_db
conn = ibm_db.connect("dsn=sample", "user", "password")
sql = "DELETE FROM EMPLOYEE"
try:
    stmt = ibm_db.exec_immediate(conn, sql)
except:
    print "Transaction couldn't be completed:" , ibm_db.stmt_errormsg()
else:
    print "Transaction complete."
```

## Python でのデータベース・メタデータ取得関数

`ibm_db` API に含まれる関数を使用して、IBM データベースのメタデータを取得することができます。

これらの関数を呼び出す前に、Python 環境をセットアップして Python スクリプトで `import_db` を発行する必要があり、`ibm_db.connect` または `ibm_db.pconnect` 関数呼び出しによって接続リソースを取得する必要があります。

**重要:** メタデータ関数を呼び出すと、かなりの量のスペースが使用されます。可能であれば、後続の呼び出しで使用するために、呼び出しの結果をキャッシュに入れてください。

表 15. *ibm\_db* メタデータ取得関数

関数	説明
<code>ibm_db.client_info</code>	IBM データ・サーバー・クライアントに関する情報を含む読み取り専用オブジェクトを戻します

表 15. *ibm\_db* メタデータ取得関数 (続き)

関数	説明
<code>ibm_db.column_privileges</code>	表の列および関連する特権をリストする結果セットを返します
<code>ibm_db.columns</code>	表の列および関連するメタデータをリストする結果セットを返します
<code>ibm_db.foreign_keys</code>	表の外部キーをリストする結果セットを返します
<code>ibm_db.primary_keys</code>	表の主キーをリストする結果セットを返します
<code>ibm_db.procedure_columns</code>	1 つ以上のストアド・プロシージャのパラメーターをリストする結果セットを返します
<code>ibm_db.procedures</code>	データベースに登録されたストアド・プロシージャをリストする結果セットを返します
<code>ibm_db.server_info</code>	IBM データ・サーバーに関する情報を含む読み取り専用オブジェクトを返します
<code>ibm_db.special_columns</code>	表の固有の行 ID 列をリストする結果セットを返します
<code>ibm_db.statistics</code>	表の索引と統計をリストする結果セットを返します
<code>ibm_db.table_privileges</code>	データベース内の表および関連する特権をリストする結果セットを返します

`ibm_db` API について詳しくは、<http://code.google.com/p/ibm-db/wiki/APIs> を参照してください。

## 例

例 1: IBM データ・サーバー・クライアントに関する情報を表示します

```
import ibm_db

conn = ibm_db.connect("dsn=sample", "user", "password")
client = ibm_db.client_info(conn)

if client:
    print "DRIVER_NAME: string(%d) ¥"%s¥" % (len(client.DRIVER_NAME), client.DRIVER_NAME)
    print "DRIVER_VER: string(%d) ¥"%s¥" % (len(client.DRIVER_VER), client.DRIVER_VER)
    print "DATA_SOURCE_NAME: string(%d) ¥"%s¥" % (len(client.DATA_SOURCE_NAME), client.DATA_SOURCE_NAME)
    print "DRIVER_ODBC_VER: string(%d) ¥"%s¥" % (len(client.DRIVER_ODBC_VER), client.DRIVER_ODBC_VER)
    print "ODBC_VER: string(%d) ¥"%s¥" % (len(client.ODBC_VER), client.ODBC_VER)
    print "ODBC_SQL_CONFORMANCE: string(%d) ¥"%s¥" % (len(client.ODBC_SQL_CONFORMANCE), client.ODBC_SQL_CONFORMANCE)
    print "APPL_CODEPAGE: int(%s)" % client.APPL_CODEPAGE
    print "CONN_CODEPAGE: int(%s)" % client.CONN_CODEPAGE
    ibm_db.close(conn)
else:
    print "Error."
```

例 2: IBM データ・サーバーに関する情報を表示します

```
import ibm_db

conn = ibm_db.connect("dsn=sample", "user", "password")
server = ibm_db.server_info(conn)

if server:
    print "DBMS_NAME: string(%d) ¥"%s¥" % (len(server.DBMS_NAME), server.DBMS_NAME)
```

```
print "DBMS_VER: string(%d) ¥"%s¥" % (len(server.DBMS_VER), server.DBMS_VER)
print "DB_NAME: string(%d) ¥"%s¥" % (len(server.DB_NAME), server.DB_NAME)
ibm_db.close(conn)
else:
print "Error."
```



---

## 第 4 章 Ruby on Rails アプリケーションの開発

---

### IBM\_DB Ruby ドライバーおよび Rails アダプター

Ruby on Rails フレームワークのサポートが導入されたことにより、Rails アプリケーションが IBM データ・サーバーのデータにアクセスできるようになりました。

IBM\_DB Ruby ドライバーと Rails アダプターは、合わせて IBM\_DB gem と呼ばれ、Ruby アプリケーションはこれらによって以下のデータベース管理システムにアクセスすることができます。

- DB2 バージョン 9 for Linux, UNIX, and Windows
- DB2 Universal Database (DB2 UDB) バージョン 8 for Linux, UNIX, and Windows
- DB2 UDB for AS/400® および iSeries® バージョン 5 リリース 1 以降 (DB2 Connect 経由)
- DB2 for z/OS バージョン 8 およびバージョン 9 (DB2 Connect 経由)
- Informix Dynamic Server バージョン 11.10 以降

注: Informix Dynamic Server バージョン 11.10 にアクセスする場合、クライアント・アプリケーションは IBM Data Server Driver バージョン 9.5 以降を使用する必要があります。それより前のバージョンはサポートされていません。また、クライアント・アプリケーションは IBM Data Server Runtime Client または IBM Data Server Client を使用することもできます。

IBM\_DB Ruby ドライバーを使用して、上記の IBM データ・サーバーへの接続、およびそのデータへのアクセスを行うことができます。IBM\_DB Ruby アダプターにより、バックエンドにデータベースを使用する Rails アプリケーションが IBM データ・サーバーとインターフェースをとることができます。

IBM Ruby プロジェクトおよび RubyForge オープン・ソース・コミュニティについて詳しくは、次の Web サイトを参照してください。<http://rubyforge.org/projects/rubyibm/>

DB2 データベース製品のインストール要件のリストについては、<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.qb.server.doc/doc/r0025127.html> を参照してください。

IBM Informix Dynamic Server のインストール要件のリストについては、[http://publib.boulder.ibm.com/infocenter/idshelp/v111/topic/com.ibm.expr.doc/ids\\_in\\_004x.html](http://publib.boulder.ibm.com/infocenter/idshelp/v111/topic/com.ibm.expr.doc/ids_in_004x.html) を参照してください。

IBM Data Server Driver Package (DS Driver) のダウンロードについては、Web サイト <http://www.ibm.com/software/data/support/data-server-clients/index.html> を参照してください。

## IBM Data Servers on Rails 入門情報

IBM データ・サーバーを使用して Ruby on Rails アプリケーションの開発を始めるには、IBM データ・サーバーを使用する Rails 環境をセットアップする必要があります。始めに、フリー版の DB2 をダウンロードして、DB2 を使用した Rails アプリケーションの開発を開始することができます。

引用符で囲まれた数値を正しく処理するには、以下のいずれかのクライアント・タイプのバージョン 9.1 フィックスパック 2 以降を使用する必要があります。IBM Data Server Driver Package、IBM Data Server Client、または IBM Data Server Driver for ODBC and CLI。

環境をセットアップして IBM\_DB を開始するには、以下のようになります。

1. <http://www-306.ibm.com/software/data/servers/> から、DB2または IBM Informix Dynamic Server をダウンロードしてインストールします。
2. <http://www.ruby-lang.org/en/downloads/> から、最新のバージョンの Ruby をダウンロードしてインストールします。
3. 以下の gem インストール・コマンドを発行して、Rails gem とその従属関係をインストールします。

```
gem install rails --include-dependencies
```

これで、IBM\_DB Ruby ドライバーおよび Rails アダプターを gem としてインストールする用意ができました。必要に応じて、Rails 用の統合開発環境 (IDE) をセットアップすることもできます。

### Rails 用統合開発環境のセットアップ

Rails には、特別なファイル形式や統合開発環境 (IDE) は必要ありません。コマンド行プロンプトやテキスト・エディターを使用して始めることができます。しかし、RadRails (Eclipse 用 Rails 環境) などのように、Rails をサポートするようになった、さまざまな IDE を使用できます。

RadRails について詳しくは、<http://www.radrails.org/> を参照してください。

Ruby on Rails (RoR) 開発用に Eclipse ベースの IDE をセットアップするには、次のようになります。

1. <http://www.eclipse.org/downloads/> から Eclipse をインストールします。
2. 次の Eclipse プラグインを、その後に示す Eclipse リモート更新サイトからインストールします。
  - a. Ruby Development Tools (<http://rubyclipse.sourceforge.net/download.rdt.html>)
  - b. RubyRails IDE フィーチャー (<http://radrails.sourceforge.net/update>)
  - c. Subclipse プラグイン (<http://subclipse.tigris.org/update>)

### IBM\_DB アダプターおよびドライバーを Ruby gem としてインストールする

Ruby Gems は、Ruby ランタイム環境でのライブラリーおよびアプリケーション用の標準パッケージおよびインストール・フレームワークです。バンドルごとの単一ファイルは gem と呼ばれ、これはパッケージ形式に準拠したものです。このパッケ



ージは、中央リポジトリに配布され保管されます。これにより、同じライブラリ一またはアプリケーションの複数のバージョンを同時にデプロイすることができます。

Linux 配布版で使用されるパッケージ管理およびバンドル (.rpm、.deb) と同様、これらの gem も gem エンド・ユーザー・ユーティリティーを介して、照会、インストール、アンインストール、および操作することができます。

gem ユーティリティーでは、リモートの RubyForge 中央リポジトリの照会や、すぐに入手可能な多くのユーティリティーの検索およびインストールをシームレスに実行することができます。IBM\_DB gem がインストールされると、この機能は、以下を使用して Ruby ランタイム環境の任意のスクリプト (またはアプリケーション) からアクセスできます。

```
require 'ibm_db'
```

あるいは Windows の場合:

```
require 'mswin32/ibm_db'
```

IBM\_DB アダプターおよびドライバーを Ruby gem としてインストールするには、以下のようにします。

1. Linux、UNIX、および Mac OS X プラットフォームの場合、環境変数を設定し、オプションで DB2 プロファイルを読み込みます。
  - a. 以下のコマンドを発行して、環境変数 **IBM\_DB\_INCLUDE** および **IBM\_DB\_LIB** を設定します。

```
$ export IBM_DB_INCLUDE=DB2HOME/include
$ export IBM_DB_LIB=DB2HOME/lib
```

ここで、*DB2HOME* は、IBM データ・サーバーがインストールされているディレクトリーです。以下に例を示します。

```
$ export IBM_DB_INCLUDE=/home/db2inst1/sqllib/include
$ export IBM_DB_LIB=/home/db2inst1/sqllib/lib
```

ibm\_db 1.0.0 以前を使用する場合、IBM\_DB\_INCLUDE を設定する代わりに、環境変数 **IBM\_DB\_DIR** を *DB2HOME* に設定する必要があります。

#### 環境変数の設定の詳細:

IBM データ・サーバーがインストールされているアーキテクチャーに応じて、*DB2HOME* の下の lib ディレクトリーは、lib32 または lib64 のいずれかへのリンクになります。Ruby がコンパイルされるアーキテクチャーに従って、**IBM\_DB\_LIB** を設定できます。32 ビット・アーキテクチャーの場合、*DB2HOME* の下の lib32 ディレクトリーに **IBM\_DB\_LIB** を設定します。64 ビット・アーキテクチャーの場合、*DB2HOME* の下の lib64 ディレクトリーに **IBM\_DB\_LIB** を設定します。

- b. 以下の例に示すように、DB2 プロファイルを読み込みます。

```
$ ./home/db2inst1/sqllib/db2profile
```
2. サポートされるすべてのプラットフォームで、以下の gem コマンドを発行して IBM\_DB アダプターおよびドライバーをインストールします。

```
$ gem install ibm_db
```

選択元となる gem のリストが表示されます。以下に例を示します。

1. ibm\_db 1.0.1 (mswin32)
  2. ibm\_db 1.0.1 (ruby)
  3. ibm\_db 1.0.0 (ruby)
  4. ibm\_db 1.0.0 (mswin32)
3. Ruby gem のいずれかを選択し、固有の拡張モジュール (IBM\_DB ドライバー) をビルドして IBM\_DB gem をインストールします。

これで、IBM\_DB gem がワークステーション上にインストールされました。

以下の例は、IBM\_DB アダプターおよびドライバーを Ruby gem としてインストールする場合に使用可能なオプションを示しています。

```
$ gem install ibm_db

Select which gem to install for your platform (i686-linux)
1. ibm_db 1.0.1 (mswin32)
2. ibm_db 1.0.1 (ruby)
3. ibm_db 1.0.0 (ruby)
4. ibm_db 1.0.0 (mswin32)
...
> 2
Building native extensions. This could take a while...
Successfully installed ibm_db-1.0.1
Installing ri documentation for ibm_db-1.0.1...
Installing RDoc documentation for ibm_db-1.0.1...
```

このトピックの例では、インストールを行うバージョンの情報が含まれています。実際にインストールを実行する際には、使用可能な 2 つの最新バージョンの gem から選択することができます。

## DB2 Express-C での IBM\_DB gem のインストールの検査

DB2 Express-C での IBM\_DB gem のインストールを検証するには、データベースに接続し、SELECT ステートメントを発行した後、結果セットの最初の行をフェッチします。

IBM\_DB gem (Ruby-1.8.6 パッチ・レベル 111) を Windows または Linux オペレーティング・システムにインストールし、インストールを検査するには、以下のコマンドを使用します。コマンドの出力も示されています。

- インストールを実行するには、gem install ibm\_db のコマンドを発行します。以下に例を示します。

```
D:¥>gem install ibm_db
Select which gem to install for your platform (i386-mswin32)
1. ibm_db 1.0.1 (ruby)
2. ibm_db 1.0.1 (mswin32)
2. ibm_db 1.0.0 (ruby)
3. ibm_db 1.0.0 (mswin32)
4. Skip this gem
5. Cancel installation
> 2
Successfully installed ibm_db-1.0.0-mswin32
Installing ri documentation for ibm_db-1.0.0-mswin32...
Installing RDoc documentation for ibm_db-1.0.0-mswin32...
```

注: このトピックの例では、インストールを行うバージョンの情報が含まれています。実際にインストールを実行する際には、使用可能な 2 つの最新バージョン

の gem から選択することができます。

これで、IBM\_DB gem がマシン上にインストールされました。

- インストールを検証するには、以下のコマンドを実行してください。

IBM Informix Dynamic Server、DB2 Database for Linux, UNIX, and Windows、DB2 for IBM i、および DB2 for z/OS でのインストールを検証するには、この手順に従うことができます。DB2 Connect を使用して、DB2 for IBM i および DB2 for z/OS データ・サーバーにアクセスすることができます。

```
C:¥>irb
irb(main):001:0> require 'mwin32/ibm_db' (if using Linux based
platform then issue require 'ibm_db')
=>true
irb(main):002:0> conn = IBM_DB::connect
'devdb','username','password' (Here 'devdb' is the database cataloged in
client's database directory)
=> #<IBM_DB::Connection:0x2dddf40>
irb(main):003:0> stmt = IBM_DB::exec conn,'select * from cars'
=> #<IBM_DB::Statement:0x2beaabc>
irb(main):004:0> IBM_DB::fetch_assoc stmt (will fetch the first row of
the result set)
```

これらのコマンドが正常に実行されたら、gem が正しくインストールされています。これで、Rails アプリケーションの構築を開始することができます。

## IBM Data Servers on Rails アプリケーションでのインストールの検査

IBM\_DB ドライバーおよびアダプターが正しくインストールされたことを検証するには、IBM データ・サーバーに接続して SELECT ステートメントを発行することにより IBM\_DB ドライバー・アクセスをテストした後、サンプル Rails アプリケーションを構築および実行することにより IBM\_DB アダプター・アクセスをテストします。

インストールを検査するには、次のようにします。

1. 最新バージョンの IBM\_DB gem をインストールします。
2. IBM\_DB ドライバーのアクセスをテストします。

例えば、IBM\_DB ドライバー (および基礎となる DB2 Connect と IBM Data Server Driver for ODBC and CLI) を介した i5 データ・サーバーへのアクセスをテストするには、次のようにします。

```
D:¥ws¥RoR¥TeamRoom>irb
irb(main):001:0> require 'mwin32/ibm_db'
=> true
irb(main):002:0> conn = IBM_DB::connect 'testdb', 'user', 'pass'
=> #<IBM_DB::Connection:0x2f79d40>
irb(main):003:0> stmt = IBM_DB::exec conn, 'select * from qsys2.qsqptab1'
=> #<IBM_DB::Statement:0x2f762f8>
irb(main):004:0> IBM_DB::fetch_assoc stmt
```

3. IBM\_DB アダプターのアクセスをテストします。

IBM\_DB アダプターを介した IBM データ・サーバーへのアクセスをテストするには、以下のような手順に従ってサンプル Rails アプリケーションを構築します。

- a. 次のコマンドを発行して、新しい Rails アプリケーションを作成します。

```
C:¥>rails newapp --database=ibm_db
create
create app/controllers
create app/helpers
create app/models
create app/views/layouts
create config/environments
create config/initializers
create db
[.....]
create log/server.log
create log/production.log
create log/development.log
create log/test.log
```

- b. 新しく作成された newapp ディレクトリーに移動します。

```
C:¥>cd newapp
```

- c. database.yml ファイルを編集して、Rails アプリケーション用の接続を構成します。詳しくは、71 ページの『IBM データ・サーバーへの Rails アプリケーション接続の構成』を参照してください。

Rails 2.0 より前のバージョンを使用している場合、IBM\_DB アダプターを Rails フレームワーク内の接続アダプターのリストに登録する必要があります。これを行うには、<RubyHome>¥gems¥1.8¥gems¥activerecord-1.15.6¥lib¥active\_record.rb の 77 行目付近で、接続アダプターのリストに ibm\_db を次のように手動で追加します。

```
RAILS_CONNECTION_ADAPTERS = %w( mysql postgresql sqlite firebird
sqlserver db2 oracle sybase openbase frontbase ibm_db )
```

- d. 次のコマンドを発行して、モデルおよび骨組みを作成します。

```
C:¥>ruby script/generate scaffold Tool name:string model_num:integer
exists app/models/
exists app/controllers/
[....]
create db/migrate
create db/migrate/200807161103959_create_tools.rb
```

- e. Rails の migrate コマンドを発行して、データベース (devdb) 内に表 (tools) を作成します。

```
C:¥ >rake db:migrate
(in C:/ruby trials/newapp)
== 20080716111617 CreateTools: migrating
=====
-- create_table(:tools)
-> 0.5320s
== 20080716111617 CreateTools: migrated (0.5320s)
```

これで、Rails アプリケーションは Tools 表にアクセスして表に対する操作を実行できるようになりました。

- f. 次のコマンドを発行して、アプリケーションをテストします。

```
C:¥ruby trials¥newapp>ruby script/console
Loading development environment (Rails 2.1.0)
>> tool = Tool.new
=> #<Tool id: nil, name: nil, model_num: nil, created_at: nil,
updated_at: nil>
>> tool.name = 'chistel'
=> "chistel"
>> tool.model_num = '007'
=> "007"
>> tool.save
```

```

=> true
>> Tool.find :all
=> [#<Tool id: 100, name: "chistel", model_num: 7, created_at:
"2008-07-16 11:29:35", updated_at: "2008-07-16 11:29:35">]
>>

```

## IBM データ・サーバーへの Rails アプリケーション接続の構成

database.yml ファイル内に接続の詳細を指定することによって、Rails アプリケーション用のデータベース接続を構成します。

Rails アプリケーション用のホスト・データ・サーバー接続を構成するには、次のようにします。

rails\_application\_path¥config¥database.yml にあるデータベース構成の詳細を編集して、以下の接続属性を指定します。

```

# The IBM_DB Adapter requires the native Ruby driver (ibm_db)
# for IBM data servers (ibm_db.so).
# +config+ the hash passed as an initializer argument content:
# == mandatory parameters
# adapter: 'ibm_db' // IBM_DB Adapter name
# username: 'db2user' // data server (database) user
# password: 'secret' // data server (database) password
# database: 'DEVDB' // remote database name (or catalog entry alias)
# == optional (highly recommended for data server auditing and monitoring purposes)
# schema: 'rails123' // name space qualifier
# account: 'tester' // OS account (client workstation)
# app_user: 'test11' // authenticated application user
# application: 'rtests' // application name
# workstation: 'plato' // client workstation name
# == remote TCP/IP connection (required when no local database catalog entry available)
# host: 'Socrates' // fully qualified hostname or IP address
# port: '50000' // data server TCP/IP port number
#
# When schema is not specified, the username value is used instead.

```

注: このファイルの接続情報に対する変更は、サーバー始動時に Rails 環境が初期化されたときに適用されます。初期化後に行った変更は、作成される接続には影響しません。

IBM Informix Dynamic Server では、schema、account、app\_user、application および workstation はサポートされません。

## IBM Ruby ドライバーおよびトラステッド・コンテキスト

IBM\_DB Ruby ドライバーは、接続ストリング・キーワードを使用してトラステッド・コンテキストをサポートします。

トラステッド・コンテキストにより、3 層のアプリケーションをより高速で、より安全に構築できます。ユーザーの ID は、監査およびセキュリティーのために必ず保存されます。セキュア接続が必要な場合、新規接続を行う必要がないため、トラステッド・コンテキストはパフォーマンスを向上させます。

### 例

以下の例では、トラステッド接続を確立し、同じ接続上でユーザーを切り替えます。

```

def trusted_connection(database,hostname,port,auth_user,auth_pass,tc_user,tc_pass)
  dsn = "DATABASE=#{database};HOSTNAME=#{hostname};PORT=#{port};PROTOCOL=TCP;UID=#{auth_user};PWD=#{auth_pass};"
  conn_options = {IBM_DB::SQL_ATTR_USE_TRUSTED_CONTEXT => IBM_DB::SQL_TRUE}
  tc_options = {IBM_DB::SQL_ATTR_TRUSTED_CONTEXT_USERID => tc_user, IBM_DB::SQL_ATTR_TRUSTED_CONTEXT_PASSWORD => tc_pass}
  tc_conn = IBM_DB.connect dsn, '', '', conn_options

```

```

if tc_conn
  puts "Trusted connection established successfully."
  val = IBM_DB.get_option tc_conn, IBM_DB::SQL_ATTR_USE_TRUSTED_CONTEXT, 1
  if val
    userBefore = IBM_DB.get_option tc_conn, IBM_DB::SQL_ATTR_TRUSTED_CONTEXT_USERID, 1
    #do some work as user 1
    #....
    #....
    #switch the user
    result = IBM_DB.set_option tc_conn, tc_options, 1
    userAfter = IBM_DB.get_option tc_conn, IBM_DB::SQL_ATTR_TRUSTED_CONTEXT_USERID, 1
    if userBefore != userAfter
      puts "User has been switched."
      #do some work as user 2
      #....
      #....
    end
  end
end
IBM_DB.close tc_conn
else
  puts "Attempt to connect failed due to: #{IBM_DB.conn_errormsg}"
end
end

```

## IBM\_DB Rails アダプターの従属関係とその影響

IBM\_DB アダプター (ibm\_db\_adapter.rb) は IBM\_DB ドライバーに対して直接の従属関係にあり、IBM データ・サーバーへの接続に IBM Data Server Driver for ODBC and CLI を使用します。IBM コール・レベル・インターフェース (CLI) は、Open Database Connectivity (ODBC) に準拠した IBM データ・サーバーに対する呼び出し可能な SQL インターフェースです。

この従属関係は、IBM\_DB アダプターおよびドライバーにいくつかの影響を及ぼします。

- IBM\_DB の要件を満たす、IBM Data Server Driver for ODBC and CLI のインストールが必要です。

IBM Data Server Driver for ODBC and CLI は DB2 データベースのフル・インストールに含まれていますが、別個にこれを入手することも可能です。

**注:** IBM Data Server Driver for ODBC and CLI は以下のクライアント・パッケージに含まれています。

- IBM Data Server Client
- IBM Data Server Runtime Client
- IBM Data Server Driver Package
- CLI キーワードを使用することにより、Rails アプリケーションの外部でドライバーの動作を変更することができます。

これらの CLI キーワードを使用して、特定のトランザクション動作を Rails アプリケーションの外部で変更することができます。例えば CLI キーワードを使用して、現行スキーマを設定したり、トランザクション要素の変更 (自動コミット動作の無効化など) を行ったりすることができます。CLI キーワードの詳細情報については、以下のリンクを参照してください。

バージョン 9 の場合: <http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.apdv.cli.doc/doc/r0007964.htm>

バージョン 9.5 の場合: <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.cli.doc/doc/r0007964.html>



バージョン 9.7 の場合: <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.apdv.cli.doc/doc/r0007964.html>

- 診断情報を収集するには CLI ドライバーのトレースが必要です。

IBM\_DB ドライバー経由の要求はすべて IBM Data Server Driver for ODBC and CLI を通してインプリメントされるため、CLI トレース機能は、IBM\_DB アダプターおよびドライバーを使用するアプリケーションの問題を識別することができます。

CLI トレースでは、アプリケーションによる IBM Data Server Driver for ODBC and CLI へのすべての API 呼び出し (すべての入力パラメーターを含む) がキャプチャーされ、ドライバーからアプリケーションに戻されるすべての値がキャプチャーされます。これは、アプリケーションと IBM Data Server Driver for ODBC and CLI との対話方法をキャプチャーし、ドライバー内部の処理に関する情報を提供するインターフェース・トレースです。

## IBM\_DB Ruby ドライバーおよび Rails アダプターは JRuby でサポートされない

IBM\_DB アダプターは、JRuby ではサポートされていません。

IBM\_DB アダプターは JRuby ではサポートされていません。その理由は、JRuby Wiki の『Getting Started』で「Many Gems will work fine in JRuby, however some Gems build native C libraries as part of their install process. These Gems will not work in JRuby unless the Gem has also provided a Java™ equivalent to the native library. (多くの Gem は JRuby で正常に動作しますが、一部の Gem はインストール・プロセスの一部としてネイティブ C ライブラリーをビルドします。それらの Gem は、Java でネイティブ・ライブラリーと等価のものも提供しない限り、JRuby では動作しません。)」と説明されているとおりです。詳しくは、<http://kenai.com/projects/jruby/pages/GettingStarted> を参照してください。

IBM\_DB アダプターは、IBM\_DB Ruby ドライバー (C 拡張) と IBM Data Server Driver for ODBC and CLI を利用して、IBM データ・サーバー上のデータベースにアクセスします。また、通常の Ruby の C インプリメンテーションを使用するか、JDBC\_adapter を使用してデータベースにアクセスすることもできます。

## ActiveRecord-JDBC と IBM\_DB アダプター

IBM\_DB gem を 0.6.0 以降に更新した場合、必要とされる数値を引用符で囲む処理がいくらか異なります。

以前のバージョンのアダプターでは、さまざまなプラットフォーム上の DB2 データ・サーバーで予期される動作に適応させるために、回避策としてそのような数値を囲む引用符の使用のふるい分けを試行していましたが、新しいインプリメンテーションでは、その回避策に代わって IBM データ・サーバー・クライアントで永続的な修正が行われました。これにより、さまざまなプラットフォームで IBM データ・サーバーが使用できるようになるだけでなく、以前にはふるい分けをすり抜けていたかもしれないすべての Rails API の処理が、より信頼できるものになります。以前のバージョンのアダプターで提供されていた回避策は、Rails フレームワーク・コンポーネント (ActiveRecord) で流動的に開発されていたという性質上、かな



り不安定なものでした。置き換えられた方法でのふるい分けをいくつかの Rails API がすり抜けたことも確認されているため、ActiveRecord-JDBC アダプターで使用される回避策では、いくつかの追加の状況进行处理することが必要になる場合があります。

JRuby ランタイムでは、データ・サーバーと特定の内部対話を行うため、同じ修正を使用しても利点はありません。DB2 for IBM i は、この問題を公表しておらず (V5R3 および V5R4 で修正済み)、IBM Informix Dynamic Server についても同様です。当面は、JRuby および ActiveRecord-JDBC アダプターが完成するまで、「クラシック Ruby」(C インプリメンテーション) および IBM\_DB アダプター/ドライバを使用するのが最善の代替策です。IBM\_DB アダプターが提供していた、以前の処理をエミュレートできる、ActiveRecord-JDBC アダプターの修正を考慮することもできます。

## DB2 on Rails でのヒープ・サイズの考慮事項

DB2 上の Rails アプリケーションでは、**applheapsz** データベース構成パラメーターを 1024 より大きい値に設定する必要があります。

DB2 on Rails アプリケーションを実行するデータベースごとに、このパラメーターを設定する必要があります。以下のコマンドを使用して、**applheapsz** パラメーターを更新します。

```
db2 update db cfg for database_name using APPLHEAPSZ 1024
```

このパラメーターを使用可能にするには、DB2 インスタンスを再始動する必要があります。

---

## 付録 A. DB2 技術情報の概説

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

**注:** DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、[ibm.com](http://ibm.com) にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン ([ibm.com](http://ibm.com)) でアクセスできます。DB2 Information Management ソフトウェア・ライブラリー・サイト (<http://www.ibm.com/software/data/sw-library/>) にアクセスしてください。

### 資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、[db2docs@ca.ibm.com](mailto:db2docs@ca.ibm.com) まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

## DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、IBM Publications Center ([www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss](http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss)) から利用できる DB2 ライブラリーについて説明しています。英語および翻訳された DB2 バージョン 9.7 のマニュアル (PDF 形式) は、[www.ibm.com/support/docview.wss?rs=71&uid=swg2700947](http://www.ibm.com/support/docview.wss?rs=71&uid=swg2700947) からダウンロードできます。

この表には印刷資料が入手可能かどうかを示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: DB2 インフォメーション・センターは、PDF やハードコピー資料よりも頻繁に更新されます。

表 16. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SC88-5883-02	入手可能	2010 年 9 月
管理ルーチンおよびビュー	SC88-5880-02	入手不可	2010 年 9 月
コール・レベル・イン ターフェース ガイドお よびリファレンス 第 1 巻	SC88-5885-02	入手可能	2010 年 9 月
コール・レベル・イン ターフェース ガイドお よびリファレンス 第 2 巻	SC88-5886-02	入手可能	2010 年 9 月
コマンド・リファレン ス	SC88-5884-02	入手可能	2010 年 9 月
データ移動ユーティリ ティー ガイドおよびリ ファレンス	SC88-5903-00	入手可能	2009 年 8 月
データ・リカバリーと 高可用性 ガイドおよび リファレンス	SC88-5904-02	入手可能	2010 年 9 月
データベース: 管理の 概念および構成リファ レンス	SC88-5870-02	入手可能	2010 年 9 月
データベースのモニタ リング ガイドおよびリ ファレンス	SC88-5872-02	入手可能	2010 年 9 月
データベース・セキュ リティー・ガイド	SC88-5905-01	入手可能	2009 年 11 月

表 16. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Text Search ガイド	SC88-5902-02	入手可能	2010 年 9 月
ADO.NET および OLE DB アプリケーションの開発	SC88-5874-01	入手可能	2009 年 11 月
組み込み SQL アプリケーションの開発	SC88-5875-01	入手可能	2009 年 11 月
Java アプリケーションの開発	SC88-5878-02	入手可能	2010 年 9 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SC88-5879-01	入手不可	2010 年 9 月
SQL および外部ルーチンの開発	SC88-5876-01	入手可能	2009 年 11 月
データベース・アプリケーション開発の基礎	GI88-4201-01	入手可能	2009 年 11 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4202-00	入手可能	2009 年 8 月
グローバリゼーション・ガイド	SC88-5906-00	入手可能	2009 年 8 月
DB2 サーバー機能 インストール	GC88-5888-02	入手可能	2010 年 9 月
IBM データ・サーバー・クライアント機能 インストール	GC88-5889-01	入手不可	2010 年 9 月
メッセージ・リファレンス 第 1 巻	SC88-5897-00	入手不可	2009 年 8 月
メッセージ・リファレンス 第 2 巻	SC88-5898-00	入手不可	2009 年 8 月
Net Search Extender 管理およびユーザズ・ガイド	SC88-5901-02	入手不可	2010 年 9 月
パーティションおよびクラスタリングのガイド	SC88-5907-01	入手可能	2009 年 11 月
pureXML ガイド	SC88-5895-01	入手可能	2009 年 11 月
Query Patroller 管理およびユーザズ・ガイド	SC88-5908-00	入手不可	2009 年 8 月

表 16. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能 かどうか	最終更新
<i>Spatial Extender</i> および <i>Geodetic Data</i> <i>Management Feature</i> ユ ーザーズ・ガイドおよ びリファレンス	SC88-5900-01	入手不可	2010 年 9 月
<i>SQL</i> プロシージャ言 語: アプリケーション のイネーブルメントお よびサポート	SC88-5877-02	入手可能	2010 年 9 月
<i>SQL</i> リファレンス 第 1 巻	SC88-5881-02	入手可能	2010 年 9 月
<i>SQL</i> リファレンス 第 2 巻	SC88-5882-02	入手可能	2010 年 9 月
問題判別およびデータ ベース・パフォーマンス のチューニング	SC88-5871-02	入手可能	2010 年 9 月
<i>DB2</i> バージョン 9.7 へ のアップグレード	SC88-5887-02	入手可能	2010 年 9 月
<i>Visual Explain</i> チュー リアル	SC88-5899-00	入手不可	2009 年 8 月
<i>DB2</i> バージョン 9.7 の 新機能	SC88-5893-02	入手可能	2010 年 9 月
ワークロード・マネー ジャー ガイドおよびリ ファレンス	SC88-5894-02	入手可能	2010 年 9 月
<i>XQuery</i> リファレンス	SC88-5896-01	入手不可	2009 年 11 月

表 17. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能 かどうか	最終更新
<i>DB2 Connect Personal</i> <i>Edition</i> インストールお よび構成	SC88-5891-02	入手可能	2010 年 9 月
<i>DB2 Connect</i> サーバー 機能 インストールおよ び構成	SC88-5892-02	入手可能	2010 年 9 月
<i>DB2 Connect</i> ユーザー ズ・ガイド	SC88-5890-02	入手可能	2010 年 9 月

表 18. Information Integration の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Information Integration: フェデレーテッド・システム管理ガイド	SC88-4166-02	入手可能	2009 年 8 月
Information Integration: レプリケーションおよびイベント・パブリッシングのための ASNCLP プログラム・リファレンス	SC88-4167-04	入手可能	2009 年 8 月
Information Integration: フェデレーテッド・データ・ソース構成ガイド	SC88-4185-02	入手不可	2009 年 8 月
Information Integration: SQL レプリケーションガイドとリファレンス	SC88-4168-02	入手可能	2009 年 8 月
Information Integration: レプリケーションとイベント・パブリッシング 概説	GC88-4187-02	入手可能	2009 年 8 月

## DB2 の印刷資料の注文方法

DB2 の印刷資料が必要な場合、オンラインで購入することができますが、すべての国および地域で購入できるわけではありません。DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。DB2 PDF ドキュメンテーション DVD の一部のソフトコピー・ブックは、印刷資料では入手できないことに留意してください。例えば、「DB2 メッセージ・リファレンス」はどちらの巻も印刷資料としては入手できません。

DB2 PDF ドキュメンテーション DVD で利用できる DB2 の印刷資料の大半は、IBM に有償で注文することができます。国または地域によっては、資料を IBM Publications Center からオンラインで注文することもできます。お客様の国または地域でオンライン注文が利用できない場合、DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。DB2 PDF ドキュメンテーション DVD に収録されている資料の中には、印刷資料として提供されていないものもあります。

注: 最新で完全な DB2 資料は、DB2 インフォメーション・センター (<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7>) で参照することができます。

DB2 の印刷資料は以下の方法で注文することができます。

- 日本 IBM 発行のマニュアルはインターネット経由でご購入いただけます。詳しくは <http://www.ibm.com/shop/publications/order> をご覧ください。資料の注文情報にアクセスするには、お客様の国、地域、または言語を選択してください。その後、各ロケーションにおける注文についての指示に従ってください。

- DB2 の印刷資料を IBM 営業担当員に注文するには、以下のようになります。
  1. 以下の Web サイトのいずれかから、営業担当員の連絡先情報を見つけてください。
    - IBM Directory of world wide contacts ([www.ibm.com/planetwide](http://www.ibm.com/planetwide))
    - IBM Publications Web サイト (<http://www.ibm.com/shop/publications/order>)。国、地域、または言語を選択し、お客様の所在地に該当する Publications ホーム・ページにアクセスしてください。このページから、「このサイトについて」のリンクにアクセスしてください。
  2. 電話をご利用の場合は、DB2 資料の注文であることをご指定ください。
  3. 担当者に、注文する資料のタイトルと資料番号をお伝えください。タイトルと資料番号は、76 ページの『DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)』でご確認いただけます。

---

## コマンド行プロセッサから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果の原因になったと考えられる条件の SQLSTATE 値を戻します。SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

SQL 状態ヘルプを開始するには、コマンド行プロセッサを開いて以下のように入力します。

```
? sqlstate or ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。

例えば、? 08003 を指定すると SQL 状態 08003 のヘルプが表示され、? 08 を指定するとクラス・コード 08 のヘルプが表示されます。

---

## 異なるバージョンの DB2 インフォメーション・センターへのアクセス

DB2 バージョン 9.8 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/> です。

DB2 バージョン 9.7 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/> です。

DB2 バージョン 9.5 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/> です。

DB2 バージョン 9.1 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9/> です。

DB2 バージョン 8 のトピックについては、DB2 インフォメーション・センターの URL <http://publib.boulder.ibm.com/infocenter/db2luw/v8/> にアクセスしてください。



---

## DB2 インフォメーション・センターでの希望する言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、DB2 インフォメーション・センターでは英語でトピックが表示されます。

• Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。

1. Internet Explorer の「ツール」->「インターネット オプション」->「言語 ...」ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
  - リストに新しい言語を追加するには、「追加...」ボタンをクリックします。

注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。

- リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「上に移動」ボタンをクリックします。
3. ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。
- Firefox または Mozilla Web ブラウザーの場合に、希望する言語でトピックを表示するには、以下のようにします。

1. 「ツール」->「オプション」->「詳細」ダイアログの「言語」セクションにあるボタンを選択します。「設定」ウィンドウに「言語」パネルが表示されます。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
  - リストに新しい言語を追加するには、「追加...」ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
  - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「上に移動」ボタンをクリックします。
3. ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。

ブラウザとオペレーティング・システムの組み合わせによっては、オペレーティング・システムの地域の設定も希望のロケールと言語に変更しなければなりません。

---

## コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

ローカルにインストールされた DB2 インフォメーション・センターは、定期的に更新する必要があります。

DB2 バージョン 9.7 インフォメーション・センターが既にインストールされている必要があります。詳しくは、「DB2 サーバー機能 インストール」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

既存の DB2 インフォメーション・センターは、自動で更新することも。手動で更新することもできます。

- 自動更新 - 既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、更新中にインフォメーション・センターが使用できなくなる時間が最小限で済むというメリットもあります。さらに、自動更新は、定期的に行う他のバッチ・ジョブの一部として実行されるように設定することができます。
- 手動更新 - 更新処理中にフィーチャーまたは言語を追加する場合に使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動で更新するには、次のようにします。

1. Linux オペレーティング・システムの場合、次のようにします。
  - a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`/opt/ibm/db2ic/V9.7` ディレクトリーにインストールされています。
  - b. インストール・ディレクトリーから `doc/bin` ディレクトリーにナビゲートします。
  - c. 次のように `ic-update` スクリプトを実行します。

```
ic-update
```
2. Windows オペレーティング・システムの場合、次のようにします。
  - a. コマンド・ウィンドウを開きます。
  - b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`<Program Files>%IBM%DB2 Information Center%Version 9.7` ディレクトリーにインストールされています (`<Program Files>` は「Program Files」ディレクトリーのロケーション)。
  - c. インストール・ディレクトリーから `doc%bin` ディレクトリーにナビゲートします。

- d. 次のように `ic-update.bat` ファイルを実行します。

```
ic-update.bat
```

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、`doc\%eclipse%\configuration` ディレクトリにあります。ログ・ファイル名はランダムに生成された名前です。例えば、`1239053440785.log` のようになります。

---

## コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

ローカルにインストールされた *DB2* インフォメーション・センター を手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の *DB2* インフォメーション・センター を停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。*DB2* インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

**注:** ご使用の環境において、インターネットに接続されていないマシンに *DB2* インフォメーション・センター の更新をインストールする必要がある場合、インターネットに接続されていて *DB2* インフォメーション・センター がインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。

更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。

3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の *DB2* インフォメーション・センター を再開します。

**注:** Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、「管理者として実行」を選択します。

コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを更新するには、以下のようにします。

1. DB2 インフォメーション・センターを停止します。
  - Windows では、「スタート」→「コントロール パネル」→「管理ツール」→「サービス」をクリックします。次に、「DB2 インフォメーション・センター」サービスを右クリックして「停止」を選択します。
  - Linux では、以下のコマンドを入力します。

```
/etc/init.d/db2icdv97 stop
```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
  - Windows の場合:
    - a. コマンド・ウィンドウを開きます。
    - b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`Program_Files\IBM\DB2 Information Center\Version 9.7` ディレクトリーにインストールされています (`Program_Files` は Program Files ディレクトリーのロケーション)。
    - c. インストール・ディレクトリーから `doc\bin` ディレクトリーにナビゲートします。
    - d. 次のように `help_start.bat` ファイルを実行します。

```
help_start.bat
```
  - Linux の場合:
    - a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`/opt/ibm/db2ic/V9.7` ディレクトリーにインストールされています。
    - b. インストール・ディレクトリーから `doc/bin` ディレクトリーにナビゲートします。
    - c. 次のように `help_start` スクリプトを実行します。

```
help_start
```

システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「更新」ボタン (🔄) をクリックします。(ブラウザーで JavaScript™ が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「更新の検索 (Find Updates)」をクリックします。既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「更新のインストール」をクリックします。
5. インストール・プロセスが完了したら、「完了」をクリックします。
6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
  - Windows の場合は、インストール・ディレクトリーの `doc\bin` ディレクトリーにナビゲートしてから、次のように `help_end.bat` ファイルを実行します。

```
help_end.bat
```

注: help\_end バッチ・ファイルには、help\_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help\_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。

- Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help\_end スクリプトを実行します。

```
help_end
```

注: help\_end スクリプトには、help\_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help\_start スクリプトを停止しないでください。

#### 7. DB2 インフォメーション・センター を再開します。

- Windows では、「スタート」 → 「コントロール パネル」 → 「管理ツール」 → 「サービス」をクリックします。次に、「DB2 インフォメーション・センター」サービスを右クリックして「開始」を選択します。
- Linux では、以下のコマンドを入力します。

```
/etc/init.d/db2icdv97 start
```

更新された DB2 インフォメーション・センター に、更新された新しいトピックが表示されます。

---

## DB2 チュートリアル

DB2 チュートリアルは、DB2 製品のさまざまな機能について学習するのを支援します。この演習をとおして段階的に学習することができます。

### はじめに

インフォメーション・センター (<http://publib.boulder.ibm.com/infocenter/db2help/>) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

### DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML* ガイド」の『**pureXML**』

XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

「*Visual Explain* チュートリアル」の『**Visual Explain**』

Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

---

## DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

## DB2 ドキュメンテーション

トラブルシューティング情報は、「問題判別およびデータベース・パフォーマンスのチューニング」または *DB2* インフォメーション・センターの『データベースの基本』セクションにあります。ここでは、*DB2* 診断ツールおよびユーティリティーを使用して、問題を切り分けて識別する方法、最も頻繁に起こる幾つかの問題に対するソリューションについての情報、および *DB2* データベース製品を使用する際に発生する可能性のある問題の解決方法についての他のアドバイスがあります。

## DB2 Technical Support の Web サイト

現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、*DB2* Technical Support の Web サイトを参照してください。

Technical Support サイトには、最新の *DB2* 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

*DB2* Technical Support の Web サイト ([http://www.ibm.com/software/data/db2/support/db2\\_9/](http://www.ibm.com/software/data/db2/support/db2_9/)) にアクセスしてください。

---

## ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

**個人使用:** これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

**商業的使用:** これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。



IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態を提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。





---

## 付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒242-8502  
神奈川県大和市下鶴間1623番14号  
日本アイ・ビー・エム株式会社  
法務・知的財産  
知的財産権ライセンス渉外

**以下の保証は、国または地域の法律に沿わない場合は、適用されません。** IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態を提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited  
U59/3600  
3600 Steeles Avenue East  
Markham, Ontario L3R 9Z7  
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができませんが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者をお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があります、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、

利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは、現存するままの状態を提供されるものであり、いかなる種類の保証も提供されません。IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. \_年を入れる\_. All rights reserved.

## 商標

IBM、IBM ロゴおよび [ibm.com](http://www.ibm.com)<sup>®</sup> は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、<http://www.ibm.com/legal/copytrade.shtml> をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標です。
- UNIX は The Open Group の米国およびその他の国における登録商標です。
- Intel<sup>®</sup>、Intel ロゴ、Intel Inside<sup>®</sup>、Intel Inside ロゴ、Intel<sup>®</sup> Centrino<sup>®</sup>、Intel Centrino ロゴ、Celeron<sup>®</sup>、Intel<sup>®</sup> Xeon<sup>®</sup>、Intel SpeedStep<sup>®</sup>、Itanium<sup>®</sup>、Pentium<sup>®</sup> は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft<sup>®</sup>、Windows、Windows NT<sup>®</sup>、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。



---

## 索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

### [ア行]

アプリケーション設計

Perl でのプロトタイピング 1

エラー

Perl 5

PHP 29, 45

Python 60

### [カ行]

関数

PHP

db2\_autocommit 28

db2\_bind\_param 20, 22, 25

db2\_client\_info 31

db2\_close 22, 26

db2\_columns 31

db2\_column\_privileges 31

db2\_commit 28

db2\_connect 16

db2\_conn\_error 29

db2\_conn\_errormsg 29

db2\_exec 18

db2\_execute 20, 22, 25

db2\_fetch\_array 22, 26

db2\_fetch\_assoc 22, 26

db2\_fetch\_both 22, 26

db2\_fetch\_object 22, 24

db2\_fetch\_row 22, 26

db2\_foreign\_keys 31

db2\_next\_result 26

db2\_pconnect 16

db2\_prepare 20, 22, 25

db2\_primary\_keys 31

db2\_procedures 31

db2\_procedure\_columns 31

db2\_result 22

db2\_rollback 28

db2\_server\_info 31

db2\_special\_columns 31

db2\_statistics 31

db2\_stmt\_error 29

db2\_stmt\_errormsg 29

db2\_table\_privileges 31

関数 (続き)

Python

ibm\_db.autocommit 59

ibm\_db.bind\_param 53, 56

ibm\_db.client\_info 61

ibm\_db.close 54, 58

ibm\_db.columns 61

ibm\_db.column\_privileges 61

ibm\_db.commit 59

ibm\_db.connect 51

ibm\_db.conn\_error 60

ibm\_db.conn\_errormsg 60

ibm\_db.execute 53, 56

ibm\_db.exec\_immediate 52

ibm\_db.fetch\_assoc 54, 58

ibm\_db.fetch\_both 54, 58

ibm\_db.fetch\_row 54, 58

ibm\_db.fetch\_tuple 54, 58

ibm\_db.foreign\_keys 61

ibm\_db.next\_result 58

ibm\_db.pconnect 51

ibm\_db.prepare 53, 56

ibm\_db.primary\_keys 61

ibm\_db.procedures 61

ibm\_db.procedure\_columns 61

ibm\_db.result 54

ibm\_db.rollback 59

ibm\_db.server\_info 61

ibm\_db.special\_columns 61

ibm\_db.statistics 61

ibm\_db.stmt\_error 60

ibm\_db.stmt\_errormsg 60

ibm\_db.table\_privileges 61

行

フェッチ

Perl 3

PHP 22, 39

Python 54

更新

DB2 インフォメーション・センター 82, 83

コミット・モード

PHP アプリケーション 28, 43

Python アプリケーション 59

ご利用条件

資料 86

### [サ行]

サンプル

Perl 8

## 資料

- 印刷 76
- 概要 75
- 使用に関するご利用条件 86
- 注文 79
- PDF ファイル 76

## ストアド・プロシージャ

### PHP

- 結果の取得 26, 43
- 呼び出し 25, 41

### Python

- 結果の取得 58
- 呼び出し 56

## 静的 SQL

- Perl ではサポートされていない 5

## 接続

- Rails アプリケーション 71

## [タ行]

### チュートリアル

- トラブルシューティング 86
- 問題判別 86
- リスト 85
- Visual Explain 85

### 動的 SQL

- Perl サポート 1

### 特記事項 89

### トラステッド・コンテキスト

- IBM\_DB Ruby ドライバー・サポート
- 詳細情報 71

- PHP アプリケーション 17

### トラブルシューティング

- オンライン情報 86
- チュートリアル 86

### トランザクション

- PHP 28, 43
- Python 59

## [ハ行]

### パラメーター・マーカー

- Perl 4

### プロシージャ

- PHP 25, 41
- Python 56

### ヘルプ

- 言語の構成 81
- SQL ステートメント 80

### ホスト変数

- Perl 3

## [マ行]

### メソッド

#### Perl

- 状態 5
- connect 2
- disconnect 2
- err 5
- errstr 5
- execute 3
- fetchrow 3
- prepare 3

#### PHP

- PDOStatement::bindColumn 41
- PDOStatement::bindParam 36, 38, 41
- PDOStatement::execute 36, 38, 41
- PDOStatement::fetch 39, 41, 43
- PDOStatement::fetchAll 39, 43
- PDOStatement::fetchColumn 39
- PDOStatement::nextRowset 43
- PDO::beginTransaction 43
- PDO::commit 43
- PDO::exec 35
- PDO::prepare 36, 38, 41
- PDO::query 35
- PDO::rollback 43

### メタデータ

#### リトリーブ

- PHP 31
- Python 61

### 問題判別

- チュートリアル 86
- 利用できる情報 86

## [ラ行]

### ラージ・オブジェクト (LOB)

#### 挿入

- PHP 22, 38

#### フェッチ

- PHP 24, 41

## A

### ActiveRecord-JDBC アダプター

- IBM\_DB アダプターの比較 73

### autocommit 関数 (ibm\_db) 59

## B

### bind\_param 関数 (ibm\_db)

- 呼び出し 53, 56



## C

CALL ステートメント

PHP 25, 41

Python 56

client\_info 関数 (ibm\_db) 61

close 関数 (ibm\_db)

結果セットからのフェッチ 54

複数の結果セットの取得 58

columns 関数 (ibm\_db) 61

column\_privileges 関数 (ibm\_db) 61

commit 関数 (ibm\_db) 59

connect 関数 (ibm\_db) 51

connect メソッド (Perl DBI) 2

conn\_error 関数 (ibm\_db) 60

conn\_errormsg 関数 (ibm\_db) 60

## D

DB2 インフォメーション・センター

言語 81

更新 82, 83

バージョン 80

DB2 資料の印刷方法 79

DB2::DB2 ドライバー

ダウンロード 1

リソース 1

pureXML のサポート 5

db2\_autocommit 関数 (ibm\_db2) 28

db2\_bind\_param 関数 (ibm\_db2)

ストアド・プロシージャの呼び出し 25

変数入力を含む SQL ステートメントの実行 20

変数入力を含む SQL ステートメントの準備 20

ラージ・オブジェクトの挿入 22

db2\_client\_info 関数 (ibm\_db2) 31

db2\_close 関数 (ibm\_db2) 22

db2\_columns 関数 (ibm\_db2) 31

db2\_column\_privileges 関数 (ibm\_db2) 31

db2\_commit 関数 (ibm\_db2) 28

db2\_connect 関数 (ibm\_db2) 16

db2\_conn\_error 関数 (ibm\_db2) 29

db2\_conn\_errormsg 関数 (ibm\_db2) 29

db2\_exec 関数 (ibm\_db2) 18

db2\_execute 関数 (ibm\_db2)

ストアド・プロシージャの呼び出し 25

ラージ・オブジェクトの挿入 22

SQL ステートメントの実行 20

db2\_fetch\_array 関数 (ibm\_db2)

結果セットからのデータのフェッチ 22

複数の結果セットの取得 26

db2\_fetch\_assoc 関数 (ibm\_db2)

結果セットからのデータのフェッチ 22

複数の結果セットの取得 26

db2\_fetch\_both 関数 (ibm\_db2)

結果セットからのデータのフェッチ 22

複数の結果セットの取得 26

db2\_fetch\_object 関数 (ibm\_db2) 24

結果セットからのデータのフェッチ 22

db2\_fetch\_row 関数 (ibm\_db2)

結果セットからのデータのフェッチ 22

複数の結果セットの取得 26

db2\_foreign\_keys 関数 (ibm\_db2) 31

db2\_next\_result 関数 (ibm\_db2)

複数の結果セットの取得 26

db2\_pconnect 関数 (ibm\_db2) 16

db2\_prepare 関数 (ibm\_db2)

ストアド・プロシージャの呼び出し 25

ラージ・オブジェクトの挿入 22

SQL ステートメントの準備 20

db2\_primary\_keys 関数 (ibm\_db2) 31

db2\_procedures 関数 (ibm\_db2) 31

db2\_procedure\_columns 関数 (ibm\_db2) 31

db2\_result 関数 (ibm\_db2) 22

db2\_rollback 関数 (ibm\_db2) 28

db2\_server\_info 関数 (ibm\_db2) 31

db2\_special\_columns 関数 (ibm\_db2) 31

db2\_statistics 関数 (ibm\_db2) 31

db2\_stmt\_error 関数 (ibm\_d2b) 29

db2\_stmt\_errormsg 関数 (ibm\_db2) 29

db2\_table\_privileges 関数 (ibm\_db2) 31

disconnect メソッド (Perl DBI) 2

Django

IBM データ・サーバー環境のセットアップ 49

## E

err メソッド 5

errstr メソッド 5

execute 関数 (ibm\_db)

ストアド・プロシージャの呼び出し 56

変数入力を含む SQL ステートメントの実行 53

execute メソッド (Perl DBI) 3

exec\_immediate 関数 (ibm\_db) 52

## F

fetchrow メソッド (Perl DBI) 3

fetch\_assoc 関数 (ibm\_db)

行のフェッチ 54

複数の結果セットのフェッチ 58

列のフェッチ 54

fetch\_both 関数 (ibm\_db)

行のフェッチ 54

複数の結果セットのフェッチ 58

列のフェッチ 54

fetch\_row 関数 (ibm\_db)

行のフェッチ 54

複数の結果セットのフェッチ 58

列のフェッチ 54

fetch\_tuple 関数 (ibm\_db)

行のフェッチ 54

fetch\_tuple 関数 (ibm\_db) (続き)  
 複数の結果セットのフェッチ 58  
 列のフェッチ 54  
foreign\_keys 関数 (ibm\_db) 61

## I

ibm\_db API  
 概要 51  
 詳細情報 47  
IBM\_DB Ruby ドライバーおよび Rails アダプター  
 インストール検査  
 DB2 Express-C 68  
 IBM データ・サーバー 69  
 環境のセットアップ 66  
 従属関係 72  
 詳細情報 65  
 統合開発環境のセットアップ 66  
 トラステッド・コンテキスト 71  
 ActiveRecord-JDBC アダプターの比較 73  
 JRuby のサポート 73  
 Ruby gem のインストール 67  
ibm\_db2 API  
 詳細情報 11  
 トラステッド・コンテキスト 17  
 PHP アプリケーション開発 16  
ibm\_db\_dbi API  
 詳細情報 47  
ibm\_db\_sa アダプター  
 詳細情報 47

## J

JRuby  
 IBM\_DB Ruby ドライバーおよび Rails アダプター 73

## N

next\_result 関数 (ibm\_db) 58

## P

pconnect 関数 (ibm\_db) 51  
PDOStatement::bindColumn メソッド (PDO) 41  
PDOStatement::bindParam メソッド (PDO) 36, 38, 41  
PDOStatement::execute メソッド (PDO) 36, 38, 41  
PDOStatement::fetch メソッド (PDO) 39, 41, 43  
PDOStatement::fetchAll メソッド (PDO) 39, 43  
PDOStatement::fetchColumn メソッド (PDO) 39  
PDOStatement::nextRowset メソッド (PDO) 43  
PDO::beginTransaction メソッド (PDO) 43  
PDO::commit メソッド (PDO) 43  
PDO::exec メソッド (PDO) 35  
PDO::prepare メソッド (PDO) 36, 38, 41  
PDO::query メソッド (PDO) 35

PDO::rollback メソッド (PDO) 43  
pdo\_ibm  
 詳細情報 11  
 PHP アプリケーションの開発 34

## Perl

エラー 5  
概要 1  
行のフェッチ 3  
サンプル・プログラム 8  
資料 1  
制限 5  
ダウンロード 1  
データベースへの接続 2  
ドライバ 1  
パラメーター・マーカー 4  
メソッド  
 connect 2  
 disconnect 2  
 err 5  
 errstr 5  
 execute 3  
 fetchrow 3  
 prepare 3  
 state 5  
問題報告 1  
pureXML のサポート 5  
SQLCODE 5  
SQLSTATE 5

## PHP

アプリケーション開発 11, 16  
エラー処理 29, 45  
関数  
 db2\_autocommit 28  
 db2\_bind\_param 25  
 db2\_client\_info 31  
 db2\_close 22, 26  
 db2\_columns 31  
 db2\_column\_privileges 31  
 db2\_commit 28  
 db2\_connect 16  
 db2\_conn\_error 29  
 db2\_conn\_errormsg 29  
 db2\_exec 18  
 db2\_execute 25  
 db2\_fetch\_array 22, 26  
 db2\_fetch\_assoc 22, 26  
 db2\_fetch\_both 22, 26  
 db2\_fetch\_object 22, 24  
 db2\_fetch\_row 22, 26  
 db2\_foreign\_keys 31  
 db2\_next\_result 26  
 db2\_pconnect 16  
 db2\_prepare 25  
 db2\_primary\_keys 31  
 db2\_procedures 31  
 db2\_procedure\_columns 31

- PHP (続き)
  - 関数 (続き)
    - db2\_result 22
    - db2\_rollback 28
    - db2\_server\_info 31
    - db2\_special\_columns 31
    - db2\_statistics 31
    - db2\_stmt\_error 29
    - db2\_stmt\_errormsg 29
    - db2\_table\_privileges 31
  - 行のフェッチ 22, 39
  - 資料 12
  - ストアド・プロシージャー
    - 結果の取得 26, 43
    - 呼び出し 25, 41
  - セットアップ
    - 概要 12
    - Linux 14
    - UNIX 14
  - ダウンロード 12
  - データベースへの接続 16, 34
  - データベース・メタデータの取得 31
  - トラステッド・コンテキスト
    - 概要 17
  - トランザクション 28, 43
  - プロシージャー 25, 41
  - メソッド
    - PDOStatement::bindColumn 41
    - PDOStatement::bindParam 36, 38, 41
    - PDOStatement::execute 36, 38, 41
    - PDOStatement::fetch 39, 41, 43
    - PDOStatement::fetchAll 39, 43
    - PDOStatement::fetchColumn 39
    - PDOStatement::nextRowset 43
    - PDO::beginTransaction 43
    - PDO::commit 43
    - PDO::exec 35
    - PDO::prepare 36, 38, 41
    - PDO::query 35
    - PDO::rollback 43
  - ラージ・オブジェクト 22, 38
  - ラージ・オブジェクトのフェッチ 24, 41
  - IBM データ・サーバー環境のセットアップ (Windows) 13
  - IBM データ・サーバー用の拡張モジュール 11
  - ibm\_db2 API
    - 概要 16
    - データベースへの接続 16
  - PDO を使用したアプリケーションの開発 34
  - PDO\_IBM 拡張モジュール
    - データベースへの接続 34
    - SQL ステートメントの発行 35
  - SQL ステートメント 18, 20, 22, 35, 36, 38, 39, 41
- prepare 関数 (ibm\_db) 53, 56
- prepare メソッド (Perl DBI) 3
- primary\_keys 関数 (ibm\_db) 61
- procedures 関数 (ibm\_db) 61
- procedure\_columns 関数 (ibm\_db) 61
- pureXML
  - DB2::DB2 ドライバー 5
- Python
  - アプリケーション開発 47, 51
  - エラー処理 60
  - 拡張機能のダウンロード 48
  - 関数
    - ibm\_db.autocommit 59
    - ibm\_db.bind\_param 53, 56
    - ibm\_db.client\_info 61
    - ibm\_db.close 54, 58
    - ibm\_db.columns 61
    - ibm\_db.column\_privileges 61
    - ibm\_db.commit 59
    - ibm\_db.connect 51
    - ibm\_db.conn\_error 60
    - ibm\_db.conn\_errormsg 60
    - ibm\_db.execute 53, 56
    - ibm\_db.exec\_immediate 52
    - ibm\_db.fetch\_assoc 54, 58
    - ibm\_db.fetch\_both 54, 58
    - ibm\_db.fetch\_row 54, 58
    - ibm\_db.fetch\_tuple 54, 58
    - ibm\_db.foreign\_keys 61
    - ibm\_db.next\_result 58
    - ibm\_db.pconnect 51
    - ibm\_db.prepare 53, 56
    - ibm\_db.primary\_keys 61
    - ibm\_db.procedures 61
    - ibm\_db.procedure\_columns 61
    - ibm\_db.result 54
    - ibm\_db.rollback 59
    - ibm\_db.server\_info 61
    - ibm\_db.special\_columns 61
    - ibm\_db.statistics 61
    - ibm\_db.stmt\_error 60
    - ibm\_db.stmt\_errormsg 60
    - ibm\_db.table\_privileges 61
  - 行のフェッチ 54
  - ストアド・プロシージャー
    - 結果の取得 58
    - 呼び出し 56
  - データベースへの接続 51
  - データベース・メタデータの取得 61
  - トランザクション 59
  - プロシージャー 56
  - API 資料 48
  - IBM データ・サーバー環境のセットアップ 49
  - IBM データ・サーバー用の拡張モジュール 47
  - ibm\_db 51
  - SQL ステートメント 52, 53

## R

### RadRails

Rails での IBM データ・サーバーのセットアップ 66, 67

### Rails アダプター

インストール検査

DB2 Express-C 68

IBM データ・サーバー 69

従属関係 72

詳細情報 65

統合開発環境のセットアップ 66

入門情報 66

IBM\_DB アダプターおよびドライバーのインストール 67

JRuby のサポート 73

### Rails アプリケーション

接続の構成 71

result 関数 (ibm\_db) 54

rollback 関数 (ibm\_db) 59

### Ruby on Rails

ヒープ・サイズの問題 74

### Ruby ドライバー

インストール検査

DB2 Express-C 68

IBM データ・サーバー 69

詳細情報 65

統合開発環境のセットアップ 66

トラステッド・コンテキスト 71

入門情報 66

IBM\_DB アダプターおよびドライバーのインストール 67

JRuby のサポート 73

## S

server\_info 関数 (ibm\_db) 61

special\_columns 関数 (ibm\_db) 61

### SQL ステートメント

ヘルプ

表示 80

PHP 18, 20, 22, 35, 36, 38, 39, 41

Python 52, 53

### SQLAlchemy

拡張機能のダウンロード 48

IBM データ・サーバー環境のセットアップ 49

IBM データ・サーバー用のアダプター 47

state メソッド 5

statistics 関数 (ibm\_db) 61

stmt\_error 関数 (ibm\_db) 60

stmt\_errormsg 関数 (ibm\_db) 60

## T

table\_privileges 関数 (ibm\_db) 61





Printed in Japan

SC88-5879-01



**日本アイ・ビー・エム株式会社**  
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IBM DB2 9.7 for Linux, UNIX, and Windows バージョン 9 リリース 7 Perl、PHP、Python および Ruby on Rails アプリケーションの開発

