**MySQLインストール**

広告

現在MySQLは MySQL Enterprise Edition と MySQL Community Server の2つのタイプがあります。MySQL Enterpriseは商用版のMySQLとなりオフィシャルサポートの提供や管理ツールが付随してきます。これに対してMySQL Community Serverは従来通りの無償データベースとなります。

ここではMySQL最新版のダウンロード方法及びインストール方法について解説します。

関連カテゴリ:[MySQLの使い方](http://www.dbonline.jp/mysql/)

**MySQLインストール**

2015/12の時点で最新バージョンである MySQL Community Server 5.7.9 のインストールを行います。なおインストールはWindows7(32bit)のPCに行っています。

* **MySQLインストール**
* [MySQL 5.7.9 ダウンロード](http://www.dbonline.jp/mysqlinstall/install/index1.html)
* [MySQL 5.7.9 インストール](http://www.dbonline.jp/mysqlinstall/install/index2.html)
* [Windowsサービスの確認](http://www.dbonline.jp/mysqlinstall/install/index4.html)
* [PATHの確認](http://www.dbonline.jp/mysqlinstall/install/index8.html)
* [専用クライアントとコマンドプロンプトを使ってMySQLへ接続](http://www.dbonline.jp/mysqlinstall/install/index5.html)
* [コマンドプロンプトから接続する場合の文字化け対策(文字コードの変更)](http://www.dbonline.jp/mysqlinstall/install/index7.html)
* [データディレクトリの変更](http://www.dbonline.jp/mysqlinstall/install/index6.html)

**ドキュメント**

MySQLを利用する上で参考になるドキュメントについてご紹介します。

* **ドキュメント**
* [MySQL 日本語リファレンスの参照](http://www.dbonline.jp/mysqlinstall/help/index1.html)

MySQLをインストールするにはXAMPPを利用することもできます。XAMPPにつきましては「[XAMPPの使い方](http://www.adminweb.jp/xampp/)」を参照して下さい。

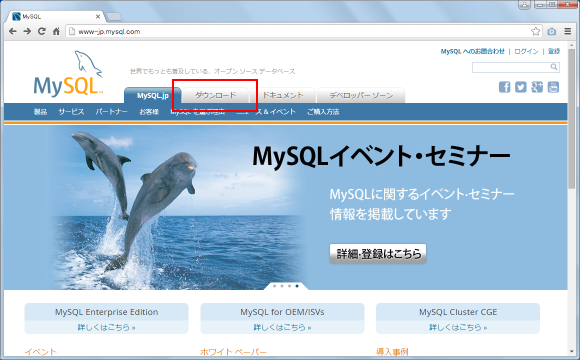
**MySQL 5.7.9 ダウンロード**

それではMySQLのダウンロードから開始します。次のURLから開始します。

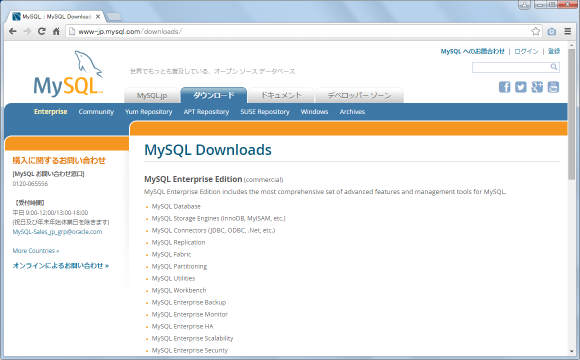
* [**http://www-jp.mysql.com/**](http://www-jp.mysql.com/)



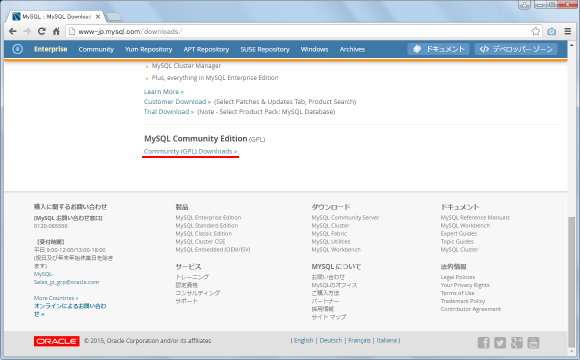
画面上部にある「ダウンロード」と書かれたタブをクリックして下さい。



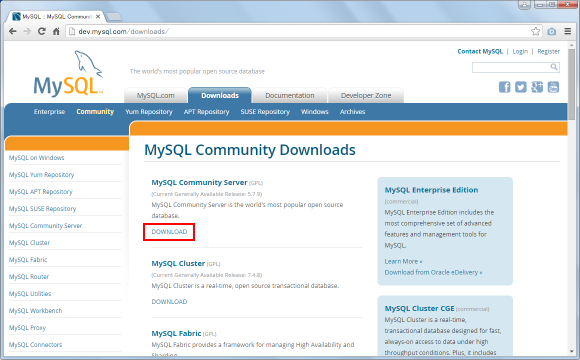
MySQLのダウンロードに関するページが表示されます。



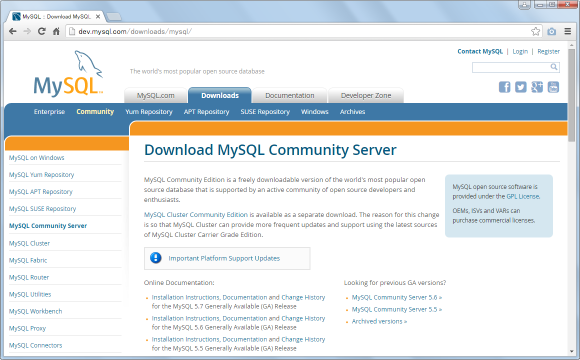
画面の下の方に「MySQL Community Edition (GPL)」と書かれたブロックがあります。そこに表示されている「Community (GPL) Downloads」と書かれたリンクをクリックして下さい。



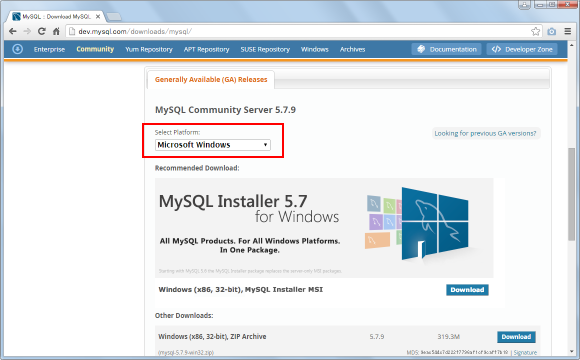
「MySQL Community Server (GPL)」ブロックの中にある「DOWNLOAD」と書かれたリンクをクリックして下さい。



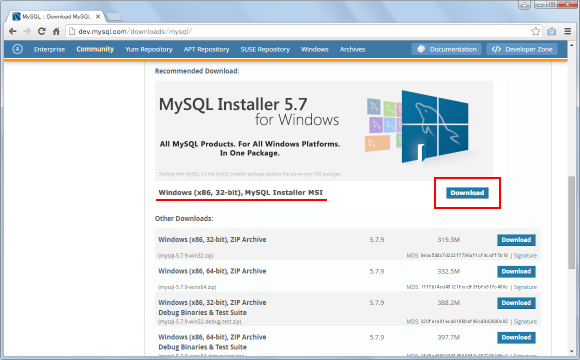
次のような画面が表示されます。



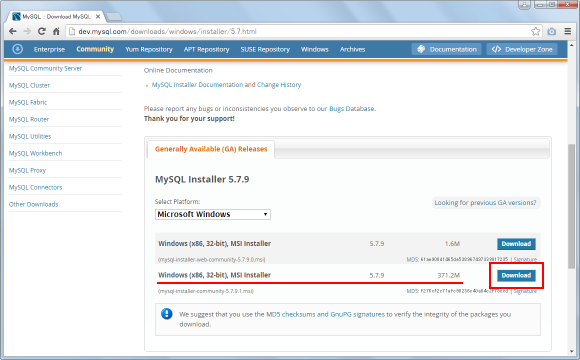
下の方へスクロールして行くと次のような画面が表示されます。MySQLのダウンロードを行うために、利用されるプラットフォームを「Select Platform」に下に表示されているドロップダウンメニューで選択します。今回はWindows7環境にインストールしますので「Microsoft Windows」が選択されていることを確認して下さい。



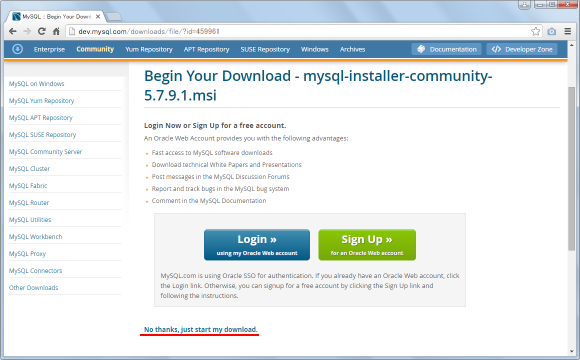
ダウンロードするMySQLの種類がいくつか用意されています。今回は「Windows (x86, 32-bit), MySQL Installer MSI」の右に表示されている「DownLoad」と書かれたリンクをクリックして下さい。



次の画面でも少し下の方へスクロールしてから「Windows (x86, 32-bit), MSI Installer 5.7.9 371.2M」の右に表示されている「DownLoad」と書かれたリンクをクリックして下さい。



MySQL.com アカウントのログイン画面が表示されます。新規登録なども行えるようですが、今回はアカウントの登録は行わずにダウンロードのみ行います。画面下部にある「No thanks, just start my download.」と書かれたリンクをクリックして下さい。



ダウンロードが開始されます。任意の場所に保存しておいて下さい。ダウンロードは以上で終了です。

**MySQL 5.7.9 インストール**

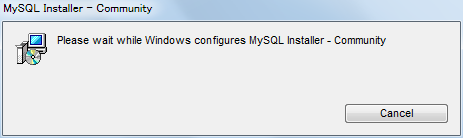
広告

MySQLのインストールを行う方法について解説します。インストール完了後に初期設定も合わせて行います。

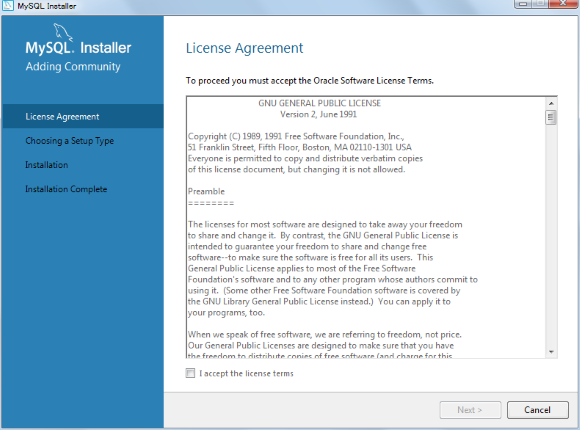
1.[MySQLのインストール](http://www.dbonline.jp/mysqlinstall/install/index2.html#section1)  
2.[MySQLの初期設定](http://www.dbonline.jp/mysqlinstall/install/index2.html#section2)

**MySQLのインストール**

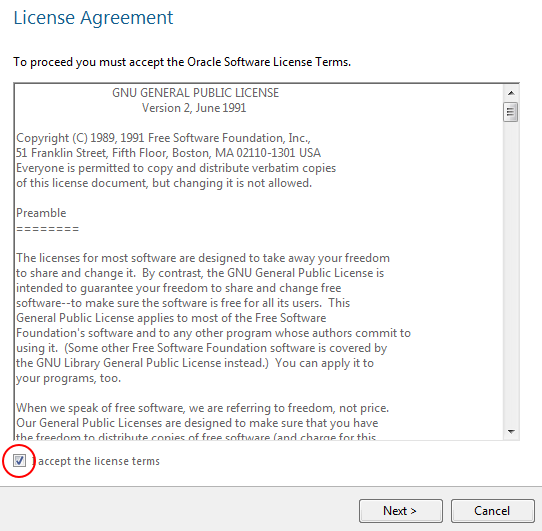
ダウンロードしたファイル名は「mysql-installer-community-5.7.9.1.msi」でインストールパッケージファイルとなっています。ダブルクリックして頂くとインストールが開始されます。



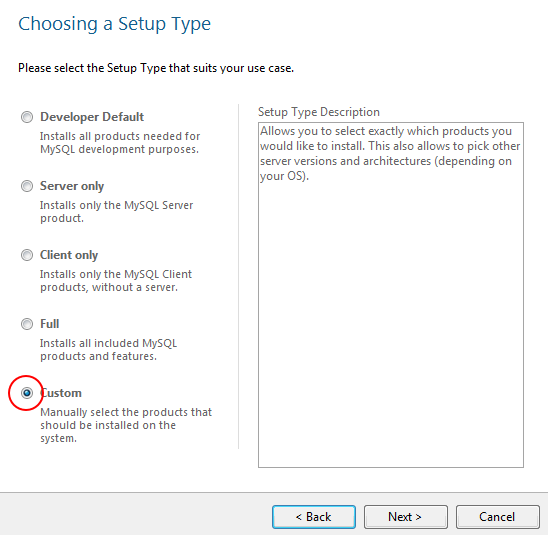
この画面から開始となります。最初はライセンスの確認です。



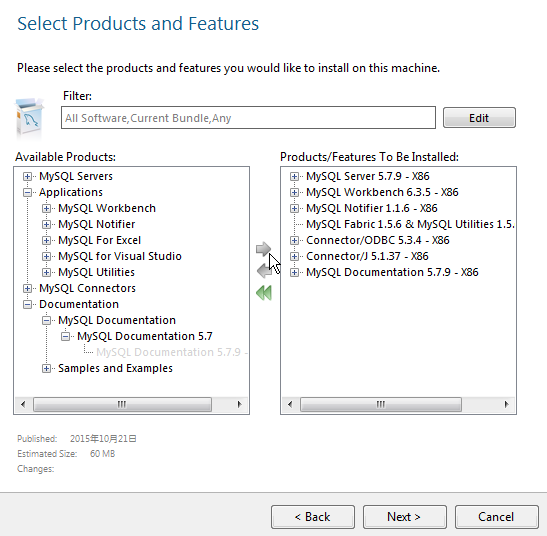
よく読んで頂き、同意できる場合に「I accept the license terms」の左側にあるチェックボックスをチェックして下さい。その後で「Next」をクリックして下さい。



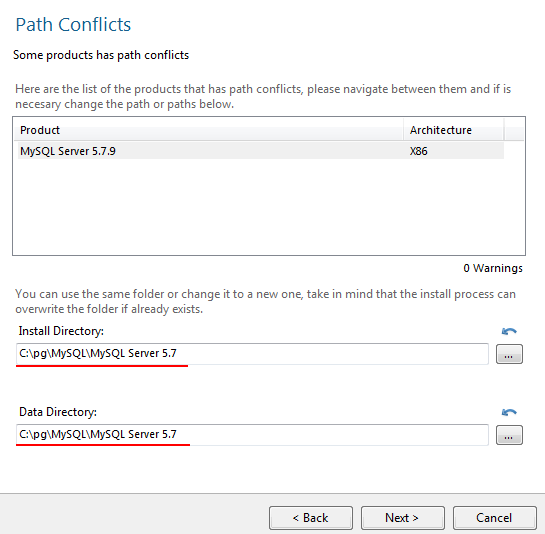
セットアップタイプの選択です。いくつかの種類が用意されていますので利用する目的にあったものを選択して下さい。今回はインストール場所を指定したいため「Custom」を選択しました。選択が終わりましたら「Next」をクリックして下さい。



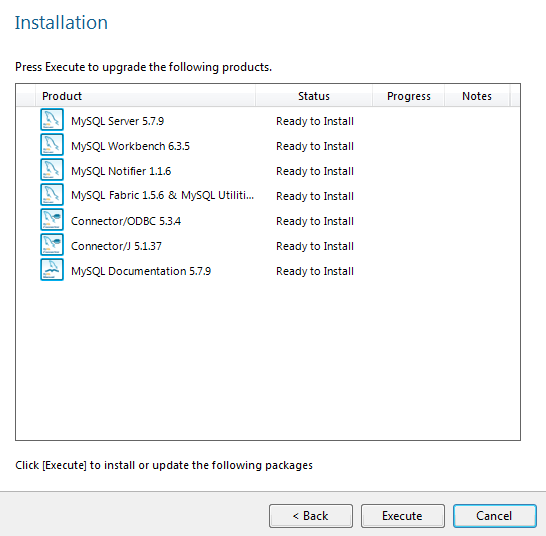
インストールしたい機能を選んで右向きの矢印をクリックして下さい。右側のリストに表示されたものがインストールされます。もしよく分からない場合は「Back」をクリックして一つ前の画面に戻り「Developer Default」や「Full」を選択して下さい。選択が終わりましたら「Next」をクリックして下さい。



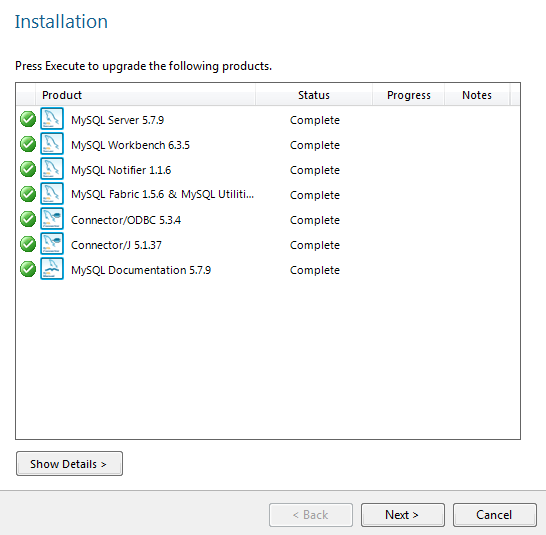
MySQLをインストールするディレクトリとデータを保存するディレクトリを指定して下さい。問題なければデフォルトのままでも結構です(今回は変更してあります)。設定が終わりましたら「Next」をクリックして下さい。



最終確認です。問題が無ければ「Execute」をクリックして下さい。インストールが開始されます。

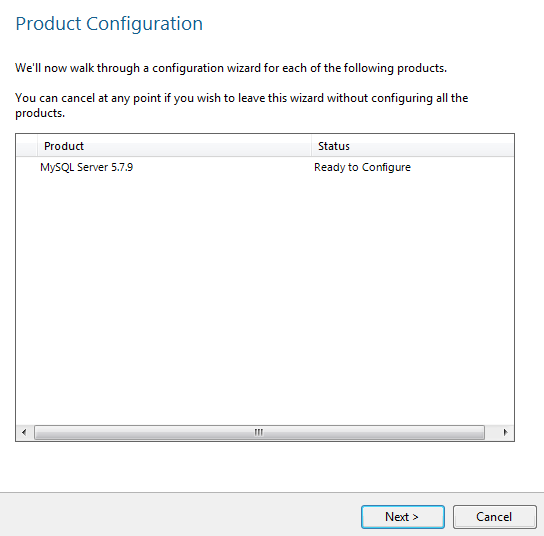


最終的に次のような画面が表示されればインストールは完了です。続いて初期設定を行いますので「Next」をクリックして下さい。

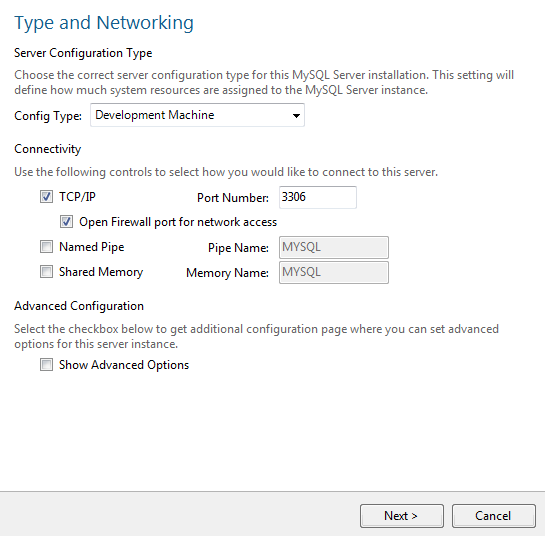


**MySQLの初期設定**

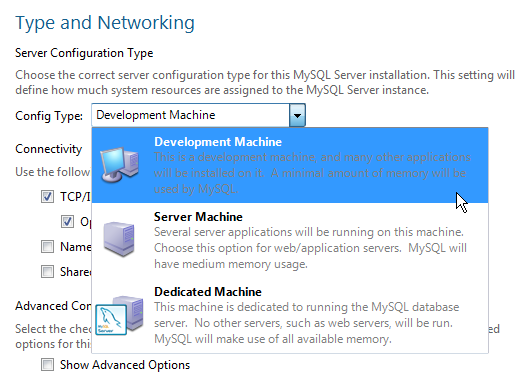
インストールしたMySQLの初期設定を行います。「Next」ボタンをクリックして下さい。



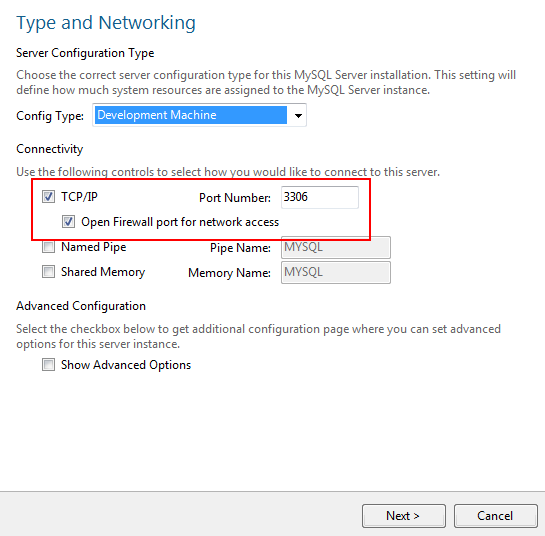
コンフィギュレーションタイプとMySQLとの通信に関する設定画面が表示されます。



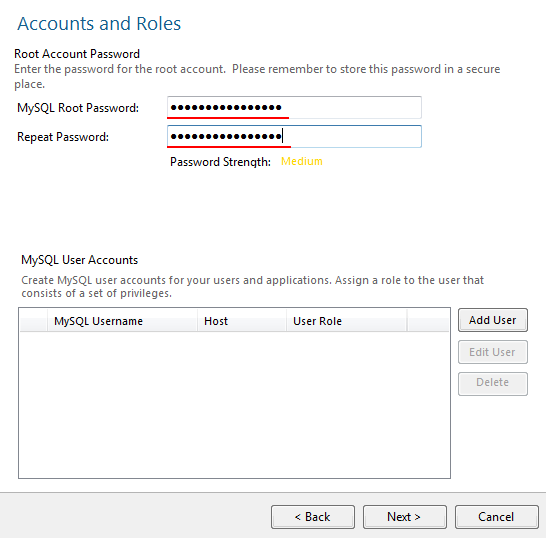
コンフィギュレーションタイプの設定は「Development Machine」「Server Machine」「Dedicated Machine」の3つから選択します。今回は開発用に使用しますので「Development Machine」を選択しました。



MySQLとの通信手段としてTCP/IPでの接続の可否と、可の場合のポート番号を指定します。ポート番号はデフォルトの3306のままで構いませんが、既に他のアプリケーションで使用している場合などは変更して下さい。今回はデフォルトのままとしました。設定が終わりましたら「Next」をクリックして下さい。

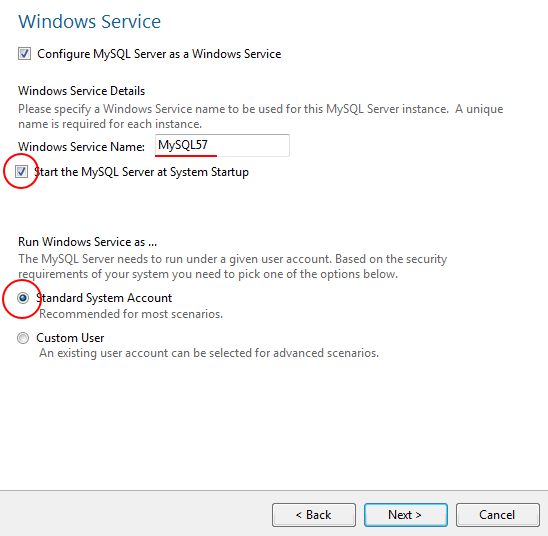


続いてrootアカウントのパスワードの設定とユーザー追加の為の画面が表示されます。管理者アカウントであるrootアカウントのパスワードを設定して下さい。確認のために2か所に同じパスワードを入力して下さい。管理者以外のユーザーアカウントはここでは作成しません(後から追加できます)。設定が終わりましたら「Next」をクリックして下さい。

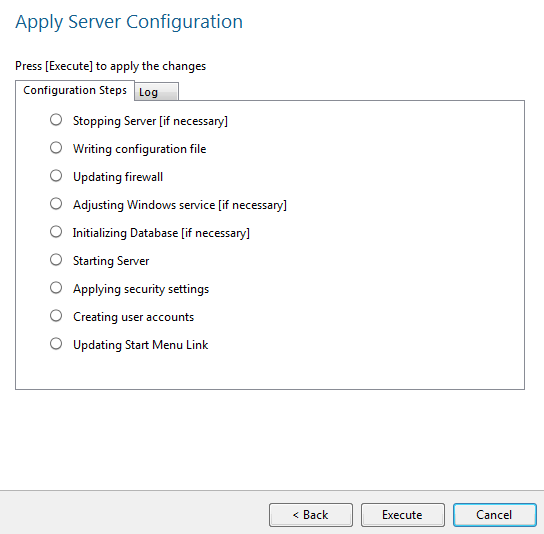


MySQLをWindowsのサービスとして動かすかどうかの設定画面が表示されます。サービスとして動かす場合には「Start the MySQL Server at System Startup」にチェックを行いサービス名を入力して下さい。今回はチェックを行いサービス名はデフォルトで設定されていた「MySQL57」のままとしました。

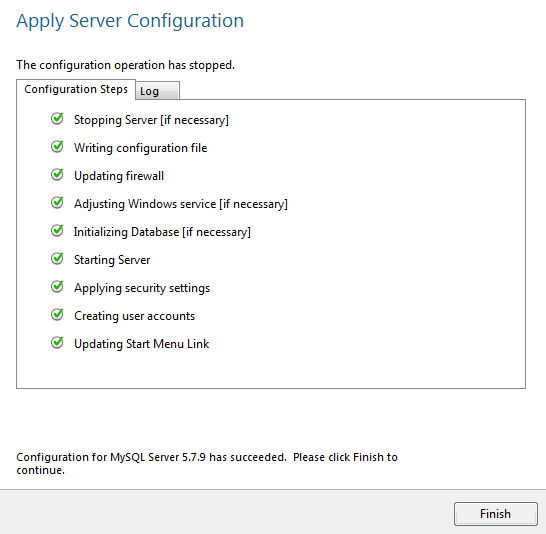
またサービスを動かすWindowsのユーザー名を選択します。特定のユーザーで動かしたい場合には「Custom User」を選択して下さい。今回はデフォルトのままとしました。設定が終わりましたら「Next」をクリックして下さい。



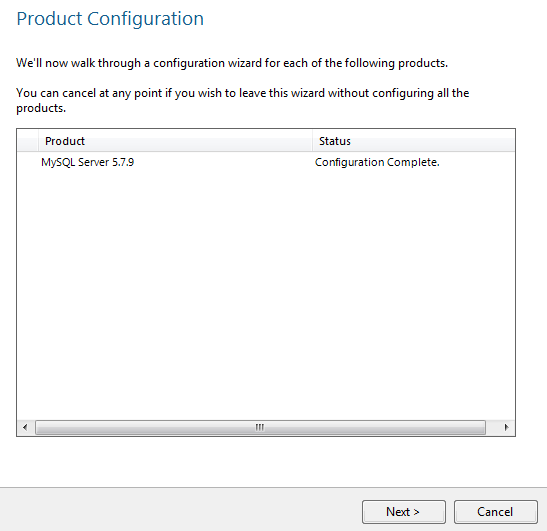
最終確認画面です。よろしければ「Execute」をクリックして下さい。



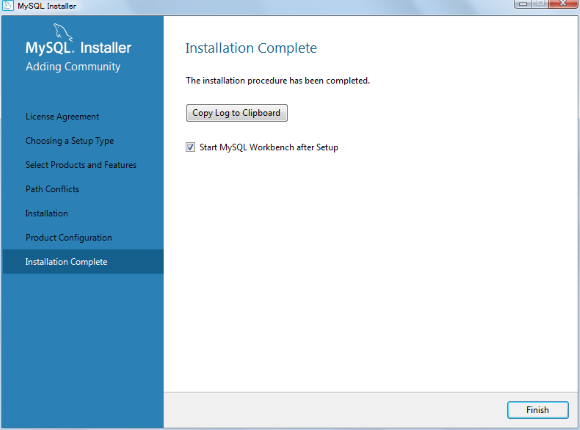
下記の画面が表示されれば初期設定は完了です。「Finish」をクリックして下さい。



次のような画面が表示されます。「Next」をクリックして下さい。



最後に次のような画面が表示されます。「Finish」をクリックして下さい。



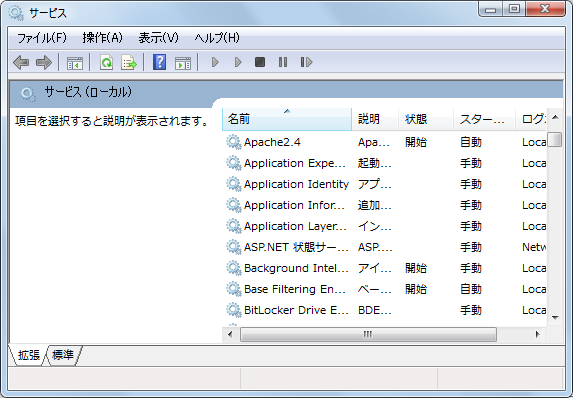
MySQLがサービスとして起動します。以上でMySQLのインストールは完了です。

**Windowsサービスの確認**

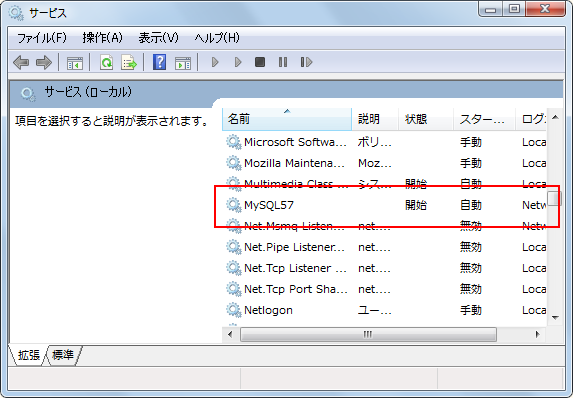
広告

MySQLの初期設定を行った時にWindowsサービスとして動かく設定にしているとMySQLを自動的にWindowsのサービスとして登録してくれます。ここではどのようにサービスとして起動されているかどうかを確認してみます。

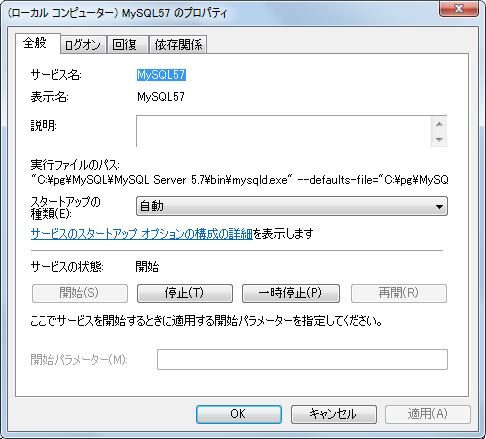
まずサービスの一覧を表示します。Windows7の場合にサービス一覧を表示するには「[サービスの管理](http://www.adminweb.jp/windows7/service/index.html)」を参照して下さい。(Windows XPの場合は「コントロールパネル」>「管理ツール」>「サービス」をクリックして下さい)。



「名前」の欄に「MySQL57」と書かれた行を探します。



その行をダブルクリックして下さい。「MySQL」サービスに関するプロパティが表示されます。



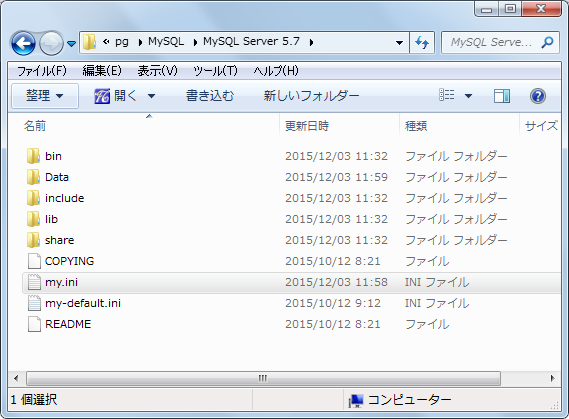
「スタートアップの種類」が「自動」に設定されているのでWindowsが起動する時に自動的にMySQLもサービスとして開始されます。手動でMySQLの起動を行いたい場合は「スタートアップの種類」を「手動」に変更して下さい。

また実行ファイルのパスは次のようになっています(長いので途中で改行しています)。

"C:\pg\MySQL\MySQL Server 5.7\bin\mysqld.exe"

--defaults-file="C:\pg\MySQL\MySQL Server 5.7\my.ini" MySQL57

「--defaults-file="C:\pg\MySQL\MySQL Server 5.7\my.ini"」が指定されていますので、MySQL起動時に指定された位置の「my.ini」ファイルを読み込みます。my.iniファイルは自動的に作成されています。



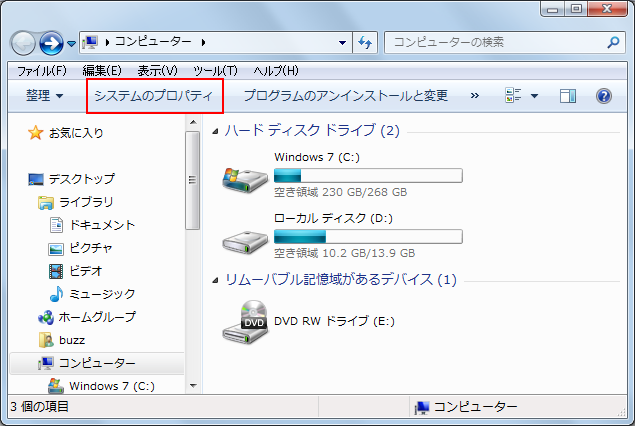
my.iniファイルの中身については別のページで解説します。Windowsサービスの確認は以上です。

**PATHの設定**

広告

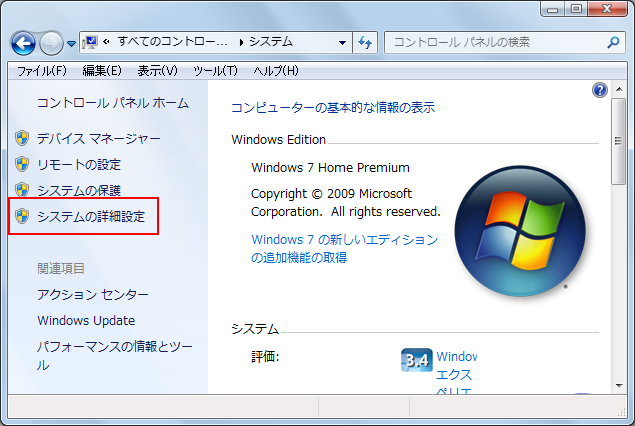
現在のMySQLではMySQL接続用のクライアントが用意されるようになりましたので以前のようにPATHの設定は必要ありませんが、コマンドプロンプトから接続を行いたい場合にはPATHの設定を行っておくと便利です。ここではPATHの設定の方法について解説します。

それではデスクトップ左下にあるWindowsマークをクリックして表示されたメニューの中から「コンピュータ」をクリックして下さい。(Windows 7の場合です)。そして画面上部にある「システムのプロパティ」をクリックして下さい。

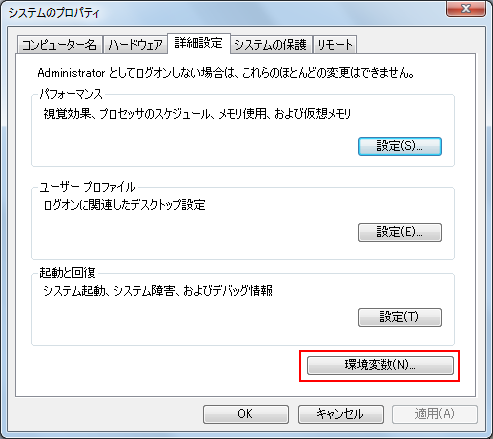


左側メニューの中にある「システムの詳細設定」をクリックして下さい。

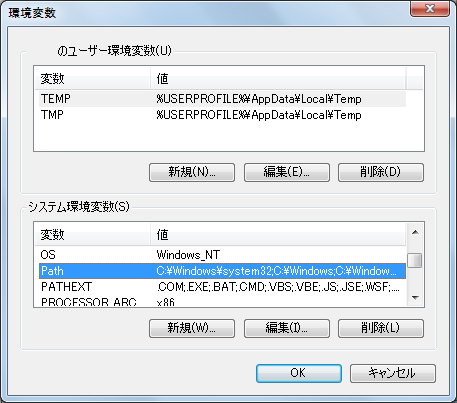
Windows XPの場合には。「コントロールパネル」の中にある「システム」をクリックして開いて下さい。そして「詳細設定」タブをクリックして下さい。



画面下にある「環境変数」をクリックして下さい。

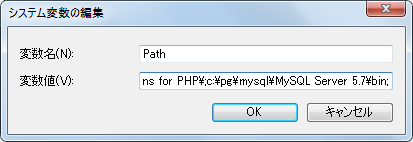


「環境変数」ダイアログが表示されます。



それでは「システム環境変数」の「変数」が「Path」となっている行を選択し、「編集」ボタンを押して下さい。そして「変数値」の最後(またはいずれか)に次の値を追加して下さい。

c:\pg\mysql\MySQL Server 5.7\bin;



この「(MySQLをインストールしたディレクトリ)\bin」ディレクトリにはMySQLで使用される色々なプログラムが格納されています。



「bin」ディレクトリにPATHを設定しておくことでコマンドプロンプトからMySQLへ接続する時に使用する「mysql.exe」がどのディレクトリからでも実行することができます。

**専用クライアントとコマンドプロンプトを使ってMySQLへ接続**

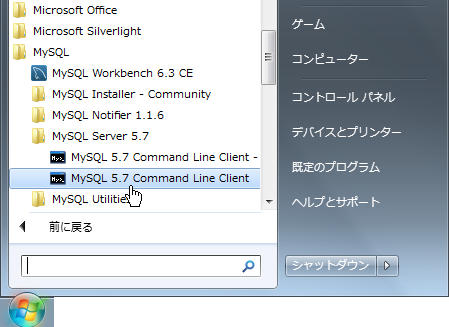
広告

それではMySQLへクライアントから接続を行ってみます。以前はコマンドプロンプトから行っていましたが現在は「MySQL 5.6 Command Line Client」というものが用意されています。ここでは両方の方法で試してみます。

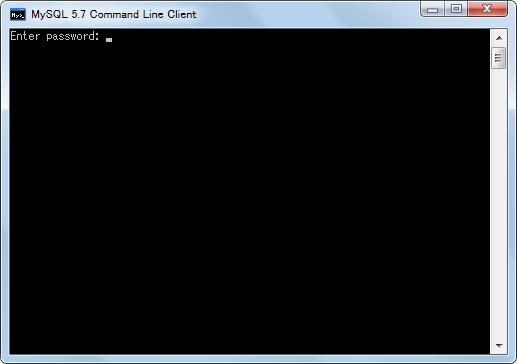
1.[クライアントを使って接続](http://www.dbonline.jp/mysqlinstall/install/index5.html#section1)  
2.[コマンドプロンプトを使って接続](http://www.dbonline.jp/mysqlinstall/install/index5.html#section2)

**クライアントを使って接続**

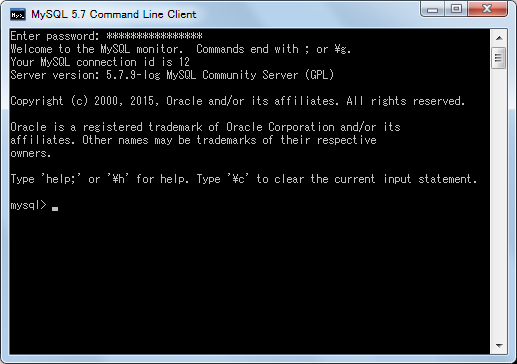
まずは専用クライアントから接続を行います。Windowsのスタートボタンをクリックして「MySQL」>「MySQL Server 5.7」>「MySQL 5.7 Command Line Client」をクリックして下さい。



次のような画面が表示されます。

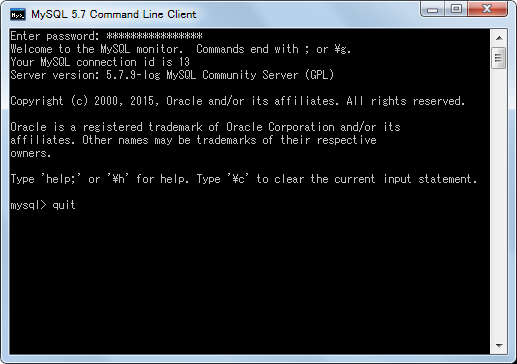


インストール時に指定したrootユーザーのパスワードを入力しリターンキーをクリックして下さい。パスワードを正しく入力できれば下記のように表示されます。この状態がMySQLに接続している状態です。



プログラムなどからではなくクライアントから直接MySQLに対して接続して色々な操作を行いたい場合にはこのようにrootユーザーでログインを行ってから作業を行います。

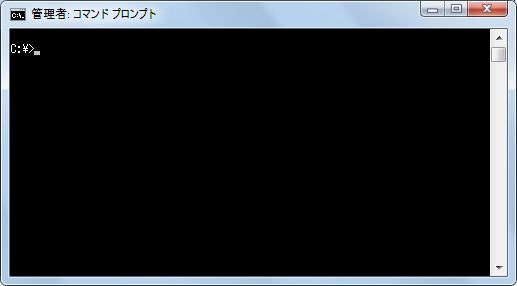
ではMySQLとの接続を終了します。「quit」と入力してEnterキーを押して下さい。



MySQLとの接続が切れます。専用クライアントを使っている場合は接続が切れるとウィンドウも閉じます。

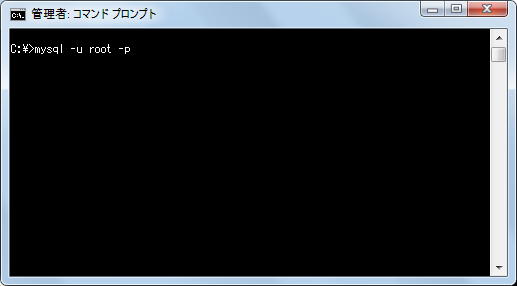
**コマンドプロンプトを使って接続**

続いてコマンドプロンプトから接続を行ってみます。コマンドプロンプトから接続を行う場合はあらかじめパスの設定を行っておいて下さい(設方法については「[PATHの設定](http://www.dbonline.jp/mysqlinstall/install/index8.html)」をご参照下さい)。それではコマンドプロンプトを起動して下さい。

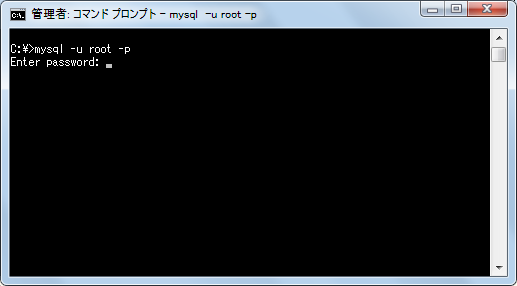


次のように入力して下さい。

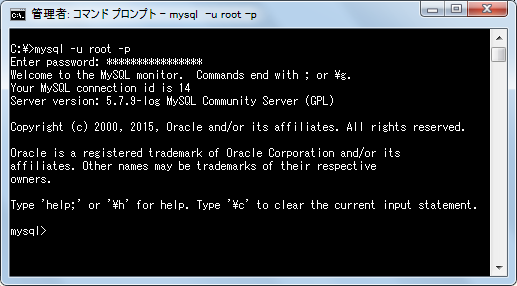
mysql -u root -p



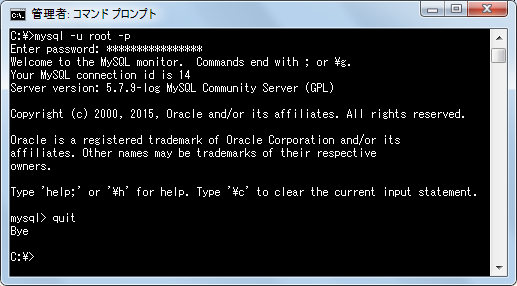
Enterキーをクリックするとrootユーザーのパスワードの入力待ちとなります。rootのパスワードを入力しEnterキーを押して下さい。



パスワードを正しく入力できれば下記のように表示されます。これでコマンドプロンプトからMySQLに接続することができました。



MySQLとの接続を終了するには「quit」と入力してEnterキーを押して下さい。次のように接続が切れます。



専用クライアントから接続した場合とコマンドプロンプトから接続した場合で違いはありません。使いやすい方を利用して下さい。

**コマンドプロンプトから接続する場合の文字化け対策(文字コードの変更)**

広告

コマンドプロンプトからMySQLへ接続する場合、コマンドプロンプトで使用する文字コードをUTF-8にすることができません。(chcp 65001とすることで文字コードだけは変更できるのですが、変更した環境で日本語フォントが選択できないため結局使うことができません)。そこでMySQLコマンドラインツールを使用する場合のデフォルト文字コードとしてcp932を設定する方法について解説します。

※このページに記載している内容が現時点でも正しいかどうか確認が取れていません。あらかじめご了承下さい。

※文字コードの設定について詳しくは「[文字コードの設定](http://www.dbonline.jp/mysql/myini/index3.html)」を参照して下さい。

ではmy.iniファイルをテキストエディタで開いて下さい。55行目付近に次の記述があります。

[client]

no-beep

# pipe

# socket=0.0

port=3306

[mysql]

default-character-set=utf8

[client]の後に書かれているのが全クライアントで共通の設定を記述するところです。また[mysql]の後に書かれているのがコマンドプロンプト経由で接続する場合に使用される設定を記述するところです。

ではこの部分を次のように変更します。

[client]

no-beep

# pipe

# socket=mysql

port=3306

default-character-set=utf8

[mysql]

default-character-set=cp932

[client]の部分に「default-character-set=utf8」を追加し、[mysql]の部分に記述されていた部分を変更しています。

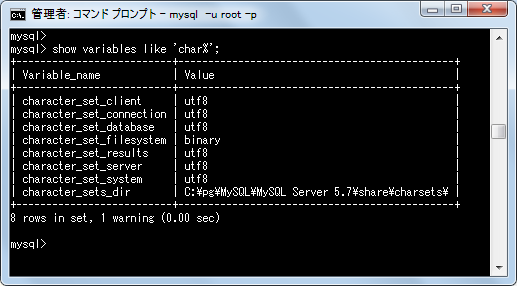
設定変更後にMySQLサーバを再起動して下さい。

では確認を行います。MySQLサーバを起動した状態でコマンドプロンプトからMySQLに接続し、次のように実行して下さい。

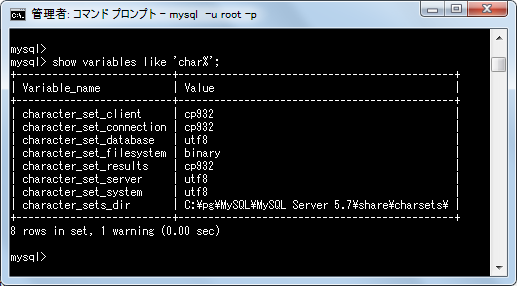
show variables like 'char%';

※MySQLコマンドラインツールを使った接続については「[MySQLに接続するユーザー名とパスワードの指定](http://www.dbonline.jp/mysql/connect/index3.html)」を参照して下さい。

設定変更前に実行した時は次のように表示されていました。



設定変更後にに実行すると次のように表示されます。



上記のように表示されれば設定は完了です。

**データディレクトリの変更**

広告

作成するデータベースのデータが格納されるのがデータディレクトリです。MySQLのインストールの時にデータディレクトリを指定していますが、後からデータディレクトリを変更したい場合などに手動で設定する方法について解説します。

1.[my.iniファイル内の設定箇所](http://www.dbonline.jp/mysqlinstall/install/index6.html#section1)  
2.[MySQLサーバの停止](http://www.dbonline.jp/mysqlinstall/install/index6.html#section2)  
3.[設定ファイルの変更](http://www.dbonline.jp/mysqlinstall/install/index6.html#section3)  
4.[ファイルの移動](http://www.dbonline.jp/mysqlinstall/install/index6.html#section4)  
5.[確認](http://www.dbonline.jp/mysqlinstall/install/index6.html#section5)

**my.iniファイル内の設定箇所**

まず現在の設定を確認します。設定はmy.iniファイルに記載されていますが、このファイルはWindowサービスのページでご説明しましたが、MySQLがサービスとして起動する時に「(現在のデータディレクトリのディレクトリ)\my.ini」を読み込んでいます。

このmy.iniファイルをテキストエディタで開きます。(このディレクトリにはオプションファイルを自分で作成するために使用する同じような名前のテンプレートファイルがたくさん含まれているので注意して下さい)。

95行目付近に次の記述があります。

# Path to the database root

datadir=C:/pg/MySQL/MySQL Server 5.7/Data

「datadir」に設定されている「C:/pg/MySQL/MySQL Server 5.7/Data」ディレクトリが現在設定されているデータディレクトリです。

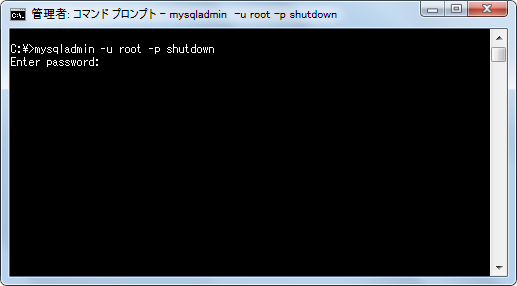
ではデータディレクトリを変更する手順を確認していきます。

**MySQLサーバの停止**

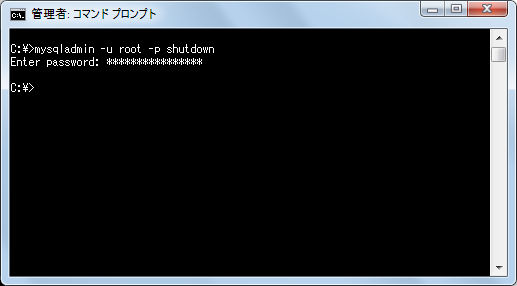
まずMySQLサーバが稼動している場合は停止します。サービスとして動かしている場合はサービス一覧の画面でMySQLを停止させても構いません。今回はコマンドプロンプト上から行います。

コマンドプロンプトを起動し、次のように実行して下さい。

mysqladmin -u root -p shutdown



rootユーザーのパスワードの入力を求められますので入力して下さい。(インストール時に指定したrootアカウントのパスワードです)。MySQLサーバが停止します。



**設定ファイルの変更**

my.iniファイルをテキストエディタで開き、datadirに記載されているディレクトリを新しいディレクトリに変更します。

(旧)

# Path to the database root

datadir=C:/pg/MySQL/MySQL Server 5.7/Data

(新)

# Path to the database root

datadir=D:/mysql/data

※指定するディレクトリは事前に作成しておいて下さい。

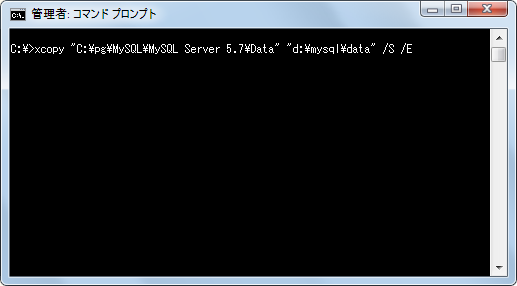
**ファイルの移動**

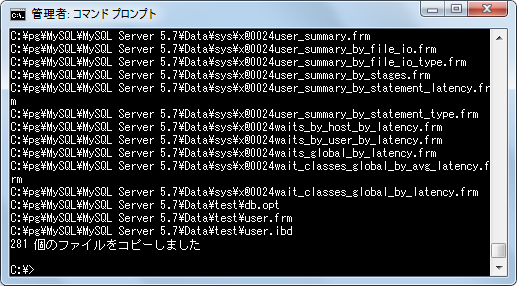
変更前のディレクトリに現在存在している全てのデータを新しいディレクトリにコピーまたは移動します。もしコマンドプロンプトから行うのであれば次のように実行します。

xcopy コピー元ディレクトリ コピー先ディレクトリ /S /E

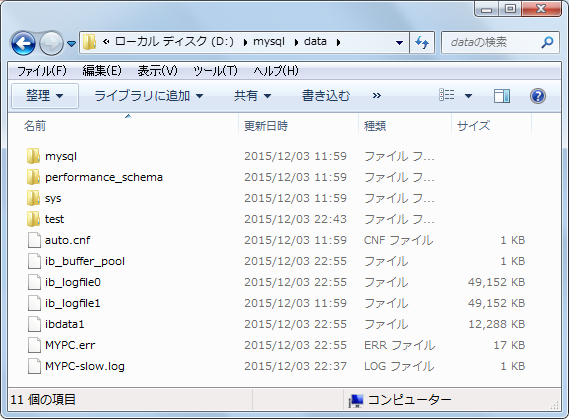
今回次のように実行しました。

xcopy "C:\pg\MySQL\MySQL Server 5.7\Data" "d:\mysql\data" /S /E





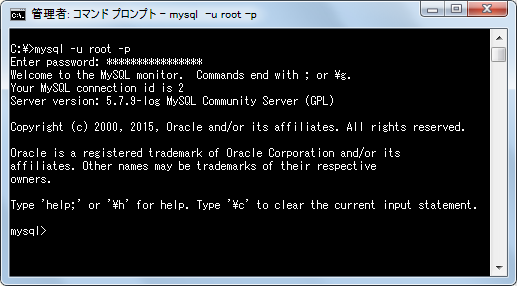
コピーを行った後に新しくデータディレクトリに指定したディレクトリを参照してみます。



ディレクトリやファイルがコピーされていることが確認できます。

**確認**

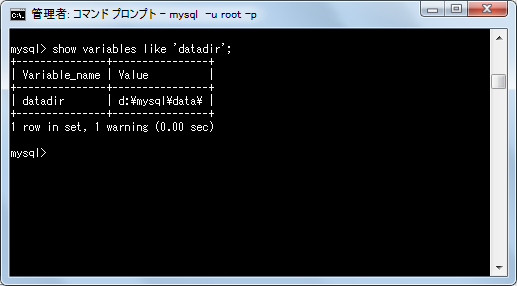
では改めてMySQLを起動し、その後でコマンドプロンプトからMySQLに接続して下さい。



※MySQLコマンドラインツールを使った接続については「[MySQLに接続するユーザー名とパスワードの指定](http://www.dbonline.jp/mysql/connect/index3.html)」を参照して下さい。

次のように実行して下さい。

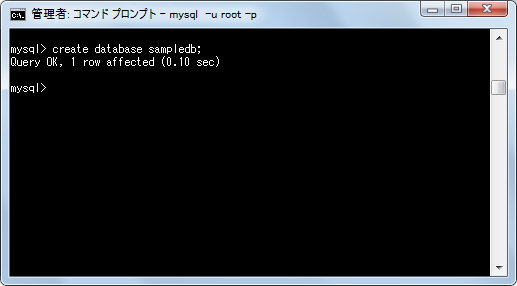
show variables like 'datadir';



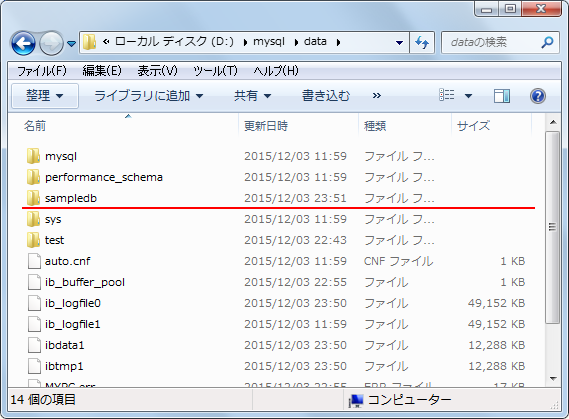
データディレクトリの変更が反映されていることが確認できます。

次にデータベースを1つ作成してみます。

create database sampledb;



「sampledb」データベースが作成されました。では改めてデータディレクトリを参照してみます。



新しく作成したデータベースに関するデータが新しく指定したデータディレクトリに作成されていることが確認できます。

**MySQL 日本語リファレンスの参照**

広告

MySQLを利用する上で参考となる日本語ドキュメントを参照する方法について解説します。

MySQLのドキュメントを参照するには次のURLから開始します。

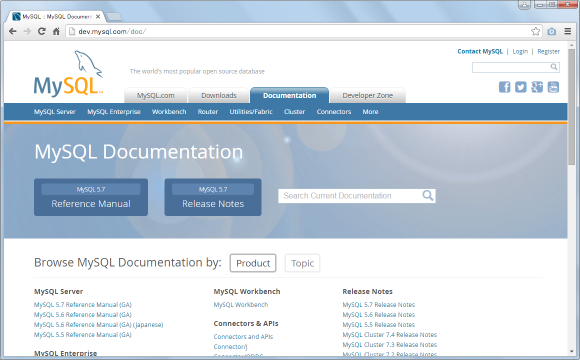
<http://www-jp.mysql.com/>



画面上部にある「ドキュメント」と書かれたタブをクリックして下さい。



ドキュメントの一覧画面が表示されます。



MySQL 5.7に関するドキュメントは残念ながら英語版しか用意されていません。日本語で用意されているのはMySQL 5.6のものになります。1つ前のバージョンのものですが今回は日本語で読めることを優先してMySQL 5.6のドキュメントを参照してみます。



それでは「MySQL 5.6 Reference Manual (GA) (Japanese)」と書かれたリンクをクリックして下さい。



「MySQL 5.6 リファレンスマニュアル」に画面が表示されます。







データベースやテーブルの作成方法などや、用意されている関数の使い方などを参照することができます。

**MySQLの使い方**

広告

MySQLを初めて使われる方を対象としてMySQLの使い方について解説します。データベース、テーブル、ユーザー、トリガなどの作成方法を確認し、実際にデータを格納したり取り出したりする方法について一つ一つ試しながら確認していきます。

MySQLをインストールされていない場合には「[MySQLインストール](http://www.dbonline.jp/mysqlinstall/)」を参照して頂き先にインストールを行っておいて下さい。

**MySQLの基礎**

MySQLをコマンドラインから操作する方法を確認し、その後で基本構文やデータ型などMySQLの基礎的な部分について解説します。

* [**MySQLコマンドラインツールの使い方**](http://www.dbonline.jp/mysql/connect/)
* [MySQLコマンドラインツールの起動](http://www.dbonline.jp/mysql/connect/index1.html)
* [オプションの指定](http://www.dbonline.jp/mysql/connect/index2.html)
* [MySQLに接続するユーザー名とパスワードの指定](http://www.dbonline.jp/mysql/connect/index3.html)
* [(全**5**記事の一覧へ ...)](http://www.dbonline.jp/mysql/connect/)
* [**オプションファイル(my.ini/my.cnf)を使ったオプションの設定**](http://www.dbonline.jp/mysql/myini/)
* [オプションファイルの設置場所](http://www.dbonline.jp/mysql/myini/index1.html)
* [現在の設定内容の確認](http://www.dbonline.jp/mysql/myini/index2.html)
* [文字コードの設定 (default-character-set)](http://www.dbonline.jp/mysql/myini/index3.html)
* [(全**4**記事の一覧へ ...)](http://www.dbonline.jp/mysql/myini/)
* [**MySQLの基本構文**](http://www.dbonline.jp/mysql/ini/)
* [文字列リテラル](http://www.dbonline.jp/mysql/ini/index1.html)
* [数値リテラル](http://www.dbonline.jp/mysql/ini/index2.html)
* [NULL値](http://www.dbonline.jp/mysql/ini/index3.html)
* [(全**10**記事の一覧へ ...)](http://www.dbonline.jp/mysql/ini/)
* [**MySQLのデータ型**](http://www.dbonline.jp/mysql/type/)
* [整数型](http://www.dbonline.jp/mysql/type/index1.html)
* [浮動小数点数型](http://www.dbonline.jp/mysql/type/index2.html)
* [日付型と時刻型](http://www.dbonline.jp/mysql/type/index4.html)
* [(全**8**記事の一覧へ ...)](http://www.dbonline.jp/mysql/type/)
* [**MySQLのファイル構造**](http://www.dbonline.jp/mysql/storage/)
* [データベースのファイル構造](http://www.dbonline.jp/mysql/storage/index1.html)
* [MyISAM型テーブル用のファイル](http://www.dbonline.jp/mysql/storage/index2.html)
* [InnoDB型テーブル用のファイル](http://www.dbonline.jp/mysql/storage/index3.html)
* [(全**3**記事の一覧へ ...)](http://www.dbonline.jp/mysql/storage/)

**MySQLの利用**

データベースやテーブルの作成からデータの操作方法まで、実際にMySQLを利用する方法について解説します。

* [**データベースの作成**](http://www.dbonline.jp/mysql/database/)
* [データベースの作成 (CREATE DATABASE文)](http://www.dbonline.jp/mysql/database/index1.html)
* [データベースに関する情報の取得 (SHOW DATABASES文, SHOW CREATE DATABASE文, INFORMATION\_SCHEMAデータベース)](http://www.dbonline.jp/mysql/database/index2.html)
* [データベースへ接続 (USE文)](http://www.dbonline.jp/mysql/database/index3.html)
* [(全**6**記事の一覧へ ...)](http://www.dbonline.jp/mysql/database/)
* [**ユーザーの作成**](http://www.dbonline.jp/mysql/user/)
* [ユーザーの作成 (CREATE USER文)](http://www.dbonline.jp/mysql/user/index1.html)
* [パスワードの設定 (SET PASSWORD文)](http://www.dbonline.jp/mysql/user/index2.html)
* [ユーザーの変更 (RENAME USER文)](http://www.dbonline.jp/mysql/user/index3.html)
* [(全**8**記事の一覧へ ...)](http://www.dbonline.jp/mysql/user/)
* [**テーブルの作成**](http://www.dbonline.jp/mysql/table/)
* [テーブルの作成 (CREATE TABLE文)](http://www.dbonline.jp/mysql/table/index1.html)
* [テーブルに関する情報の取得 (SHOW TABLES文, SHOW CREATE TABLE, SHOW TABLE STATUS文)](http://www.dbonline.jp/mysql/table/index2.html)
* [カラムに関する情報の取得 (SHOW COLUMNS文)](http://www.dbonline.jp/mysql/table/index3.html)
* [(全**20**記事の一覧へ ...)](http://www.dbonline.jp/mysql/table/)
* [**ビューの作成**](http://www.dbonline.jp/mysql/view/)
* [ビューの作成 (CREATE VIEW文)](http://www.dbonline.jp/mysql/view/index1.html)
* [ビューに関する情報を取得 (SHOW TABLES, SHOW CREATE VIEW文)](http://www.dbonline.jp/mysql/view/index2.html)
* [ビューの削除 (DROP VIEW文)](http://www.dbonline.jp/mysql/view/index3.html)
* [(全**4**記事の一覧へ ...)](http://www.dbonline.jp/mysql/view/)
* [**トリガの作成**](http://www.dbonline.jp/mysql/trigger/)
* [トリガの作成 (CREATE TRIGGER文)](http://www.dbonline.jp/mysql/trigger/index1.html)
* [トリガに関する情報の取得 (SHOW TRIGGERS文)](http://www.dbonline.jp/mysql/trigger/index2.html)
* [古い値と新しい値の参照 (OLD, NEW)](http://www.dbonline.jp/mysql/trigger/index3.html)
* [(全**4**記事の一覧へ ...)](http://www.dbonline.jp/mysql/trigger/)
* [**データの追加と削除**](http://www.dbonline.jp/mysql/insert/)
* [データの追加 (INSERT文)](http://www.dbonline.jp/mysql/insert/index1.html)
* [カラム数が多いテーブルへのデータ追加 (DEFAULT)](http://www.dbonline.jp/mysql/insert/index2.html)
* [複数のデータをまとめて追加](http://www.dbonline.jp/mysql/insert/index3.html)
* [(全**12**記事の一覧へ ...)](http://www.dbonline.jp/mysql/insert/)
* [**データの取得**](http://www.dbonline.jp/mysql/select/)
* [データの取得 (SELECT文)](http://www.dbonline.jp/mysql/select/index1.html)
* [カラムの値に対する演算結果を取得](http://www.dbonline.jp/mysql/select/index2.html)
* [カラムに別名を付ける (AS句)](http://www.dbonline.jp/mysql/select/index3.html)
* [(全**23**記事の一覧へ ...)](http://www.dbonline.jp/mysql/select/)
* [**MySQL関数の使い方**](http://www.dbonline.jp/mysql/function/)
* [DATABASE関数](http://www.dbonline.jp/mysql/function/index1.html)
* [NOW関数 (CURRENT\_TIMESTAMP, LOCALTIME, LOCALTIMESTAMP関数)](http://www.dbonline.jp/mysql/function/index3.html)
* [PASSWORD関数](http://www.dbonline.jp/mysql/function/index2.html)
* [(全**6**記事の一覧へ ...)](http://www.dbonline.jp/mysql/function/)

**MySQLコマンドラインツールの使い方**

広告

MySQLそのものはサーバアプリケーションとして動作しています。そのMySQLに対してPHPやRubyといったプログラムからデータの参照などを行ったりデータの更新を行ったりしますが、対話的にMySQLに対してSQL文を発行するクライアント用ツールとしてMySQLコマンドラインツールが用意されています。

ここではMySQLコマンドラインツールの使い方について確認していきます。

* **MySQLコマンドラインツールの使い方**
* [MySQLコマンドラインツールの起動](http://www.dbonline.jp/mysql/connect/index1.html)
* [オプションの指定](http://www.dbonline.jp/mysql/connect/index2.html)
* [MySQLに接続するユーザー名とパスワードの指定](http://www.dbonline.jp/mysql/connect/index3.html)
* [コマンドの入力と実行](http://www.dbonline.jp/mysql/connect/index4.html)
* [SQL文の実行](http://www.dbonline.jp/mysql/connect/index5.html)

**MySQLコマンドラインツールの起動**

広告

MySQLコマンドラインツールは起動するとMySQLサーバに接続を行います。MySQLコマンドラインツールを起動するにはコマンドプロンプトを立ち上げてから次のように実行します。

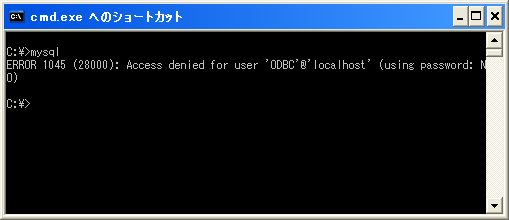
mysql [OPTIONS] [database]

オプションとデータベース名を指定することができるのですが、まずはオプションとデータベース名は省略して単に「mysql」と入力して実行してみて下さい。



次のようなエラーが表示されます。

ERROR 1045 (28000): Access denied for user 'ODBC'@'localhost' (using password: NO)



簡単に説明しますとMySQLコマンドラインツールを起動した時にユーザー名やパスワードを指定しない場合、次のデフォルトの値を使ってMySQLサーバへ接続しようとします。

ホスト名 : localhost

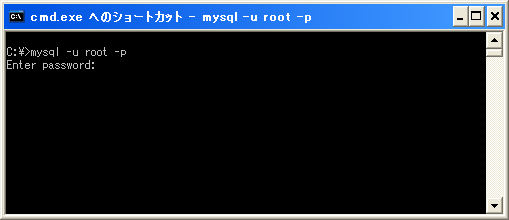
ユーザー名 : ODBC (Windowsの場合、UnixではUnixのログイン名)

MySQLではユーザーはユーザ名とホスト名の組み合わせによって識別されます。今回は何も指定していないため、デフォルトのユーザーである「'ODBC'@'localhost'」でパスワード無し接続しようとしましたが拒否されています。これはMySQLの初期設定の時にAnonymousユーザーを許可しなかったためです。

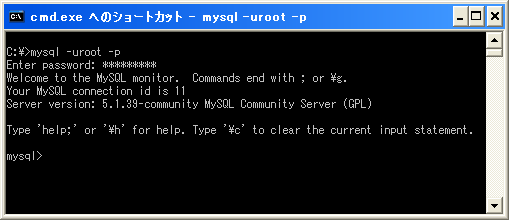
そこで今度は次のように入力して下さい。

mysql -u root -p

これはユーザー名「root」でパスワード付きで接続するという意味になります。



パスワードの入力待ちとなります。インストール時に指定した「root」ユーザーのパスワードを入力して下さい。

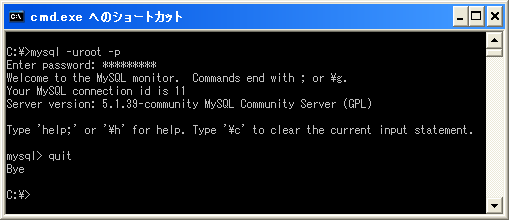


上記のような画面が表示されればMySQLコマンドラインツールが起動し、MySQLサーバへ接続が完了した状態となります。

この画面から様々なコマンドを実行したり、SQL文を送信してMySQLのデータを参照したり更新したりすることができます。

ではいったんMySQLコマンドラインツールを閉じます。次のように入力して下さい。

quit



MySQLコマンドラインツールが終了し、MySQLサーバとの接続も切れます。

**オプションの指定**

広告

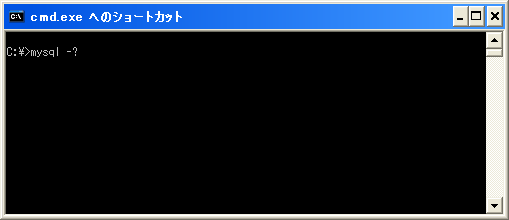
MySQLコマンドラインツールは起動する時の書式をもう一度確認すると次のようにになります。

mysql [OPTIONS] [database]

このページでは指定可能なオプションについて確認していきます。

まずはMySQLコマンドラインツールのヘルプを表示させてみます。コマンドプロンプトを起動し、次のように実行してください。

mysql -?



次のように表示されます。(分かりやすいように出力された内容をテーブルとして表示しています)。

指定可能なオプション：

|  |  |
| --- | --- |
| **OPTIONS** | **意味** |
| -?, --help | Display this help and exit. |
| -I, --help | Synonym for -? |
| --auto-rehash | Enable automatic rehashing. One doesn't need to use 'rehash' to get table and field completion, but startup and reconnecting may take a longer time. Disable with --disable-auto-rehash. |
| -A, --no-auto-rehash | No automatic rehashing. One has to use 'rehash' to get table and field completion. This gives a quicker start of mysql and disables rehashing on reconnect. WARNING: options deprecated; use --disable-auto-rehash instead. |
| -B, --batch | Don't use history file. Disable interactive behavior. (Enables --silent) |
| --character-sets-dir=name | Directory where character sets are. |
| --column-type-info | Display column type information. |
| -c, --comments | Preserve comments. Send comments to the server. The default is --skip-comments (discard comments), enable with --comments |
| -C, --compress | Use compression in server/client protocol. |
| -#, --debug[=#] | This is a non-debug version. Catch this and exit |
| --debug-check | Check memory and open file usage at exit. |
| -T, --debug-info | Print some debug info at exit. |
| -D, --database=name | Database to use. |
| --default-character-set=name | Set the default character set. |
| --delimiter=name | Delimiter to be used. |
| -e, --execute=name | Execute command and quit. (Disables --force and history file) |
| -E, --vertical | Print the output of a query (rows) vertically. |
| -f, --force | Continue even if we get an sql error. |
| -G, --named-commands | Enable named commands. Named commands mean this program's internal commands; see mysql> help . When enabled, the named commands can be used from any line of the query, otherwise only from the first line, before an enter. Disable with --disable-named-commands. This option is disabled by default. |
| -g, --no-named-commands | Named commands are disabled. Use ¥\* form only, or use named commands only in the beginning of a line ending with a semicolon (;) Since version 10.9 the client now starts with this option ENABLED by default! Disable with '-G'. Long format commands still work from the first line. WARNING: option deprecated; use --disable-named-commands instead. |
| -i, --ignore-spaces | Ignore space after function names. |
| --local-infile | Enable/disable LOAD DATA LOCAL INFILE. |
| -b, --no-beep | Turn off beep on error. |
| -h, --host=name | Connect to host. |
| -H, --html | Produce HTML output. |
| -X, --xml | Produce XML output |
| --line-numbers | Write line numbers for errors. |
| -L, --skip-line-numbers | Don't write line number for errors. WARNING: -L is deprecated, use long version of this option instead. |
| -n, --unbuffered | Flush buffer after each query. |
| --column-names | Write column names in results. |
| -N, --skip-column-names | Don't write column names in results. WARNING: -N is deprecated, use long version of this options instead. |
| -O, --set-variable=name | Change the value of a variable. Please note that this option is deprecated; you can set variables directly with --variable-name=value. |
| --sigint-ignore | Ignore SIGINT (CTRL-C) |
| -o, --one-database | Only update the default database. This is useful for skipping updates to other database in the update log. |
| -p, --password[=name] | Password to use when connecting to server. If password is not given it's asked from the tty. |
| -W, --pipe | Use named pipes to connect to server. |
| -P, --port=# | Port number to use for connection or 0 for default to, in order of preference, my.cnf, $MYSQL\_TCP\_PORT, /etc/services, built-in default (3306). |
| --prompt=name | Set the mysql prompt to this value. |
| --protocol=name | The protocol of connection (tcp,socket,pipe,memory). |
| -q, --quick | Don't cache result, print it row by row. This may slow down the server if the output is suspended. Doesn't use history file. |
| -r, --raw | Write fields without conversion. Used with --batch. |
| --reconnect | Reconnect if the connection is lost. Disable with --disable-reconnect. This option is enabled by default. |
| -s, --silent | Be more silent. Print results with a tab as separator, each row on new line. |
| --shared-memory-base-name=name | Base name of shared memory. |
| -S, --socket=name | Socket file to use for connection. |
| --ssl | Enable SSL for connection (automatically enabled with other flags). Disable with --skip-ssl. |
| --ssl-ca=name | CA file in PEM format (check OpenSSL docs, implies --ssl). |
| --ssl-capath=name | CA directory (check OpenSSL docs, implies --ssl). |
| --ssl-cert=name | X509 cert in PEM format (implies --ssl). |
| --ssl-cipher=name | SSL cipher to use (implies --ssl). |
| --ssl-key=name | X509 key in PEM format (implies --ssl). |
| --ssl-verify-server-cert | Verify server's "Common Name" in its cert against hostname used when connecting. This option is disabled by default. |
| -t, --table | Output in table format. |
| --tee=name | Append everything into outfile. See interactive help (¥h) also. Does not work in batch mode. Disable with --disable-tee. This option is disabled by default. |
| --no-tee | Disable outfile. See interactive help (¥h) also. WARNING: option deprecated; use --disable-tee instead |
| -u, --user=name | User for login if not current user. |
| -U, --safe-updates | Only allow UPDATE and DELETE that uses keys. |
| -U, --i-am-a-dummy | Synonym for option --safe-updates, -U. |
| -v, --verbose | Write more. (-v -v -v gives the table output format). |
| -V, --version | Output version information and exit. |
| -w, --wait | Wait and retry if connection is down. |
| --connect\_timeout=# | Number of seconds before connection timeout. |
| --max\_allowed\_packet=# | Max packet length to send to, or receive from server |
| --net\_buffer\_length=# | Buffer for TCP/IP and socket communication |
| --select\_limit=# | Automatic limit for SELECT when using --safe-updates |
| --max\_join\_size=# | Automatic limit for rows in a join when using --safe-updates |
| --secure-auth | Refuse client connecting to server if it uses old (pre-4.1.1) protocol |
| --server-arg=name | Send embedded server this as a parameter. |
| --show-warnings | Show warnings after every statement. |
| aaa | bbb |

現在の設定値：

|  |  |
| --- | --- |
| **Variables** | **Value** |
| auto-rehash | TRUE |
| character-sets-dir | (No default value) |
| column-type-info | FALSE |
| comments | FALSE |
| compress | FALSE |
| debug-check | FALSE |
| debug-info | FALSE |
| database | (No default value) |
| default-character-set | utf8 |
| delimiter | ; |
| vertical | FALSE |
| force | FALSE |
| named-commands | FALSE |
| ignore-spaces | FALSE |
| local-infile | FALSE |
| no-beep | FALSE |
| host | (No default value) |
| html | FALSE |
| xml | FALSE |
| line-numbers | TRUE |
| unbuffered | FALSE |
| column-names | TRUE |
| sigint-ignore | FALSE |
| port | 3306 |
| prompt | mysql> |
| quick | FALSE |
| raw | FALSE |
| reconnect | TRUE |
| shared-memory-base-name | (No default value) |
| socket | (No default value) |
| ssl | FALSE |
| ssl-ca | (No default value) |
| ssl-capath | (No default value) |
| ssl-cert | (No default value) |
| ssl-cipher | (No default value) |
| ssl-key | (No default value) |
| ssl-verify-server-cert | FALSE |
| table | FALSE |
| user | (No default value) |
| safe-updates | FALSE |
| i-am-a-dummy | FALSE |
| connect\_timeout | 0 |
| max\_allowed\_packet | 16777216 |
| net\_buffer\_length | 16384 |
| select\_limit | 1000 |
| max\_join\_size | 1000000 |
| secure-auth | FALSE |
| show-warnings | FALSE |

非常に長いのですが、前半部分が指定可能なオプション、後半部分が現在設定されている値を表示しています。

オプションの指定は二通りの指定方法があるものもあります。例えばバージョンを表示するためのオプションは次のように記載されています。

-V, --version Output version information and exit

このように記載されている場合は次のようにどちらの指定方法を使っても同じです。

mysql -V

mysql --version

全てのオプションを理解しておく必要はありません。必要に応じて設定可能なものが無いかどうかを調べるようにして下さい。

MySQLコマンドラインツールの各値にはデフォルトの値が用意されていますが、一部の値は設定ファイルを読み込んでデフォルトの値として設定しています。設定ファイルは次のファイルを順に読み込んで値を設定していきます。

C:¥WINDOWS¥my.ini

C:¥WINDOWS¥my.cnf

C:¥my.ini

C:¥my.cnf

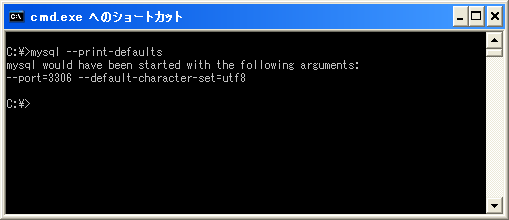
d:¥MySQL¥MySQL Server 5.1¥my.ini

d:¥MySQL¥MySQL Server 5.1¥my.cnf

存在しないファイルもあります。実際私の環境では「d:¥MySQL¥MySQL Server 5.1¥my.ini」しかありませんので、実質的にこのファイルを読み込んでデフォルトの値として設定を行っています。(このファイルはMySQLサーバ側でも読み込んでいるものです)。

ファイルから読み込んで設定している値をは次のようにして確認することができます。

mysql --print-defaults



また次のようにして読み込む設定ファイルを指定することもできます。

mysql --defaults-file="d:¥MySQL¥MySQL Server 5.1¥my.ini"

※パスに空白が含まれる場合は全体をダブルクオーテーションで囲って指定して下さい。

では次のページでMySQLサーバに接続するためのユーザー名とパスワードを指定するためのオプションについて確認します。

**MySQLに接続するユーザー名とパスワードの指定**

広告

MySQLコマンドラインツールを起動すると、自動的にMySQLサーバへ接続も行います。この時ユーザー名やパスワードを指定していしないとデフォルトのユーザー名を使いパスワードは無しで接続しようとするためエラーとなります(anonymousアカウントを許可していない場合)。

そこでここではユーザー名とパスワードの指定方法について確認します。

ユーザー名は次のオプションを使って指定します。

mysql -u ユーザー名

mysql --user=ユーザー名

上記はユーザー名と記載していますが、実際には「ユーザー名@ホスト名」の形式で指定します。

MySQLではユーザーをユーザー名＋接続ホスト名で管理しています。インストール直後とに登録されているのはユーザー名が「root」でホスト名が「localhost」のユーザーのみです。その為、ユーザー名には「root@localhost」と指定する必要があるのですが、ホスト名が「localhost」の場合に限って省略することが可能です。その為、ユーザー名には単に「root」を指定します。

mysql -u root

mysql --user=root

続いてパスワードです。パスワードは次のオプションを使って指定します。

mysql -p[パスワード]

mysql --password[=パスワード]

「-p」オプションを使ってパスワードを指定する場合は、「-p」とパスワードの間に空白をいれずに指定します。例えばパスワードが"pass"だった場合、次のいずれかの方法で指定します。

mysql -u root -ppass

mysql --user=root --password=pass

ただこの方法だとパスワードがそのまま表示されてしまいます。そこで、次のように「-p」または「--password」オプションだけを指定し実際のパスワードは入力を行わずに実行します。

mysql -u root -p

mysql --user=root --password

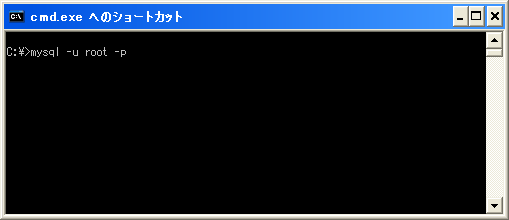
すると「Enter password:」と画面に表示されますのでパスワードの入力を行って下さい。先ほどと違う点は、ここでパスワードを入力しても画面上には伏字で表示されるためパスワードを見られる心配がありません。

正しいパスワードを入力すればMySQLサーバへ指定したユーザー名とパスワードを使って接続を行います。接続に成功すればMySQLコマンドラインツールが起動し、コマンドの入力待ちの状態となります。接続に失敗した場合はコマンドプロンプトへ制御が戻ります。

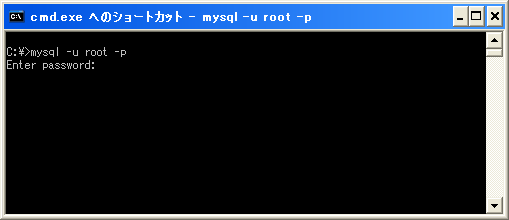
**MySQLコマンドラインツールの起動**

では実際に試してみます。

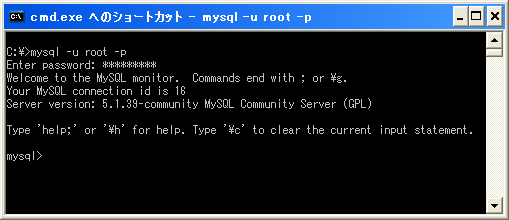
オプションでユーザー名とパスワードを指定してMySQLコマンドラインツールを起動します。



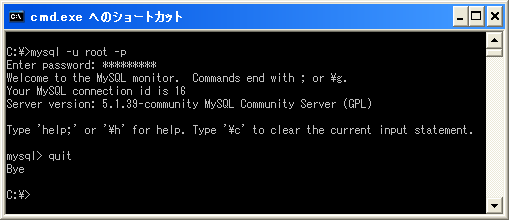
パスワードの入力待ちとなりますので、ユーザー名に対するパスワードを入力します。



指定したユーザー名とパスワードでMySQLサーバへ接続が成功すれば、下記のような画面となりコマンドまたはSQL文の入力待ちとなります。



ではいったんMySQLコマンドラインツールを終了します。「quit」と入力して下さい。



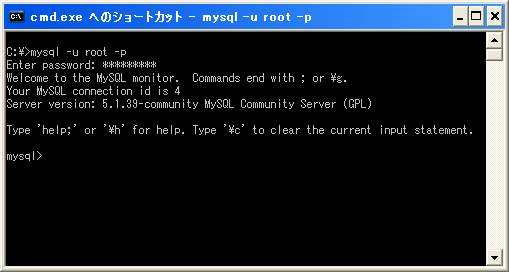
MySQLコマンドラインツールが終了し、MySQLサーバとの接続も切れます。

**コマンドの入力と実行**

広告

MySQLコマンドラインツールを起動しMySQLサーバへ接続すると、任意のSQL文を実行することができます。またSQL文を実行するだけではなく、用意されたコマンドを実行することもできます。ここではコマンドの実行方法について確認します。

まずはMySQLコマンドラインツールを起動して下さい。



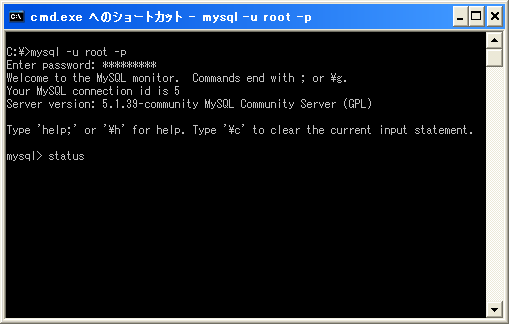
次のようなプロンプトが表示されます。

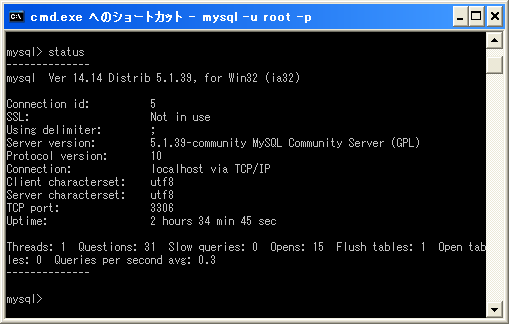
mysql>

この状態からSQL文を入力したりコマンドを入力します。コマンドを入力する時は次のように行います。

mysql> コマンド名

コマンド名を入力した後でEnterキーを押して実行して下さい。では例として「status」コマンドを実行してみます。

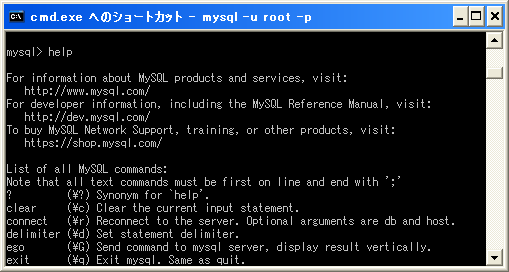




コマンドの種類によりますが、何らかの結果を画面に出力した後で入力待ちの状態となります。

**コマンドの種類**

ではどのようなコマンドが用意されているか確認してみます。「help」コマンドを実行して下さい。



コマンドの一覧と簡単な説明が表示されます。出力された内容をまとめたものを次にまとめました。

|  |  |  |
| --- | --- | --- |
| **コマンド名** | **短縮形** | **説明** |
| ? | ¥? | Synonym for `help'. |
| clear | ¥c | Clear the current input statement. |
| connect | ¥r | Reconnect to the server. Optional arguments are db and host. |
| delimiter | ¥d | Set statement delimiter. |
| ego | ¥G | Send command to mysql server, display result vertically. |
| exit | ¥q | Exit mysql. Same as quit. |
| go | ¥g | Send command to mysql server. |
| help | ¥h | Display this help. |
| notee | ¥t | Don't write into outfile. |
| print | ¥p | Print current command. |
| prompt | ¥R | Change your mysql prompt. |
| quit | ¥q | Quit mysql. |
| rehash | ¥# | Rebuild completion hash. |
| source | ¥. | Execute an SQL script file. Takes a file name as an argument. |
| status | ¥s | Get status information from the server. |
| tee | ¥T | Set outfile [to\_outfile]. Append everything into given outfile. |
| use | ¥u | Use another database. Takes database name as argument. |
| charset | ¥C | Switch to another charset. Might be needed for processing binlog with multi-byte charsets. |
| warnings | ¥W | Show warnings after every statement. |
| nowarning | ¥w | Don't show warnings after every statement. |

各コマンドは通常のものと短縮形がありどちらを使って頂いても同じです。例えばコマンドラインツールを終了するには「quit」と入力して頂いても「¥q」と入力しても同じです。

mysql> quit

mysql> ¥q

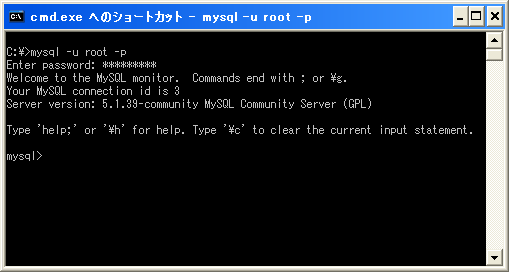
*( Written by Tatsuo Ikura )*

**SQL文の実行**

広告

MySQLコマンドラインツールを使ってSQL文を実行する方法を確認します。

まずはMySQLコマンドラインツールを起動して下さい。



次のようなプロンプトが表示されます。

mysql>

SQL文を入力する時は次のように行います。

mysql> SQL文;

mysql> SQL文¥g

SQL文を入力する時は最後にセミコロン(;)または¥Gを入力した後でEnterキーを押して実行して下さい。SQL文が実行されます。

では次のように入力した後でEnterキーを押して下さい。

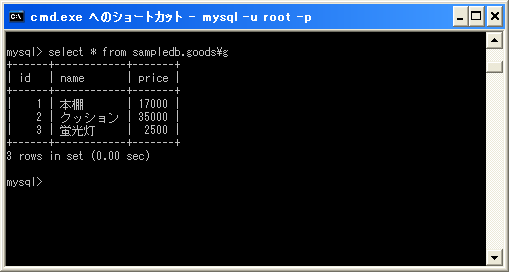
mysql> select \* from sampledb.goods;



SQL文がMySQLサーバへ送信され、結果を取得して表示しています。(なお、上記のように表示されるのは事前にテスト用のデータベースを作成し、テスト用のテーブルを追加してからデータをテーブルに追加してあるからです)。

セミコロンの代わりに¥gを最後につけた場合でも同じ結果となります。

mysql> select \* from sampledb.goods¥g

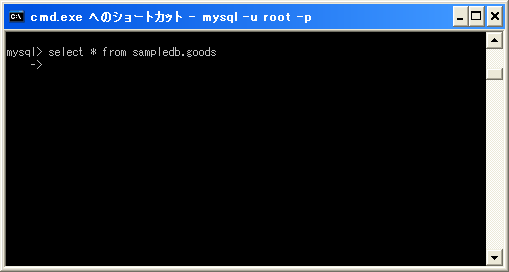


**SQL文の分割入力**

SQL文は最後にセミコロンまたは¥gが入力されるまでが一つの文として処理されます。その為、セミコロンや¥gを記述せずにEnterキーを押した場合、次のように表示され入力待ちとなります。

mysql> select \* from sampledb.goods

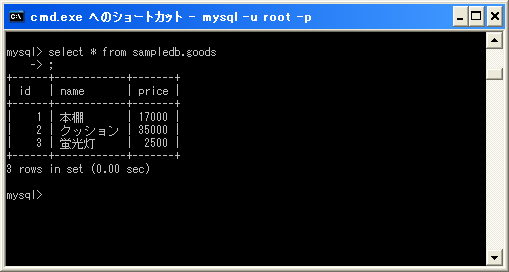
->



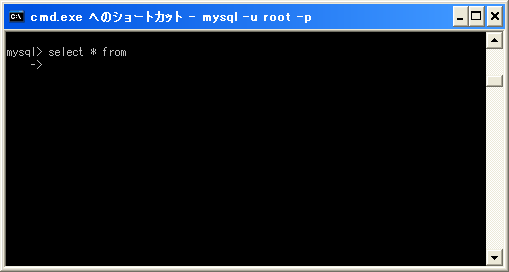
そこで残っているセミコロンなどを入力してEnterキーを押すとSQL文が完結し実行されます。

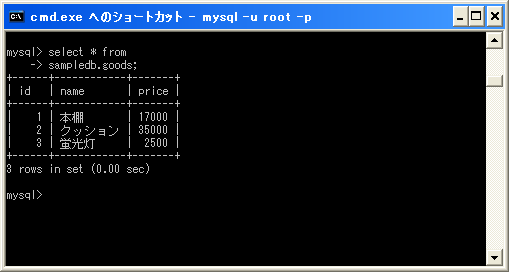
mysql> select \* from sampledb.goods

-> ;



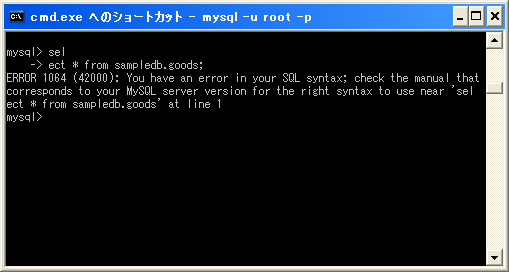
このようにSQL文は分割して入力してもまとめて1つの文として処理してくれます。この機能を利用し、長いSQL文を入力する時には途中でEnterキーを押して分割して入力することができます。





分割すると自動で空白などの区切り文字が入りますので、次の行の行頭で空白を入れる必要はありません。

ただしキーワードや識別子の途中で分割することはできません。



上記のように分割してしまうと"sel ect \* from sampledb.goods;"のように入力したのと同じ扱いになるためエラーとなります。

**オプションファイル(my.ini/my.cnf)を使ったオプションの設定**

広告

MySQLサーバやMySQLコマンドラインツールに対してオプションを設定するために使用されるのがオプションファイルです。オプションファイルに記述された内容はMySQLサーバやコマンドラインツールが起動するときに読み込まれます。

ここではオプションファイルを使ってオプションを設定する方法を確認していきます。

* **オプションの設定**
* [オプションファイルの設置場所](http://www.dbonline.jp/mysql/myini/index1.html)
* [現在の設定内容の確認](http://www.dbonline.jp/mysql/myini/index2.html)
* [文字コードの設定 (default-character-set)](http://www.dbonline.jp/mysql/myini/index3.html)
* [デフォルトストレージエンジンの設定 (default-storage-engine)](http://www.dbonline.jp/mysql/myini/index4.html)

**オプションファイルの設置場所**

広告

MySQLサーバやMySQLコマンドラインツールでは様々なオプションを指定できます。オプションの指定方法はプログラムを実行する時にコマンドの後ろにオプションを指定することもできますが、常に設定されるオプションについてはオプションファイルに記述しておき、起動時に読み込むようにしておくと便利です。

オプションファイルはあらかじめ決められたファイルを読み込みます。Windowsの場合は次のファイルを順番に探して読み込みます。

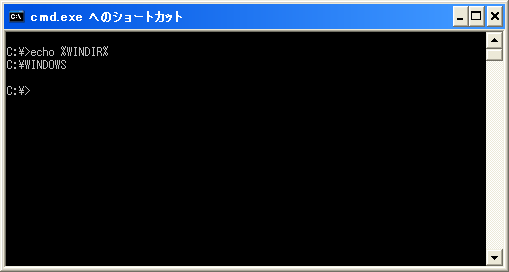
(WINDIR)¥my.ini

C:¥my.cnf

(INSTALLDIR)¥my.ini

※ 上記は何も指定しなくても読み込むファイルですが、上記以外に指定したファイルを読み込ませることも可能です。

(WINDIR)はWindowsディレクトリのことです。環境変数WINDIRを参照することで確認できますがWindowsXPであれば「C:¥WINDOWS」となります。



(INSTALLDIR)はMySQLをインストールしたディレクトリです。私の場合は「D:¥MySQL¥MySQL Server 5.1」となっています。

よって実際には次のオプションファイルを読み込もうとします。

C:¥WINDOWS¥my.ini

C:¥my.cnf

D:¥MySQL¥MySQL Server 5.1¥my.ini

**オプションファイルの設置場所**

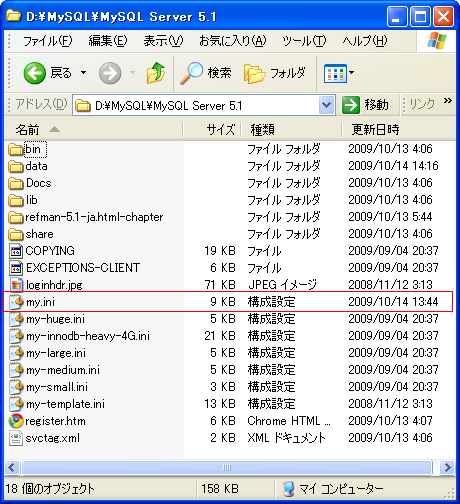
オプションファイルは先ほど記述した順に読み込んでいきます。見つからなければ次のファイルへ進むだけですし、もし見つかった場合でもそこで終わりではなく次のファイルも探して読み込もうとします。複数のファイルで同じオプションに関する記述があった場合には最後に読み込んだ値が有効となります。

MySQLをインストールした時に作成されるオプションファイルは「(INSTALLDIR)¥my.ini」ですので、基本的にはこのファイルを使用し、他のオプションファイルが存在していないことを確認しておくと混乱しないと思います。

**現在の設定内容の確認**

広告

MySQLのインストール時にコンフィギュレーションファイルの作成を選択しているとインストールディレクトリの中に「my.ini」ファイルを作成しオプションの記述が行われます。では「my.ini」ファイルの中身を確認してみます。



※もし存在していなければテキストファイルで作成して下さい。インストールディレクトリの中にいくつかある「my-xxxx.ini」をテンプレートとして、コピーして名前をmy.iniに変更した上で作成することもできます。

オプションファイルは非常に長いのですが、「#」で始まる文はコメントですので一部を除いて削除して表示します。

# CLIENT SECTION

# ----------------------------------------------------------------------

[client]

port=3306

[mysql]

default-character-set=utf8

# SERVER SECTION

# ----------------------------------------------------------------------

[mysqld]

port=3306

basedir="d:/MySQL/MySQL Server 5.1/"

datadir="d:/MySQL/MySQL Server 5.1/data/"

default-character-set=utf8

default-storage-engine=INNODB

sql-mode="STRICT\_TRANS\_TABLES,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION"

max\_connections=100

query\_cache\_size=0

table\_cache=256

tmp\_table\_size=9M

thread\_cache\_size=8

#\*\*\* MyISAM Specific options

myisam\_max\_sort\_file\_size=100G

myisam\_sort\_buffer\_size=17M

key\_buffer\_size=10M

read\_buffer\_size=64K

read\_rnd\_buffer\_size=256K

sort\_buffer\_size=256K

#\*\*\* INNODB Specific options \*\*\*

innodb\_additional\_mem\_pool\_size=2M

innodb\_flush\_log\_at\_trx\_commit=1

innodb\_log\_buffer\_size=1M

innodb\_buffer\_pool\_size=17M

innodb\_log\_file\_size=10M

innodb\_thread\_concurrency=8

大きく分けて「CLIENT SECTION」と「SERVER SECTION」に分かれています。

**CLIENT SECTION**

「CLIENT SECTION」はオプションファイルを参照するクライアントが利用する部分です。例えばMySQLコマンドラインツールはオプションファイルを読み込んで利用します。

# CLIENT SECTION

# ----------------------------------------------------------------------

[client]

port=3306

[mysql]

default-character-set=utf8

CLIENT SECTIONの中でも[client]と書かれた部分は全てのクライアントが利用するオプションです。現在設定されているのは「port=3306」ですのでクライアントからMySQLサーバへの接続に使用するポート番号として3306を使用する設定となっています。

もう一つの[mysql]はMySQLコマンドラインツールである「mysql」だけが参照するオプションです。現在設定されているのは「default-character-set=utf8」ですので使用するデフォルトの文字コートをUTF-8に設定しています。

今後クライアント共通のオプションについては[clien]の後に記述し、MySQLコマンドラインツールだけに関係するオプションについては[mysql]の後に記述して下さい。

**SERVER SECTION**

「SERVER SECTION」はMySQLサーバが利用する部分です。

# SERVER SECTION

# ----------------------------------------------------------------------

[mysqld]

port=3306

basedir="d:/MySQL/MySQL Server 5.1/"

datadir="d:/MySQL/MySQL Server 5.1/data/"

default-character-set=utf8

default-storage-engine=INNODB

#以下略

クライアントと同じく「port=3306」や「default-character-set=utf8」の他に「basedir」はMySQLがインストールされているディレクトリ、「datadir」はデータベースのデータを格納するデータディレクトリ、「default-storage-engine」はデフォルトのストレージエンジンを設定しています。

※データディレクトリはデフォルトの値から変更しています。詳しくは「[データディレクトリの変更](http://www.dbonline.jp/mysqlinstall/install/index6.html)」を参照して下さい。

今後MySQLサーバに関するオプションについては[mysqld]の後に記述して下さい。

**文字コードの設定**

広告

MySQLを利用する上で文字コードの設定はサーバ側の文字コードとクライアント側の文字コードの両方について考慮する必要があります。

サーバ側とクライアント側の文字コードを一致させておけば、クライアントからデータベースに格納するためにデータを送信する場合も、データベースに問い合わせを行って結果をサーバ側からクライアント側へ返す場合も途中で変換などが行わずにサーバとクライアント側ではデータのやり取りが行われます。この場合は文字化けなどを心配する必要はありません。

サーバ側とクライアント側の文字コードが異なる場合、クライアントからサーバへデータを送信する時に、次のような変換が行われます。

クライアント側文字コード --> Unicode(ucs2) --> サーバ側文字コード

またサーバ側からクライアントへデータを返す場合も次のような変換が行われます。

サーバ側文字コード --> Unicode(ucs2) --> クライアント側文字コード

このような二段階の変換が行われるため、設定している文字コードによっては文字化けなどが生じる場合がありますので注意が必要です。

**オプションファイルを使った文字コードの設定**

MySQLサーバ側の設定はオプションファイルなどで「default-character-set」に対して文字コードを設定します。

[mysqld]

default-character-set=文字コード

クライアント側のM設定はオプションファイルなどで「default-character-set」に対して文字コードを設定します。

[client]

default-character-set=文字コード

MySQLコマンドラインツールであれば[mysql]の後に指定しても結構です。

[mysql]

default-character-set=文字コード

**文字コードの確認**

ではオプションファイルに設定した内容がどのように反映されるのかを確認してみます。現在のオプションファイルは次のように記述されています。(※下記はインストール直後の設定です。実際に使っている設定は「[MySQLコマンドラインツールの文字コードの変更](http://www.dbonline.jp/mysqlinstall/install/index7.html)」を参照して下さい)。

# CLIENT SECTION

# ----------------------------------------------------------------------

[mysql]

default-character-set=utf8

# SERVER SECTION

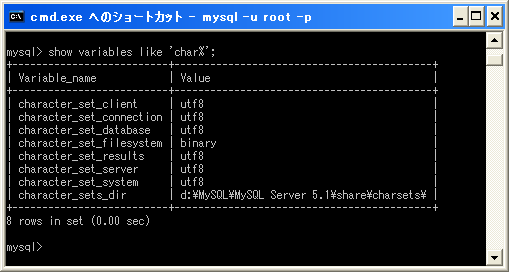
# ----------------------------------------------------------------------

[mysqld]

default-character-set=utf8

MySQLコマンドラインツールでMySQLサーバへ接続し次のように入力して下さい。

show variables like 'char%';



色々な値が表示されますが、特に関係があるのは次の値です。

character\_set\_client utf8 クライアントが送信する文字コード

character\_set\_connection utf8 文字コード情報が無い文字列の文字コード

character\_set\_results utf8 クライアントへ送信する文字コード

character\_set\_server utf8 サーバのデフォルト文字コード

ではサーバ側のdefault-character-setの値をcp932に変更します。

# CLIENT SECTION

# ----------------------------------------------------------------------

[mysql]

default-character-set=utf8

# SERVER SECTION

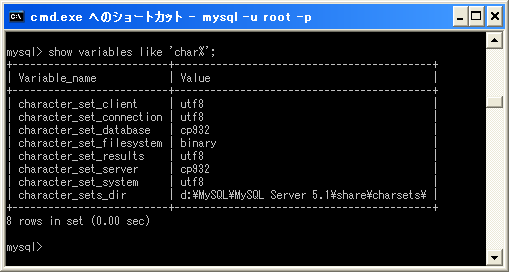
# ----------------------------------------------------------------------

[mysqld]

default-character-set=cp932

MySQLサーバを再起動して先ほどと同じように情報を取得してみます。

この場合、次のように表示されます。



今度はクライアント側のdefault-character-setの値をcp932に変更します。

# CLIENT SECTION

# ----------------------------------------------------------------------

[mysql]

default-character-set=cp932

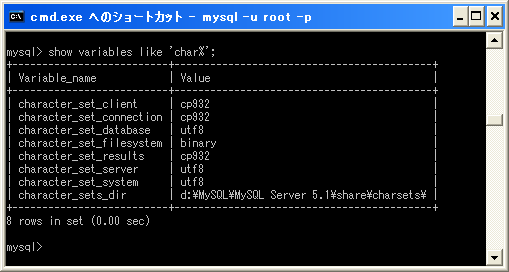
# SERVER SECTION

# ----------------------------------------------------------------------

[mysqld]

default-character-set=utf8

MySQLサーバを再起動して先ほどと同じように情報を取得してみます。



最後にクライアント側のdefault-character-setの設定を消してみます。

# CLIENT SECTION

# ----------------------------------------------------------------------

[mysql]

#default-character-set=cp932

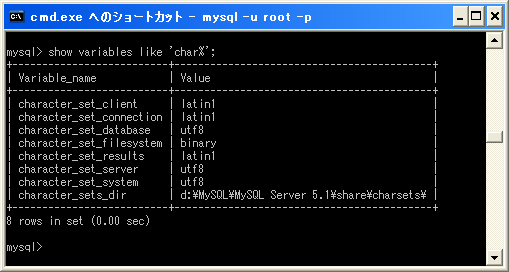
# SERVER SECTION

# ----------------------------------------------------------------------

[mysqld]

default-character-set=utf8

MySQLサーバを再起動して先ほどと同じように情報を取得してみます。



default-character-setの設定を行わない場合、latin1となるようです。

**サポートされている文字コード**

MySQLでサポートされている文字コードは、MySQLコマンドラインツールでMySQLサーバへ接続し次のように入力して下さい。

show character set;



文字コードは「Charset」の列に記載されている値を指定して下さい。

**デフォルトストレージエンジンの設定**

広告

MySQLでは複数のストレージエンジンをサポートしています。ストレージエンジンはテーブル毎に指定することが出来ますが、ストレージエンジンを明示的に指定しなかった場合にどのストレージエンジンを使うかを指定しておくことが出来ます。

デフォルトのストレージエンジンを指定するには[mysqld]の後に「default-storage-engine」に対して設定します。

[mysqld]

default-storage-engine=ストレージエンジン

InnoDBを指定する場合は次のように記述します。

[mysqld]

default-storage-engine=InnoDB

MyISAMを指定する場合は次のように記述します。

[mysqld]

default-storage-engine=MyISAM

ではオプションファイルに設定した内容がどのように反映されるのかを確認してみます。現在のオプションファイルは次のように記述されています。

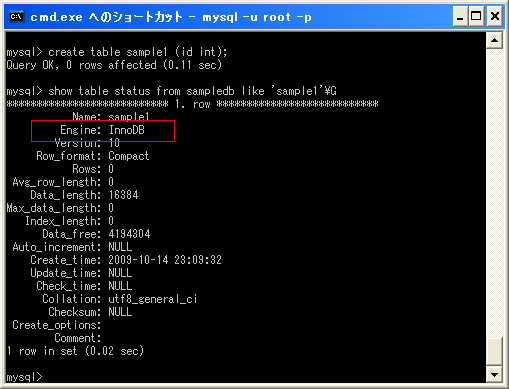
# SERVER SECTION

# ----------------------------------------------------------------------

[mysqld]

default-storage-engine=INNODB

ストレージエンジンを指定せずにテーブルを作成し、テーブルの情報を確認してみます。



作成されたテーブルのストレージエンジンが「InnoDB」となっていることが確認できます。

ではオプションファイルの設定を「default-storage-engine=MyISAM」に変更します。

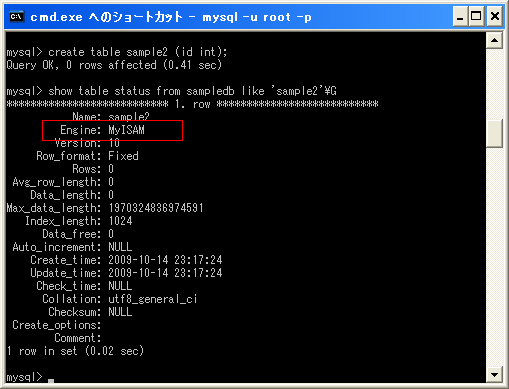
# SERVER SECTION

# ----------------------------------------------------------------------

[mysqld]

default-storage-engine=MyISAM

MySQLサーバを再起動後、再度ストレージエンジンを指定せずにテーブルを作成し、テーブルの情報を確認してみます。



作成されたテーブルのストレージエンジンが「MyISAM」となっていることが確認できます。

**MySQLの基本構文**

広告

ここではMySQLを利用する上で数値や文字列をどのように記述すればいいのかなど基本的な構文に関してと、SELECT文などで演算を行う場合の記述方法について解説します。

* **基本構文**
* [文字列リテラル](http://www.dbonline.jp/mysql/ini/index1.html)
* [数値リテラル](http://www.dbonline.jp/mysql/ini/index2.html)
* [NULL値](http://www.dbonline.jp/mysql/ini/index3.html)
* [識別子と予約語](http://www.dbonline.jp/mysql/ini/index4.html)
* [文字セットと照合順序](http://www.dbonline.jp/mysql/ini/index5.html)
* **演算子**
* [算術演算子](http://www.dbonline.jp/mysql/ini/index6.html)
* [比較演算子](http://www.dbonline.jp/mysql/ini/index7.html)
* [論理演算子](http://www.dbonline.jp/mysql/ini/index8.html)
* [ビット演算子](http://www.dbonline.jp/mysql/ini/index9.html)
* [演算子の優先順位](http://www.dbonline.jp/mysql/ini/index10.html)

**文字列リテラル**

広告

MySQLにおいて文字列リテラルを記述する場合のルールについて確認していきます。

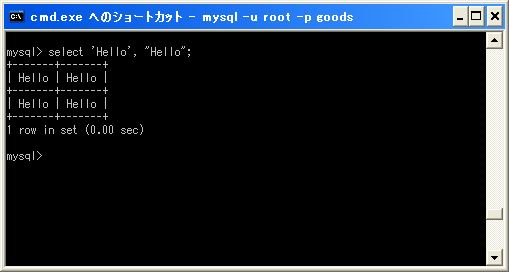
文字列型のカラムに追加する阿多を記述する場合など文字列を記述する場合はシングルクオーテーション(')またはダブルクオーテーショ(")で囲んで記述します。

'文字列'

"文字列"

どちらを使っても構いませんが、比較的シングルクオーテーションを使われているケースが多いように思います。

mysql> select 'Hello', "Hello";



**エスケープ処理**

キーボードから入力できない文字を文字列の中で表したい場合や、特別な意味を持つ文字を入力したい場合にはエスケープ処理が必要です。次のような種類があります。

¥0 ASCII 0(NUL)文字

¥' 単一引用符 (')文字

¥" 二重引用符(")文字

¥b バックスペース文字

¥n 改行文字(LF)

¥r 復帰改行文字

¥t タブ文字

¥Z ASCII 26 (Control-Z)

¥¥ '¥'文字そのもの

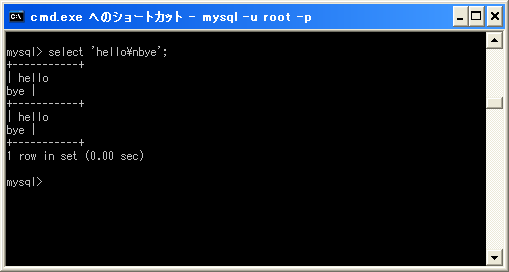
¥% '%' 文字

¥\_ '\_' 文字

※「¥%」と「¥\_」はLIKE式の中でワイルドカードとして使われる文字のため、LIKE式に対するパターン文字列の中でのみエスケープ処理が必要です。

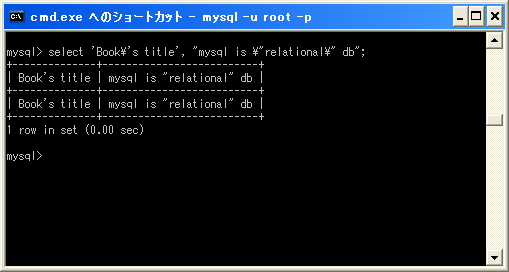
改行文字やタブなどを文字列の中に含める場合には'¥n'や'¥t'を文字列の中に記述して下さい。

mysql> select 'hello¥nbye';



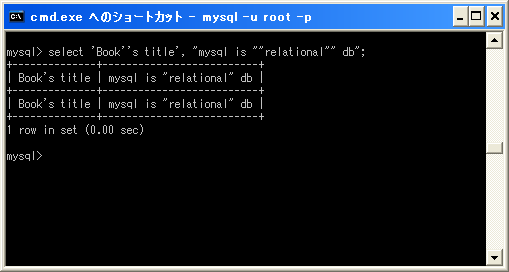
文字列をシングルクオーテーションで囲った場合に、文字列の中でシングルクオーテーションを使用したい場合にエスケープ処理を行って下さい。同じようにダブルクオーテーションで囲った場合に、文字列の中でダブルクオーテーションを使用したい場合もエスケープ処理を行って下さい。

mysql> select 'Book¥'s title', "mysql is ¥"relational¥" db";



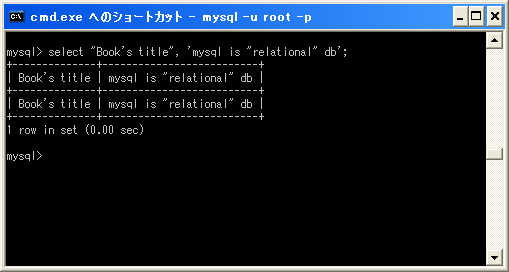
なおこの場合は「¥'」の代わりに「''」、「¥"」の代わりに「""」と使用することもできます。

mysql> select 'Book''s title', "mysql is ""relational"" db";



文字列をシングルクオーテーションで囲った場合に、文字列の中でダブルクオーテーションを使用したい場合やその逆の場合にはエスケープ処理は必要ありません。

mysql> select "Book's title", 'mysql is "relational" db';



**数値リテラル**

広告

MySQLにおいて数値リテラルを記述する場合のルールについて確認していきます。

数値リテラルを記述する場合は、文字列とは異なりそのまま数値を記述して下さい。

100

13.254

また正負を表す'-'や'+'を記述することができます。

-32

-0.332

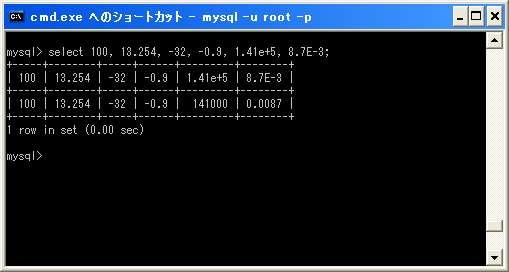
浮動小数点数については指数形式でも記述することができます。

1.414e+5 (=141400)

8.7E-3 (=0.0087)

では試してみます。

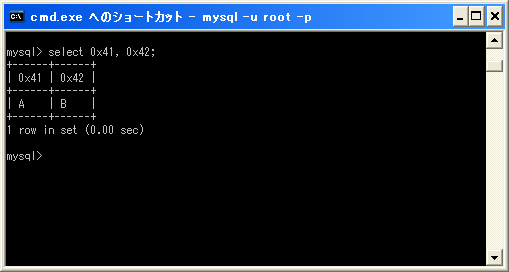
mysql> select 100, 13.254, -32, -0.9, 1.41e+5, 8.7E-3;



**16進数**

16進数をそのまま記述した場合、文字コードとして扱われ文字に変換されます。

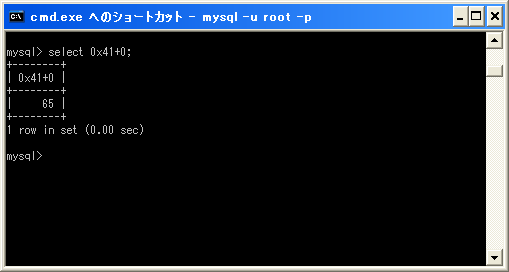
mysql> select 0x41, 0x42;



16進数を数値として扱うには2つ方法があります。1つ目は数値との演算を行う方法です。

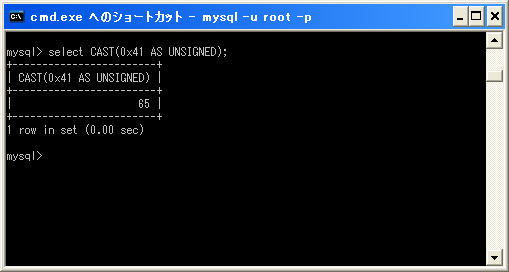
mysql> select 0x41+0;

MySQLでは必要に応じて文字列から数値へ変換したり、数値から文字列へ変換を行います。数値との演算(今回は0を加算)を行うと16進数は数値として処理されます。0を加えても結果は変わりませんので結果的に16進数が表す数値を取得します。



もう一つはキャスト関数を使う方法です。次のように使います。

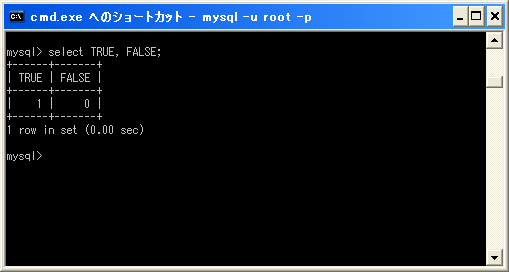
mysql> select CAST(0x41 AS UNSIGNED);



**TRUEとFALSE**

MySQLでは定数としてTRUEとFALSEが定義されており、それぞれ数値の1と0が設定されています。

mysql> select TRUE, FALSE;



なお定数は大文字小文字は区別されませんのでtrueやfalseも同じです。

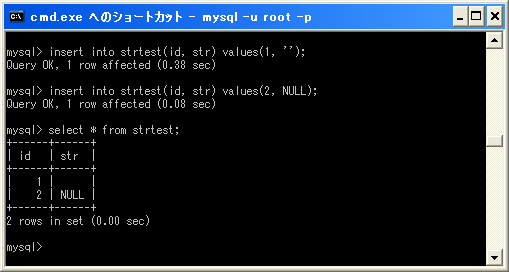
**NULL値**

広告

NULL値はデータが存在しないことを表す値です。大文字と小文字のどちらも使用できます。

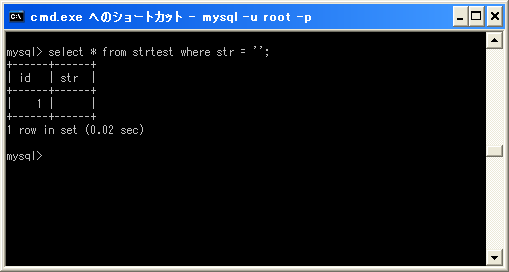
NULLはデータが存在しないということなので、文字列における空文字''や数値における0とは異なります。

例として文字列の値を格納するカラムに対して空文字とNULLを格納してみます。



ではカラムの値が空文字の値を取得してみます。

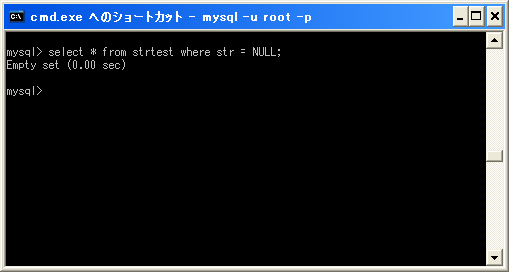
mysql> select \* from strtest where str = '';



空文字とNULLは異なりますので、NULLが格納されているデータは取得されません。

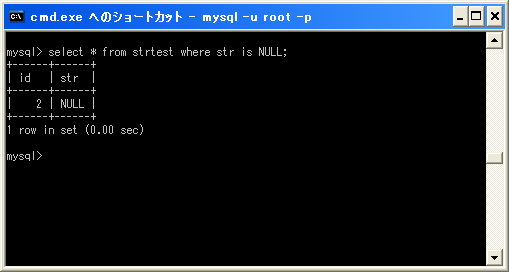
またNULLは他のどの値と比較しても偽となります。例えNULLをNULLと比較しても偽となります。次の例を見てください。

mysql> select \* from strtest where str = NULL;



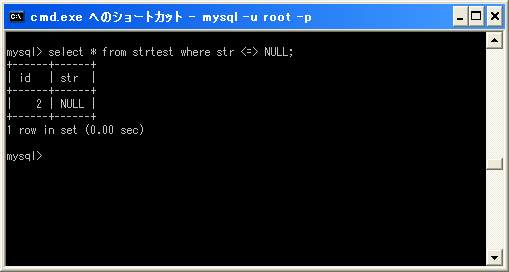
カラムの値がNULLかどうかを調べるには「IS NULL」を使います。次のように入力します。

mysql> select \* from strtest where str is NULL;



また等しいかどうかには演算子として「=」を使いますが、NULLをNULLと比較した場合も真を返す演算子として<=>演算子を使うこともできます。

mysql> select \* from strtest where str <=> NULL;



**識別子と予約語**

広告

データベースやテーブル、カラム、インデックスをなどを作成する時に名前を指定します。この名前のことを識別子と呼びます。

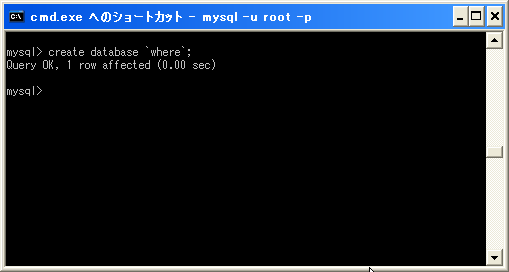
識別子は例外を除いてシングルクオーテーションなどで囲う必要はありません。そのまま記述できます。

CREATE DATABASE personal;

CREATE TABLE addresstbl (id int, address varchar(20));

以前のバージョンであれば使用できない文字もありましたが、MySQL 5.1.6以降であればASCII(0x00)とASCII(0xFF)を除く全ての文字が使用できます。ただし予約語をそのまま使用する場合や特別な文字が含まれる場合は引用符で囲う必要があります。引用符はバッククォート(`)です。

CREATE DATABASE `where`;



ただし予約語や特別な文字を含むような識別子を使うことはできる限り避けるべきだと思います。

**識別子の長さ制限**

データベース名やテーブル名として識別子を付ける場合には長さに制限があります。

データベース名 64byte

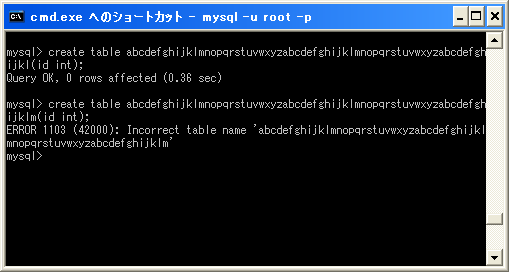
テーブル名 64byte

カラム名 64byte

インデックス名 64byte

エイリアス名 255byte

文字数ではなくバイト数である点に注意して下さい。



上記の例では64バイトのテーブル名であれば作成できますが、65バイトのテーブル名を作成しようとすると「ERROR 1103 (42000): Incorrect table name」というエラーが発生します。

**予約語**

MySQLで予約語として登録されている単語は以下の通りです。

ACCESSIBLE

ADD

ALL

ALTER

ANALYZE

AND

AS

ASC

ASENSITIVE

BEFORE

BETWEEN

BIGINT

BINARY

BLOB

BOTH

BY

CALL

CASCADE

CASE

CHANGE

CHAR

CHARACTER

CHECK

COLLATE

COLUMN

CONDITION

CONSTRAINT

CONTINUE

CONVERT

CREATE

CROSS

CURRENT\_DATE

CURRENT\_TIME

CURRENT\_TIMESTAMP

CURRENT\_USER

CURSOR

DATABASE

DATABASES

DAY\_HOUR

DAY\_MICROSECOND

DAY\_MINUTE

DAY\_SECOND

DEC

DECIMAL

DECLARE

DEFAULT

DELAYED

DELETE

DESC

DESCRIBE

DETERMINISTIC

DISTINCT

DISTINCTROW

DIV

DOUBLE

DROP

DUAL

EACH

ELSE

ELSEIF

ENCLOSED

ESCAPED

EXISTS

EXIT

EXPLAIN

FALSE

FETCH

FLOAT

FLOAT4

FLOAT8

FOR

FORCE

FOREIGN

FROM

FULLTEXT

GRANT

GROUP

HAVING

HIGH\_PRIORITY

HOUR\_MICROSECOND

HOUR\_MINUTE

HOUR\_SECOND

IF

IGNORE

IN

INDEX

INFILE

INNER

INOUT

INSENSITIVE

INSERT

INT

INT1

INT2

INT3

INT4

INT8

INTEGER

INTERVAL

INTO

IS

ITERATE

JOIN

KEY

KEYS

KILL

LEADING

LEAVE

LEFT

LIKE

LIMIT

LINEAR

LINES

LOAD

LOCALTIME

LOCALTIMESTAMP

LOCK

LONG

LONGBLOB

LONGTEXT

LOOP

LOW\_PRIORITY

MASTER\_SSL\_VERIFY\_SERVER\_CERT

MATCH

MEDIUMBLOB

MEDIUMINT

MEDIUMTEXT

MIDDLEINT

MINUTE\_MICROSECOND

MINUTE\_SECOND

MOD

MODIFIES

NATURAL

NOT

NO\_WRITE\_TO\_BINLOG

NULL

NUMERIC

ON

OPTIMIZE

OPTION

OPTIONALLY

OR

ORDER

OUT

OUTER

OUTFILE

PRECISION

PRIMARY

PROCEDURE

PURGE

RANGE

READ

READS

READ\_ONLY

READ\_WRITE

REAL

REFERENCES

REGEXP

RELEASE

RENAME

REPEAT

REPLACE

REQUIRE

RESTRICT

RETURN

REVOKE

RIGHT

RLIKE

SCHEMA

SCHEMAS

SECOND\_MICROSECOND

SELECT

SENSITIVE

SEPARATOR

SET

SHOW

SMALLINT

SPATIAL

SPECIFIC

SQL

SQLEXCEPTION

SQLSTATE

SQLWARNING

SQL\_BIG\_RESULT

SQL\_CALC\_FOUND\_ROWS

SQL\_SMALL\_RESULT

SSL

STARTING

STRAIGHT\_JOIN

TABLE

TERMINATED

THEN

TINYBLOB

TINYINT

TINYTEXT

TO

TRAILING

TRIGGER

TRUE

UNDO

UNION

UNIQUE

UNLOCK

UNSIGNED

UPDATE

USAGE

USE

USING

UTC\_DATE

UTC\_TIME

UTC\_TIMESTAMP

VALUES

VARBINARY

VARCHAR

VARCHARACTER

VARYING

WHEN

WHERE

WHILE

WITH

WRITE

XOR

YEAR\_MONTH

ZEROFILL

以下はMySQL 5.1で登場する新規の予約語です。

ACCESSIBLE

LINEAR

MASTER\_SSL\_VERIFY\_SERVER\_CERT

RANGE

READ\_WRITE

*( Written by Tatsuo Ikura )*

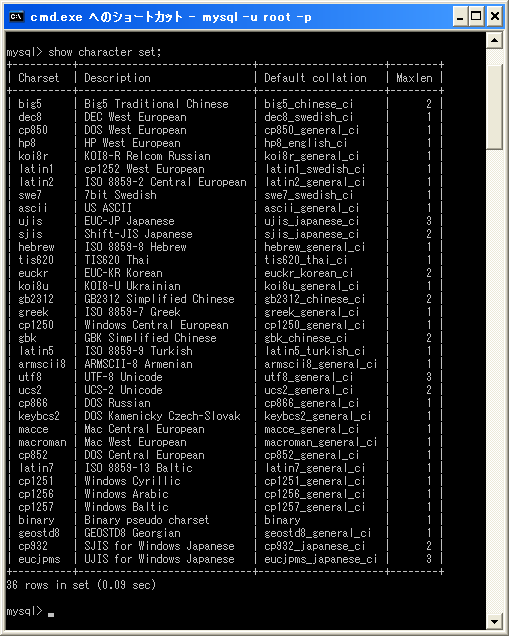
**文字セットと照合順序**

広告

MySQLではサーバ、データベース、テーブル、カラムのそれぞれに対して文字セットと照合順序を設定することができます。ここでは指定可能な値の確認方法について確認します。

まず指定可能な文字セットの一覧は次のように入力することで取得することができます。

SHOW CHARACTER SET;



現在36種類の文字セットが用意されています。比較的使用される文字セットは次の通りです。

+----------+-----------------------------+---------------------+--------+

| Charset | Description | Default collation | Maxlen |

+----------+-----------------------------+---------------------+--------+

| latin1 | cp1252 West European | latin1\_swedish\_ci | 1 |

| ascii | US ASCII | ascii\_general\_ci | 1 |

| ujis | EUC-JP Japanese | ujis\_japanese\_ci | 3 |

| sjis | Shift-JIS Japanese | sjis\_japanese\_ci | 2 |

| utf8 | UTF-8 Unicode | utf8\_general\_ci | 3 |

| ucs2 | UCS-2 Unicode | ucs2\_general\_ci | 2 |

| binary | Binary pseudo charset | binary | 1 |

| cp932 | SJIS for Windows Japanese | cp932\_japanese\_ci | 2 |

| eucjpms | UJIS for Windows Japanese | eucjpms\_japanese\_ci | 3 |

+----------+-----------------------------+---------------------+--------+

文字セットを指定する場合は「Charset」列の値である「utf8」や「cp932」を指定します。なお「Maxlen」列の値は、その文字セットで1つの文字に必要となる最大バイト数です。

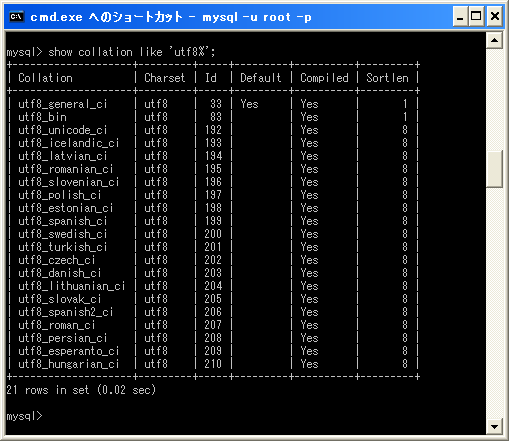
また文字セット毎に照合順序が用意されています。照合順序というのは複数の値を比較する時に、どのように比較するのかを定義しているもので各文字セットにデフォルトで設定されている照合順序は「Default collation」列に記載されています。例えば「utf8」のデフォルト照合順序は「utf8\_general\_ci」です。

では文字セット毎にどのような照合順序が用意されているかを確認してみます。次のように入力することで取得できます。

SHOW COLLATION LIKE 'latin1%';

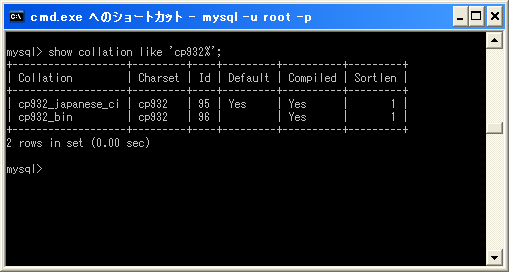
調べたい文字セットをLIKE式の後に'文字コード名%'と指定して下さい。例えばutf8ならば次のように入力します。

mysql> show collation like 'utf8%';



またcp932ならば次のように入力します。

mysql> show collation like 'cp932%';



文字セット毎に用意されている照合順序は異なります。また用意されている数も違います。

文字セットの末尾が「\_ci」となっているものは大文字と小文字を区別しません。末尾が「\_bin」となっているものはバイナリ比較を行います。(バイナリ比較とは格納されているデータをバイナリデータと見なして数値の比較します。結果的に大文字と小文字は区別されます)。また今回は出てきていませんが末尾が「\_cs」となっているものは大文字と小文字を区別します。

**算術演算子**

広告

算術演算子は四則演算を行う為の演算子です。MySQLで利用可能な算術演算子は次の通りです。

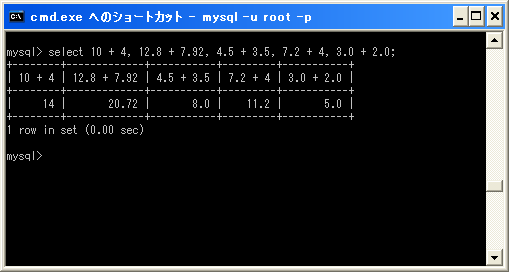
|  |  |  |
| --- | --- | --- |
| **演算子** | **使用例** | **意味** |
| + | a + b | a に b を加える |
| - | a - b | a から b を引く |
| \* | a \* b | a に b をかける |
| / | a / b | a を b で割る |
| DIV | a DIV b | a を b で割る(整数除算) |
| % | a % b | a を b で割った余り |
| MOD | a MOD b | a を b で割った余り |

では各演算子の使い方を確認していきます。

**加算**

加算を表す「+」は左辺の値に右辺の値を加えたものを返します。

mysql> select 10 + 4, 12.8 + 7.92, 4.5 + 3.5, 7.2 + 4, 3.0 + 2.0;

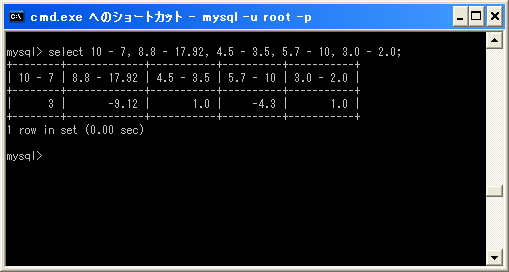


左辺及び右辺が整数の場合は結果も整数ですが、どちらかが浮動小数点数の場合は結果も浮動小数点数となります。

**減算**

減算を表す「-」は左辺の値から右辺の値を引いたものを返します。

mysql> select 10 - 7, 8.8 - 17.92, 4.5 - 3.5, 5.7 - 10, 3.0 - 2.0;

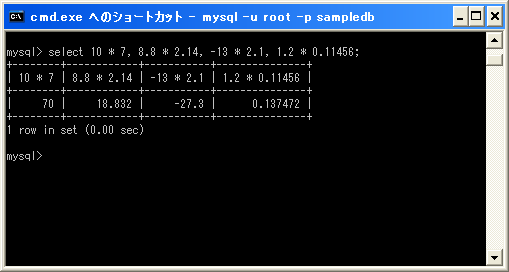


左辺及び右辺が整数の場合は結果も整数ですが、どちらかが浮動小数点数の場合は結果も浮動小数点数となります。

**乗算**

乗算を表す「\*」は左辺の値に右辺の値を掛けたものを返します。

mysql> select 10 \* 7, 8.8 \* 2.14, -13 \* 2.1, 1.2 \* 0.11456;

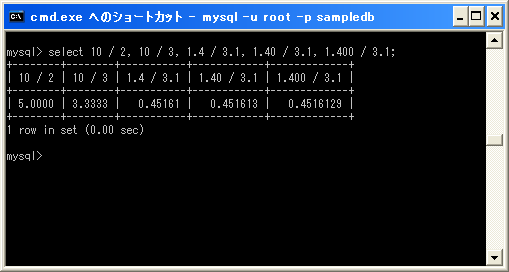


左辺及び右辺が整数の場合は結果も整数ですが、どちらかが浮動小数点数の場合は結果も浮動小数点数となります。

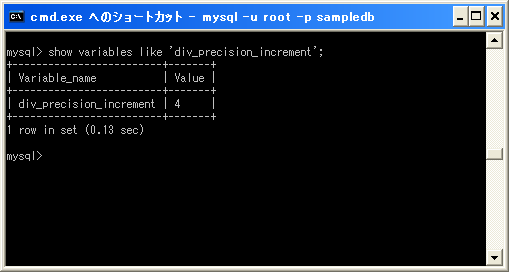
**除算**

除算を表す「/」は左辺の値を右辺の値を割ったものを返します。

mysql> select 10 / 2, 10 / 3, 1.4 / 3.1, 1.40 / 3.1, 1.400 / 3.1;



除算の場合は結果はいずれにしても浮動小数点数となります。また結果の小数点以下の桁数は左辺の数値の小数点以下の桁数にシステム変数「div\_precision\_increment」の値を加えたものとなります。下記の通りこの値のデフォルトの値は4となっています。



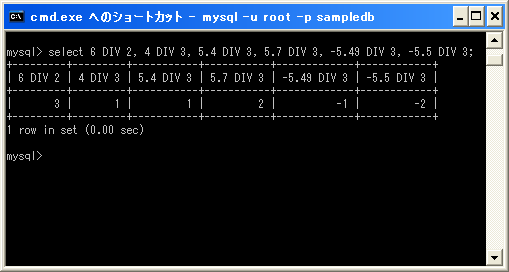
つまり「4 / 3」は左辺の数値の小数点以下の桁数が0なので結果の小数点以下の桁数は0+4=4となり「1.333」となります。「4.1 / 3」なら左辺の小数点以下の桁数が1なので結果の小数点以下の桁数は1+4=5となり「1.36667」となります。

右辺の小数点以下の桁数は関係ない点に注意して下さい。例えば「4 / 3.335」の場合は右辺の数値の小数点以下の桁数が3ですが、左辺の数値の小数点以下の桁数が0なので結果の小数点以下の桁数は0+4=4となり「1.1994」となります。

**DIV**

「DIV」は左辺の値を右辺の値を割ったものを返します。

mysql> select 6 DIV 2, 4 DIV 3, 5.4 DIV 3, 5.7 DIV 3, -5.49 DIV 3, -5.5 DIV 3;



「/」との違いは結果として整数の値を返してくるという点です。

どのような計算方法をしているのかはっきりと分からないので推測なのですが、除算した結果の整数部分を取り出したりしているのではなく、まず左辺と右辺の値を整数に変換した後で除算を行い、その結果の整数部分だけを取得しているように思われます。

そして左辺と右辺の値を整数に変換する時には四捨五入のような計算が行われると推測しています。

いずれにしてもこの演算子の詳細については、はっきりととしたことが分かりましたら書き直します。

**剰余**

剰余を表す「%」は左辺の値を右辺の値を割ったものを返します。

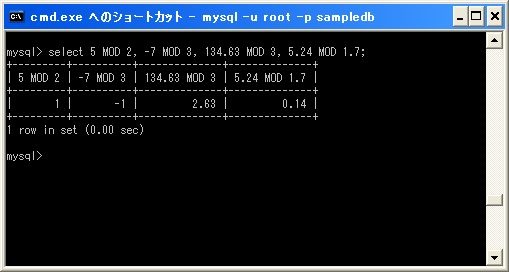
mysql> select 5 % 2, -7 % 3, 134.63 % 3, 5.24 % 1.7;



整数及び浮動小数点数のいずれであっても利用できます。

なお「%」の代わりに「MOD」を使っても同じ結果となります。

mysql> select 5 MOD 2, -7 MOD 3, 134.63 MOD 3, 5.24 MOD 1.7;



**比較演算子**

広告

比較演算子は値と値が等しいかどうか、大きいかどうかなど値を比較するために使用します。MySQLで利用可能な比較演算子は次の通りです。

|  |  |  |
| --- | --- | --- |
| **演算子** | **使用例** | **意味** |
| = | a = b | a と b は等しい |
| <=> | a <=> b | a と b は等しい(NULL対応) |
| <> | a <> b | a と b は等しくない |
| != | a != b | a と b は等しくない |
| < | a < b | a は b よりも小さい |
| <= | a <= b | a は b よりも小さいか等しい |
| > | a > b | a は b よりも大きい |
| >= | a >= b | a は b よりも大きいか等しい |

比較演算子の結果は1(=TRUE)、0(=FALSE)、またはNULLのいずれかを返します。(TRUEとFALSEは定数でそれぞれ1と0が割り当てられています)。

また下記のような演算子も用意されています。

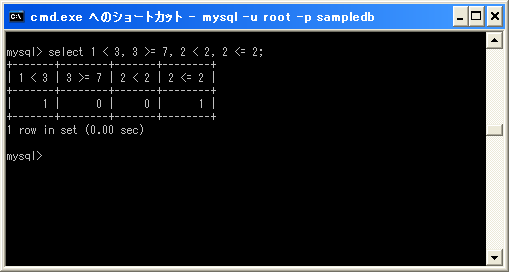
|  |  |  |
| --- | --- | --- |
| **演算子** | **使用例** | **意味** |
| IS NULL | a IS NULL | a はNULL |
| IS NOT NULL | a IS NOT NULL | a はNULLではない |
| IS | a IS boolean\_value | a はboolean\_value |
| IS NOT | a IS NOT boolean\_value | a はboolean\_valueではない |
| BETWEEN AND | a BETWEEN min AND max | a は minより大きいか等しくmaxより小さいか等しい |
| NOT BETWEEN AND | a NOT BETWEEN min AND max | NOT (a BETWEEN min AND max)のこと |
| IN | a IN (value,...) | a は 括弧内のいずれかの値に等しい |

では一つ一つ確認していきます。

**大小の比較**

大きさを比較する関係演算子は「<」「<=」「>」「>=」の4つが用意されています。それぞれ「小さい」「以下」「大きい」「以上」を判定しています。

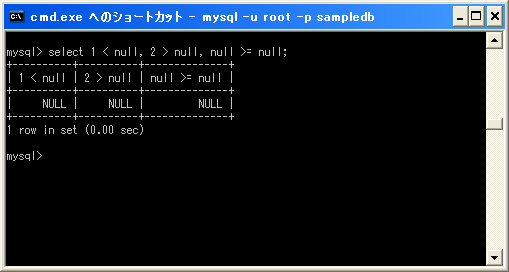
mysql> select 1 < 3, 3 >= 7, 2 < 2, 2 <= 2;



大小の評価を行い、真の場合は1を偽の場合には0を返します。

いずれかの値が1つでもNULLの場合は結果としてNULLを返します。

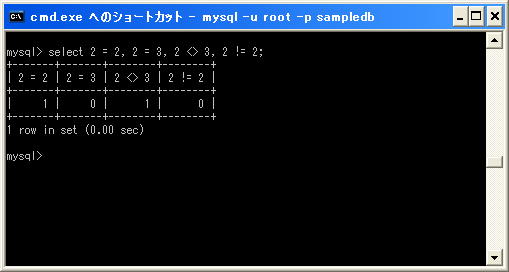
mysql> select 1 < null, 2 > null, null >= null;



**等しいかどうかの比較**

等しいかどうかを判定する関係演算子は「=」「<=>」「<>」「!=」の2つが用意されています。それぞれ「等しい」「等しい」「等しくない」「等しくない」を判定しています。

mysql> select 2 = 2, 2 = 3, 2 <> 3, 2 != 2;



等しいかどうかを評価し、真の場合は1を偽の場合には0を返します。

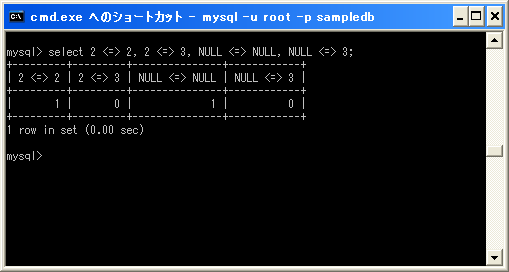
なおNULLはNULL自身を含む何の値と比較してもNULLとなります。

mysql> select NULL = 2, NULL = NULL, NULL <> NULL, NULL != 3;



NULLがNULLと等しいかどうかを含めて等しいかどうかを調べるには「=」ではなく「<=>」を使用します。

mysql> select 2 <=> 2, 2 <=> 3, NULL <=> NULL, NULL <=> 3;

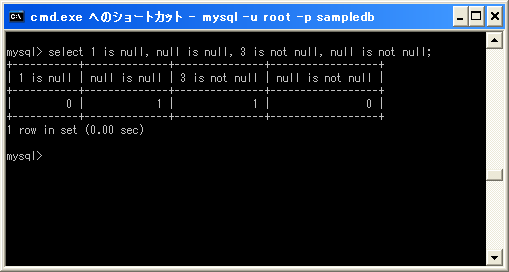


「<=>」演算子の場合、左辺及び右辺にNULLが無い場合は「=」演算子と同じです。そしてNULLをNULLと比較すると1となり、NULLをNULL以外の値と比較すると0となります。

**IS NULL/IS NOT NULL**

値がNULLかどうかを判定します。IS NULLでは対象の値がNULLなら1を返します。IS NOT NULLでは対象の値がNULL以外なら1を返します。

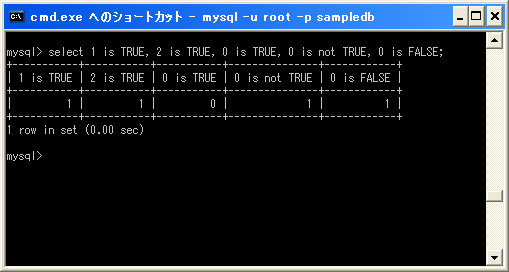
mysql> select 1 is null, null is null, 3 is not null, null is not null;



**IS/IS NOT**

値をboolean型の値と比較します。boolean型の値はTRUE、FALSE、UNKNOWNの3つです。

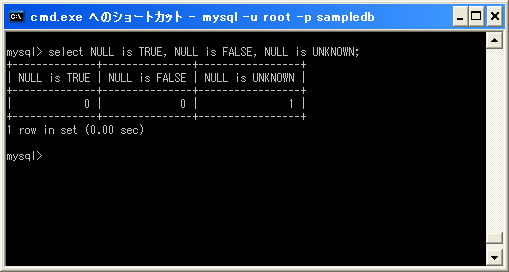
mysql> select 1 is TRUE, 2 is TRUE, 0 is TRUE, 0 is not TRUE, 0 is FALSE;



0以外の値をTRUEと比較するとISの場合は1を返し、IS NOTの場合は0を返します。また0をFALSEと比較するとISの場合は1を返し、IS NOTの場合は0を返します。

またNULLはUNKNOWNと比較した場合にISの場合は1を返し、IS NOｔの場合は0を返します。

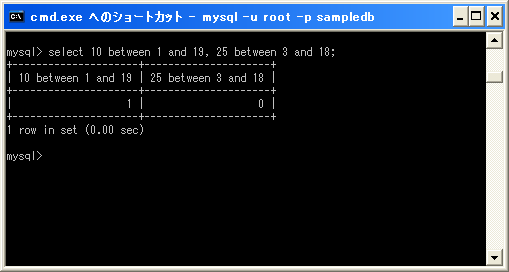
mysql> select NULL is TRUE, NULL is FALSE, NULL is UNKNOWN;



**BETWEEN AND/NOT BETWEEN AND**

値が最小値よりも大きいか等しく、かつ最大値よりも小さいか等しいかどうかを比較します。

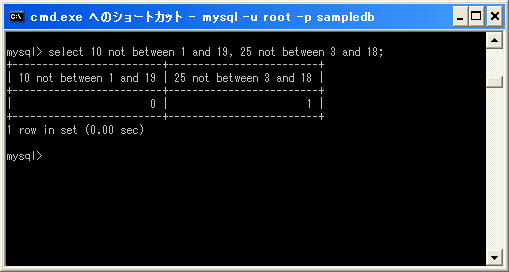
mysql> select 10 between 1 and 19, 25 between 3 and 18;



値を数値だと考えれば、対象の値が最小値以上最大値以下の場合に1、そうでない場合に0を返します。

NOT BETWEEN ANDの場合は、BETWEEN ANDの結果を反転した結果となり0なら1、1なら0を返します。

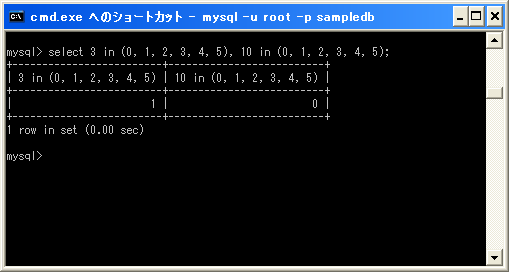
mysql> select 10 not between 1 and 19, 25 not between 3 and 18;



**IN/NOT IN**

値が別途記述した値のリストの中のいずれかに等しいかどうかを判定します。

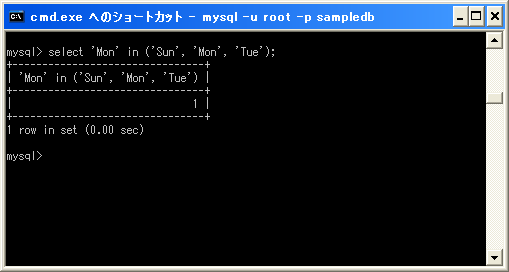
mysql> select 3 in (0, 1, 2, 3, 4, 5), 10 in (0, 1, 2, 3, 4, 5);



値がリストの中のいずれかの値と等しければ1を返し、そうでなければ0を返します。

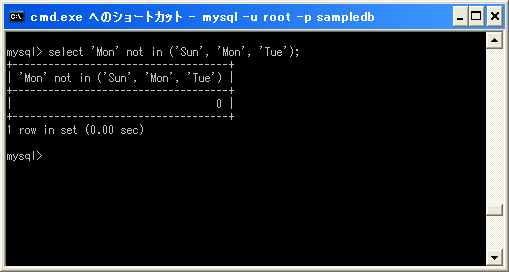
数値だけではなく文字列を比較する場合にも使えます。

mysql> select 'Mon' in ('Sun', 'Mon', 'Tue');



NOT INの場合は、INの結果を反転した結果となり0なら1、1なら0を返します。

mysql> select 'Mon' not in ('Sun', 'Mon', 'Tue');



**論理演算子**

広告

比較演算子を使うことで値と値を比較して真が偽かの判別が行えますが、さらに「AはBと等しいかCよりも大きい」などより複雑な比較を行う為に使用するのが論理演算子です。MySQLで利用可能な論理演算子は次の通りです。

|  |  |  |
| --- | --- | --- |
| **演算子** | **使用例** | **意味** |
| AND | a AND b | a と b が共にTRUEの時にTRUE |
| && | a && b | a と b が共にTRUEの時にTRUE |
| OR | a OR b | a か b の少なくとも1つがTRUEの場合にTRUE |
| || | a || b | a か b の少なくとも1つがTRUEの場合にTRUE |
| XOR | a XOR b | a か b のどちらか1つがTRUEの場合にTRUE |
| NOT | NOT a | a がTRUEならFALSE、a がFALSEならTRUE |
| ! | ! a | a がTRUEならFALSE、a がFALSEならTRUE |

論理演算子では左辺及び右辺に比較演算子を使った式を記述するのが主な利用方法です。それぞれの比較演算子を評価し、さらに論理演算子を使って全体の結果として1(=TRUE)、0(=FALSE)、またはNULLのいずれかを返します。

では一つ一つ確認していきます。

**論理積**

「AND」及び「&&」は論理積です。左辺及び右辺が共にNULLでも0でない場合に全体が1となり、左辺又は右辺のいずれか一つでも0の場合(もう片方がNULLであっても)に全体が0となります。それ以外の場合は全体がNULLとなります。

1 AND 1 -> 1

1 AND 0 -> 0

0 AND 1 -> 0

0 AND 0 -> 0

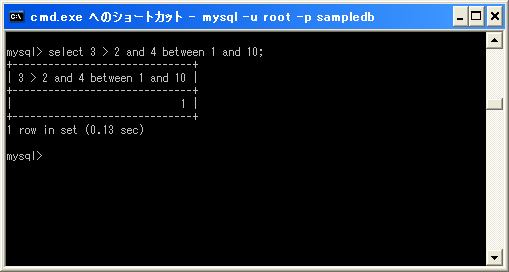
NULL AND 1 -> NULL

NULL AND 0 -> 0

NULL AND NULL -> NULL

では試してみます。

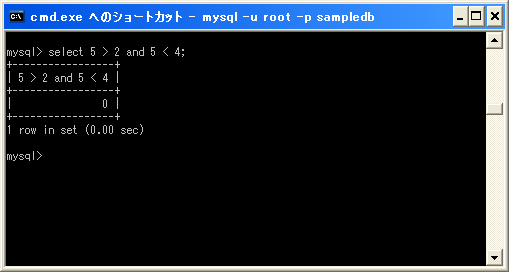
mysql> select 3 > 2 and 4 between 1 and 10;



上記の場合、「3 > 2」が1を返し「4 between 1 and 10」も1を返しますので全体としても1を返します。

もう一つ試します。

mysql> select 5 > 2 and 5 > 4;



上記の場合、「5 > 2」が1を返しますが「5 > 4;」は0を返しますので全体としては0を返します。

**論理和**

「OR」及び「||」は論理和です。左辺及び右辺がNULLでない場合、どちらか1つでも0でない場合に全体が1となり、それ以外は全体が0となります。どちらかがNULLの場合、もう片方が1の場合に全体が1とない、それ以外はNULLとなります。

1 OR 1 -> 1

1 OR 0 -> 1

0 OR 1 -> 1

0 OR 0 -> 0

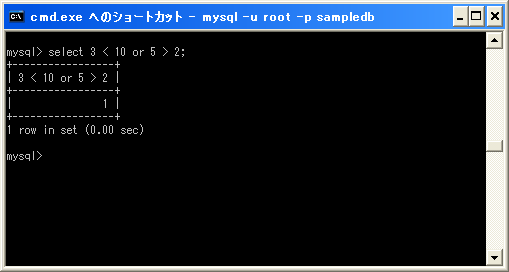
NULL OR 1 -> 1

NULL OR 0 -> NULL

NULL OR NULL -> NULL

では試してみます。

mysql> select 3 < 10 or 5 > 2;



上記の場合、「3 < 10」は0を返しますが「5 > 2」が1を返しますので全体としては1を返します。

**排他的論理和**

「XOR」は排他的論理和です。左辺及び右辺がNULLでない場合、どちらか1つだけが0の場合に全体が1となります。両方が0、又は両方が0でない場合は全体が0となります。どちらか一つでもNULLの場合、全体がNULLとなります。

1 XOR 1 -> 0

1 XOR 0 -> 1

0 XOR 1 -> 1

0 XOR 0 -> 0

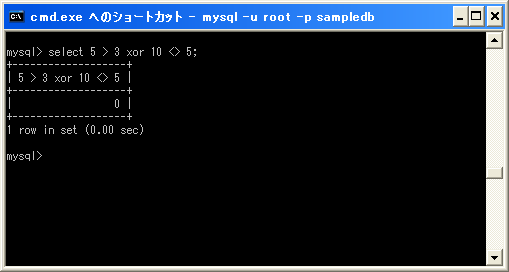
NULL XOR 1 -> NULL

NULL XOR 0 -> NULL

NULL XOR NULL -> NULL

では試してみます。

mysql> select 5 > 3 xor 10 <> 5;



上記の場合、「5 > 3」は1を返し「10 <> 5」も1を返しますので全体としては0を返します。

**論理否定**

「NOT」と「!」は論理否定です。右辺が0の場合に全体が1となり、右辺が0以外の場合に1となります。右辺がNULLの場合は全体もNULLとなります。

NOT 1 -> 0

NOT 0 -> 1

NOT NULL -> NULL

では試してみます。

mysql> select not 2009 in (2007, 2008, 2009);



上記の場合、「2009 in (2007, 2008, 2009)」は1を返しますので全体としては0を返します。

**ビット演算子**

広告

整数の値に対してビット単位で処理を行うために用意されているのがビット演算子です。次の演算子が用意されています。

<td|< td="" style="color: rgb(51, 51, 51); font-family: 'Hiragino Kaku Gothic Pro', 'ヒラギノ角ゴ Pro W4', メイリオ, Osaka, 'ＭＳ Ｐゴシック ', sans-serif; font-size: medium; font-style: normal; font-variant: normal; font-weight: normal; letter-spacing: normal; line-height: normal; orphans: auto; text-align: start; text-indent: 0px; text-transform: none; white-space: normal; widows: 1; word-spacing: 0px; -webkit-text-stroke-width: 0px;"></td|<>

|  |  |  |
| --- | --- | --- |
| **演算子** | **使用例** | **意味** |
| & | a & b | a と b のビットAND |
| a | b | a と b のビットOR |  |
| ^ | a ^ b | a か b のビットXOR |
| ~ | ~ a | a のビットNOT |
| << | a << b | a を左へ bビット分シフト |
| >> | a >> b | a を右へ bビット分シフト |

MySQLではビット演算に BIGINT(64ビット) 演算を使用し、結果は符合なしの 64 ビット整数となります。例えば14という値は64ビットで表すと次のようになります。

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00001110

では一つ一つ確認していきます。

**ビットAND**

ビットANDは演算子の左辺と右辺の同じ位置にあるビットを比較して、両方のビットが共に1の場合だけ1にします。次の例を見てください。

85 & 15

00000000 00000000 00000000 00000000 00000000 00000000 00000000 01010101 = 85

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00001111 = 15

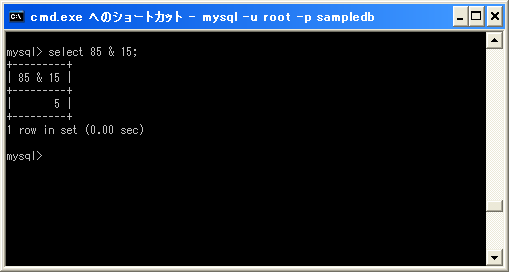
-----------------------------------------------------------------------

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000101 = 5

85と15のビットANDを取ると15を取得します。

では試してみます。

mysql> select 85 & 15;



**ビットOR**

ビットORは演算子の左辺と右辺の同じ位置にあるビットを比較して、どちらか一つでもビット1の場合に1にします。次の例を見てください。

85 | 15

00000000 00000000 00000000 00000000 00000000 00000000 00000000 01010101 = 85

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00001111 = 15

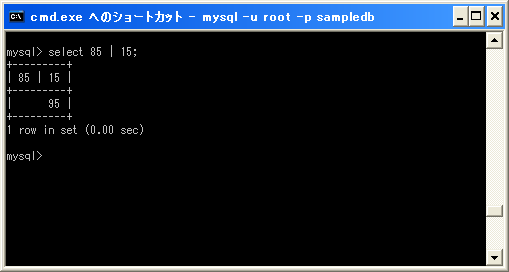
-----------------------------------------------------------------------

00000000 00000000 00000000 00000000 00000000 00000000 00000000 01011111 = 95

85と15のビットORを取ると95を取得します。

では試してみます。

mysql> select 85 | 15;



**ビットXOR**

ビットXORは演算子の左辺と右辺の同じ位置にあるビットを比較して、どちらか一つがビット1の場合に1にします。両方とも1の場合や両方とも0の場合は0にします。次の例を見てください。

85 | 15

00000000 00000000 00000000 00000000 00000000 00000000 00000000 01010101 = 85

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00001111 = 15

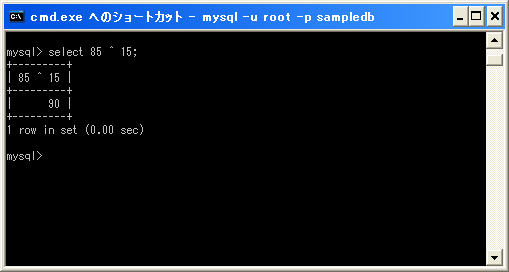
-----------------------------------------------------------------------

00000000 00000000 00000000 00000000 00000000 00000000 00000000 01011010 = 90

85と15のビットXORを取ると90を取得します。

では試してみます。

mysql> select 85 ^ 15;



**ビットNOT**

ビットNOTは演算子の右辺の値の各ビットを反転(0なら1に、1なら0にする)させます。次の例を見てください。

~85

00000000 00000000 00000000 00000000 00000000 00000000 00000000 01010101 = 85

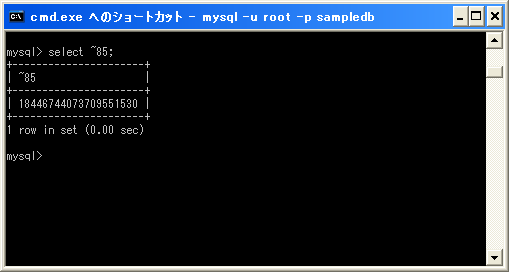
-----------------------------------------------------------------------

11111111 11111111 11111111 11111111 11111111 11111111 11111111 10101010 = 18446744073709551530

85のビットNOTを取ると18446744073709551530を取得します。

では試してみます。

mysql> select ~85;



**左シフト**

<<演算子は対象の値を指定した数だけ左へシフトします。次の例を見てください。

85 << 2

00000000 00000000 00000000 00000000 00000000 00000000 00000000 01010101 = 85

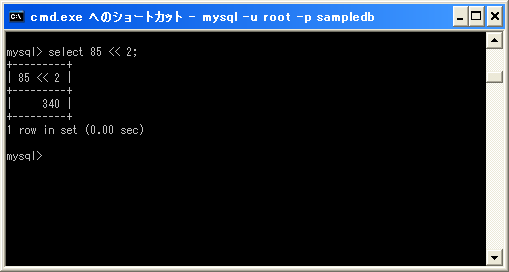
-----------------------------------------------------------------------

00000000 00000000 00000000 00000000 00000000 00000000 00000001 01010100 = 340

85を2ビット分左へシフトすると340を取得します。(1ビット左へシフトすると2倍することに相当します)。

では試してみます。

mysql> select 85 << 2;



**右シフト**

>>演算子は対象の値を指定した数だけ右へシフトします。次の例を見てください。

85 >> 2

00000000 00000000 00000000 00000000 00000000 00000000 00000000 01010101 = 85

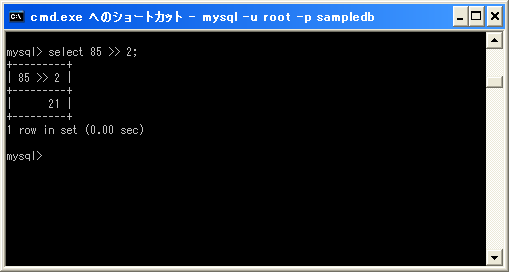
-----------------------------------------------------------------------

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00010101 = 21

85を2ビット分右へシフトすると21を取得します。(1ビット右へシフトすると1/2倍することに相当します)。

では試してみます。

mysql> select 85 >> 2;



**演算子の優先順位**

広告

様々な演算子を組み合わせて使用する場合、どの演算子から評価を行っていくのかで結果は異なってきます。

次の例を見て下さい。

SELECT 10 + 5 \* 4;

この場合、先に「10 + 5」が演算されると次に「15 \* 4」を演算され全体として60になります。逆に「5 \* 4」が先に演算されると次に「10 + 20」が演算され全体として30になります。

演算子には優先順位が定められています。複数の演算子を組み合わせている場合にはより優先順位の高い演算子から演算されることになっています。

演算子の優先順位は次の通りです。

優先順位 高い

------------------------------------------------

!

-(符号のマイナス) ~

^

\* / DIV % MOD

- +

<< >>

&

|

= <=> >= > <= < <> != IS IN

BETWEEN

NOT

&& AND

|| OR XOR

:=

------------------------------------------------

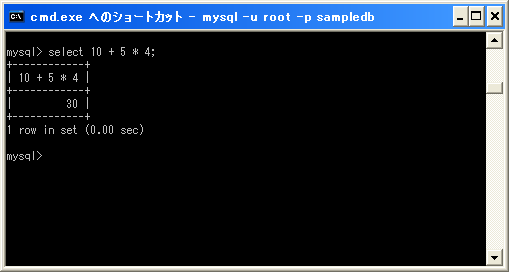
優先順位 低い

上に書かれているほど優先順位が高く、下になるほど優先順位は低くなります。同じ行に書かれている演算子は優先順位に違いはありません。

例として「\*」と「+」を見てください。「\*」の方が「+」よりも上に書かれていますので「\*」の方が優先順位が高くなっています。その為、同じ式の中で「\*」と「+」の2つの演算子があった場合にはまず先に「\*」を使った演算が行われます。

よって最初の例では「5 \* 4」が先に実行され、その後で「10 + 20」が実行されます。

mysql> select 10 + 5 \* 4;



**括弧を使った優先順位の変更**

複数の演算子が含まれていた場合は演算子の優先順位に従って演算されます。ただ優先順位に関わらず先に演算を行いたい場合もありますが、その場合は優先したい演算を括弧で括って下さい。

次の例を見て下さい。

SELECT 10 + 5 \* 4;

「+」と「\*」では優先順位が「\*」の方が高いため先に「5 \* 4」が演算されることになりますが、括弧を使うことで優先順位を変えることができます。

SELECT (10 + 5) \* 4;

括弧で囲まれた中の演算は、優先順位に関係なく優先的に行われます。よって先に「10 + 5」が演算され、次に「15 \* 4」が演算されることになります。

括弧の中に複数の演算子が含まれていた場合には、括弧の中だけで演算子の優先順位によって演算される順番が決まります。

SELECT (10 + 8 / 2) \* 4;

上記の場合はまず括弧の中の「10 + 8 / 2」が演算されますが、この中には「+」と「/」の二つの演算子が含まれているため優先順位から先に「8 / 2」が演算されます。

また括弧の中にさらに括弧を記述することもできます。

SELECT ((10 + 8) / 2) \* 4;

括弧の中に括弧がある場合でも考え方は同じです。外側の括弧の中にある「(10 + 8) / 2」が演算されますが、この式には括弧が含まれているため括弧の中の「10 + 8」がまず最初に演算されます。

では実際に試してみます。

mysql> select 10 + 8 / 2 \* 4, (10 + 8 / 2) \* 4, ((10 + 8) / 2) \* 4;



括弧を使うことで演算の順序が変わっていることが確認できます。

**MySQLのデータ型**

広告

ここではMySQLで利用可能なデータ型について確認していきます。なお全てのデータ型ではなく比較的よく利用するデータ型だけを選んでいますので、ここで記載したデータ型以外にもMySQLにはデータ型が存在します。

* **MySQLのデータ型**
* [整数型](http://www.dbonline.jp/mysql/type/index1.html)
* [浮動小数点数型](http://www.dbonline.jp/mysql/type/index2.html)
* [日付型と時刻型](http://www.dbonline.jp/mysql/type/index4.html)
* [CHAR型とVARCHAR型](http://www.dbonline.jp/mysql/type/index3.html)
* [BINARY型とVARBINARY型](http://www.dbonline.jp/mysql/type/index5.html)
* [BLOB型とTEXT型](http://www.dbonline.jp/mysql/type/index6.html)
* [ENUM型](http://www.dbonline.jp/mysql/type/index7.html)
* [SET型](http://www.dbonline.jp/mysql/type/index8.html)

**整数型**

広告

ここでは数値型の中でも整数型について確認していきます。

整数型は次の5つが用意されています。

TINYINT

-128から127 (符号無しの場合0から255)

SMALLINT

-32768から32767 (符号無しの場合0から65535)

MEDIUMINT

-8388608から8388607 (符号無しの場合0から16777215)

INT

-2147483648から2147483647 (符号無しの場合0から4294967295)

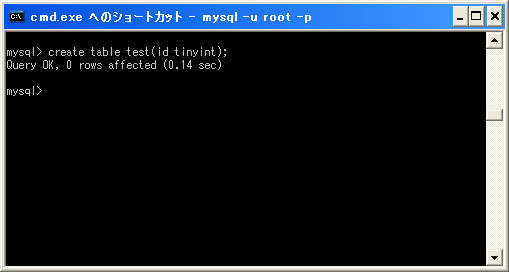
別名：INTEGER

BIGINT

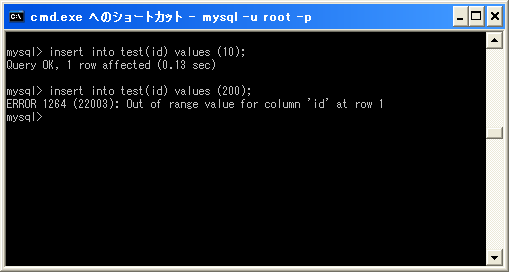
-9223372036854775808から9223372036854775807

(符号無しの場合0から18446744073709551615)

例としてTINYINT型のカラムを1つ持つテーブルを作成してみます。



整数型は5種類用意されていますが、各データ型には格納できる値の範囲が決まっています。範囲を超えた値を格納しようとするとエラーが発生します。



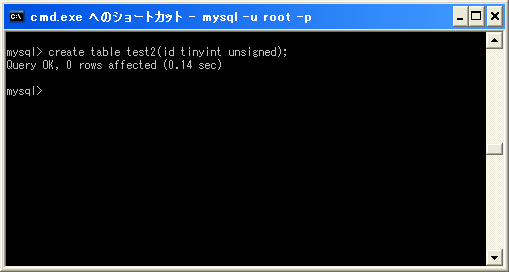
TINYINT型は格納可能な範囲は-128から127です。その為、100を格納する場合は問題ありませんが200を格納しようとすると範囲を超えているためエラーとなります。

**UNSIGNED**

各値は正の数と負の数を扱うことができますが、UNSIGNEDを付けると正の数しか格納できなくなります。その代わり範囲は2倍となります。使い方は次の通りです。

TINYINT UNSIGNED

例としてTINYINT UNSIGNED型のカラムを1つ持つテーブルを作成してみます。



データを追加してみます。



TINYINT型は格納可能な範囲は-128から127でしたがTINYINT UNSIGNED型は格納可能な範囲が0から255です。その為、200を格納する場合は問題ありませんが-10を格納しようとすると範囲を超えているためエラーとなります。

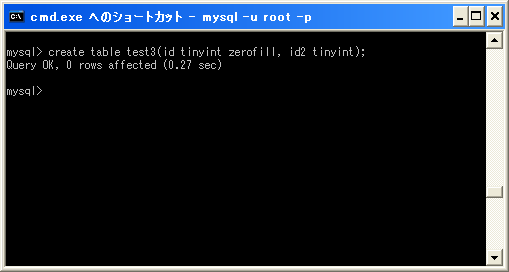
**ZEROFILL**

データ型にZEROFILLを付けるとデータ型の桁数分だけ0で埋められます。使い方は次の通りです。

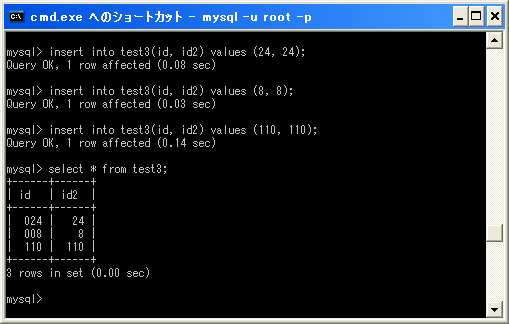
TINYINT ZEROFILL

※ZEROFILLを付けると自動的にUNSIGNEDが付きます

例としてTINYINT ZEROFILL型とTINYINT型のカラムを持つテーブルを作成してみます。



データを追加して後でテーブルのデータを取得してみます。



TINYINT型は3桁の数値を格納しますので、TINYINT ZEROFILL型の方のカラムは1桁や2桁の数値を格納された場合は3桁分まで0で埋められます。TINYINT型の方は桁数分だけ空白で埋められます。

**表示幅**

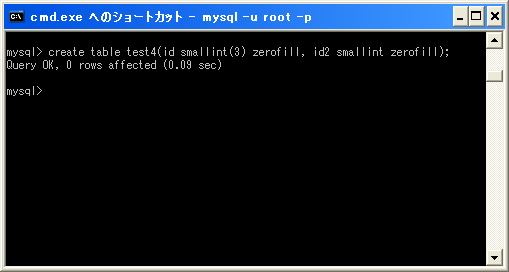
各データ型は表示幅を設定することができます。使い方は次の通りです。

TINYINT(N)

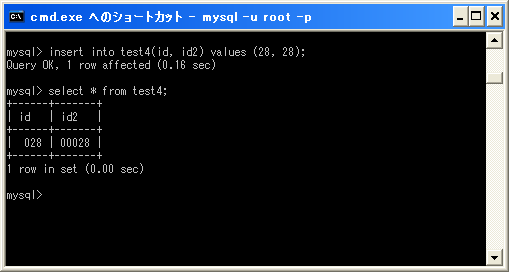
Nには幅を表す数値を指定して下さい。

幅を指定すると指定した幅で数値が表示されます。ただ、本来使用可能な桁数よりも小さい表示幅を指定した場合に、表示幅よりも大きい値を指定すると表示幅の設定は無視されます。

例としてSMALLINT(3) ZEROFILL型とSMALLINT ZEROFILL型のカラムを持つテーブルを作成してみます。

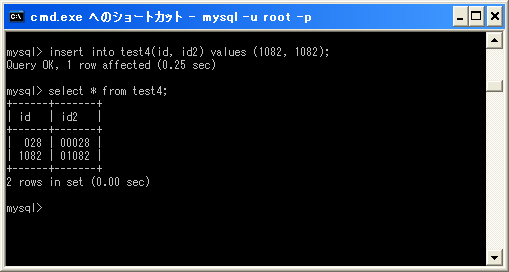


データを追加して後でテーブルのデータを取得してみます。



SMALLINT(3) ZEROFILL型は表示幅が3桁となっていますので3桁の数値として表示されます。SMALLINT型は5桁の型ですので表示幅を指定していない方は5桁の数値として表示されます。

ただ表示幅を指定していても格納可能な数値の範囲が3桁になったわけではありません。4桁以上の数値を格納することもできますし、表示する場合は4桁以上の数値は表示幅の設定に関係無く表示されます。



**浮動小数点数型**

広告

ここでは数値型の中でも浮動小数点数型について確認していきます。

浮動小数点数型は次の2つが用意されています。

FLOAT

-3.402823466E+38 から -1.175494351E-38

0

1.175494351E-38 から 3.402823466E+38

DOUBLE

-1.7976931348623157E+308 から -2.2250738585072014E-308

0

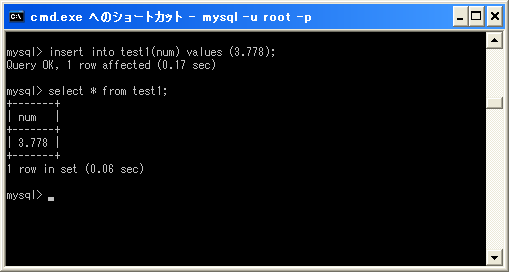
2.2250738585072014E-308 から 1.7976931348623157E+308

別名：DOUBLE PRECISION, REAL

ではDOUBLE型のカラムを持つテーブルを作成します。



整数型と同じく浮動小数点数型にも各データ型には格納できる値の範囲が決まっています。ではデータを追加した後でテーブルのデータを取得してみます。



**桁数と小数点以下の桁数**

FLOAT型とDOUBLE型は精度と小数点以下の桁数を指定することができます。書式は次の通りです。

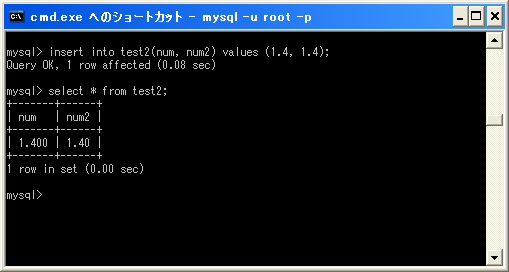
DOUBLE(桁数, 小数点以下の桁数)

桁数は全体の桁数で、小数点以下の桁数は桁数の中で小数点以下の桁数です。例えば全体が5桁で小数点以下の桁数が3桁のものはdouble(5, 3)と記述し、34.567のように表示されます。

例としてDOUBLE(4,3)型のカラムとDOUBLE(4,2)型のカラムを持つテーブルを作成してみます。

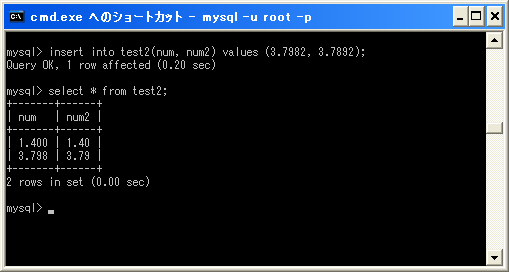


それぞれのカラムに1.4という数値を持つデータを追加した後でテーブルのデータを取得してみます。



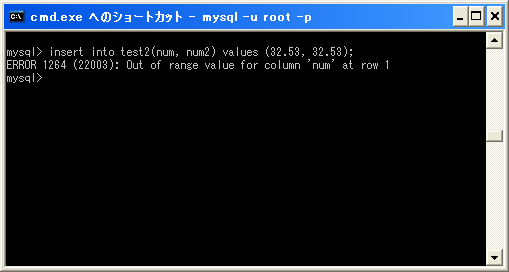
DOUBLE(4,3)型のカラムは全体が4桁で小数点以下3桁ですので1.400と表示されます。またDOUBLE(4,2)型のカラムは全体が4桁で小数点以下が2桁ですので1.40と表示されます。

今度はそれぞれのカラムに3.7982という数値を持つデータを追加した後でテーブルのデータを取得してみます。



格納しようとした値は小数点以下が4桁の数値です。各カラムのデータ型の小数点以下の桁数は3桁と2桁ですので指定した桁数までの値が格納されます。

最後はそれぞれのカラムに32.53という数値を持つデータを追加してみます。



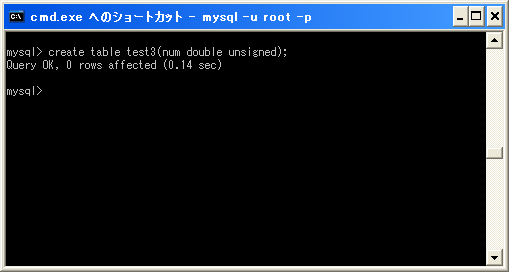
DOUBLE(4,3)型のカラムは全体が4桁で小数点以下3桁のため、整数部分が1桁です。そのカラムに整数部分が2桁の値を格納しようとした為にエラーとなりました。

**UNSIGNED**

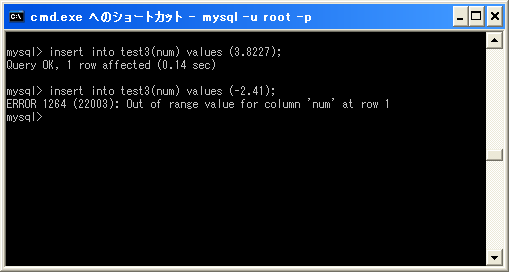
各値は正の数と負の数を扱うことができますが、UNSIGNEDを付けると正の数しか格納できなくなります。注意して頂きたいのは整数と異なり浮動小数点数はUNSIGNEDを付けても範囲は変わりません。使い方は次の通りです。

DOUBLE UNSIGNED

例としてDOUBLE UNSIGNED型のカラムを1つ持つテーブルを作成してみます。



データを追加してみます。



正の値は格納できますが、負の値を格納しようとするとエラーとなります。

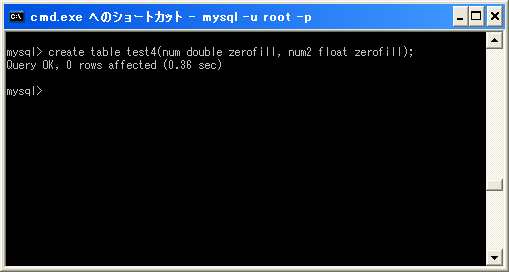
**ZEROFILL**

データ型にZEROFILLを付けるとデータ型の桁数分だけ0で埋められます。使い方は次の通りです。

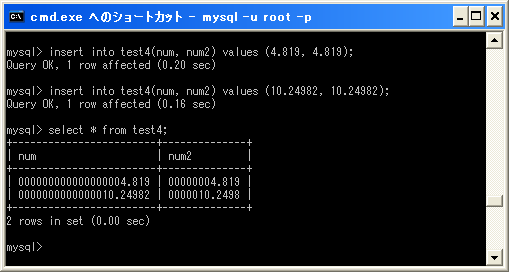
DOUBLE ZEROFILL

※ZEROFILLを付けると自動的にUNSIGNEDが付きます

例としてDOUBLE ZEROFILL型とFLOAT ZEROFILL型のカラムを持つテーブルを作成してみます。



データを追加して後でテーブルのデータを取得してみます。



桁数分だけ0で埋められて表示されます。

**日付型と時刻型**

広告

ここでは日付型と時刻型について確認していきます。次のデータ型が用意されています。

DATE

フォーマット : 'YYYY-MM-DD'

範囲 : '1000-01-01' から '9999-12-31'

DATETIME

フォーマット : 'YYYY-MM-DD HH:MM:SS'

範囲 : '1000-01-01 00:00:00' から '9999-12-31 23:59:59'

TIMESTAMP

フォーマット : 'YYYY-MM-DD HH:MM:SS'

範囲 : '1970-01-01 00:00:01' から '2037-12-31 23:59:59'

TIME

フォーマット : 'HH:MM:SS'

範囲 : '-838:59:59' から '838:59:59'

YEAR[(2|4)]

フォーマット : YYYY

範囲 : 1901 から 2155、そして 0000 (4桁の場合)

範囲 : 70 から 69 (2桁の場合、70は1970を表し69は2069を表す)

デフォルトは4桁

こららのデータ型は日付と時刻を格納するために使われます。

**DATE型**

日付を文字列として指定する場合、'YYYY-MM-DD'が基本となるフォーマットですが、次のような指定方法も許可されています。

フォーマット 入力例

----------------------- -------------------------

'YYYY-MM-DD' '2009-09-07'

'YY-MM-DD' '09-09-07'

'YYYYMMDD' '20090907'

'YYMMDD' '090907'

また区切り文字は'-'だけではなく次のような文字を使うこともできます。

フォーマット 入力例

----------------------- -------------------------

'YYYY-MM-DD' '2009-09-07'

'YYYY.MM.DD' '2009.09.07'

'YYYY/MM/DD' '2009/09/07'

'YYYY@MM@DD' '2009@09@07'

日付に区切り文字が使われている場合、1月から9月、そして1日から9日については2桁で表す必要はありません。

フォーマット 入力例

----------------------- -------------------------

'YYYY-M-D' '2009-9-7'

日付は数値として指定することもできます。

フォーマット 入力例

----------------------- -------------------------

YYYYMMDD 20090907

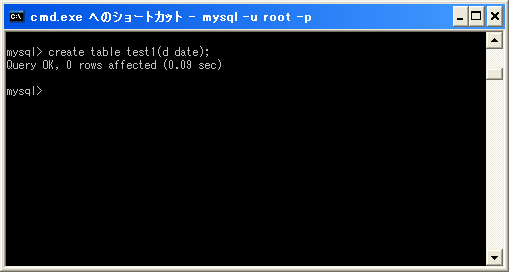
YYMMDD 090907

なお年が2桁で指定された場合は次のルールによって4桁の年に変換します。

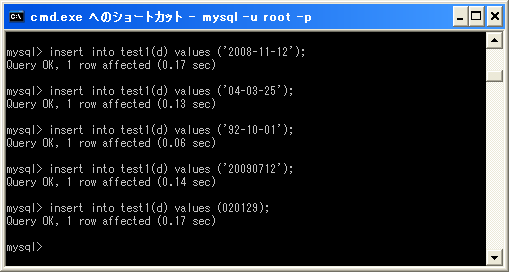
70-99 の範囲の年の値は 1970-1999

00-69 の範囲の年の値は 2000-2069

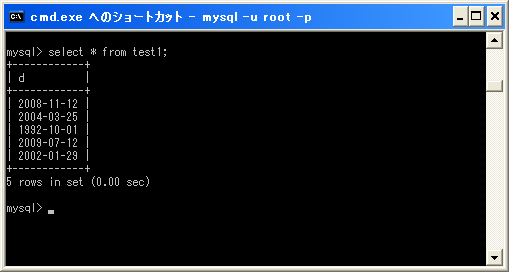
ではDATE型のカラムを持つテーブルを作成します。



様々なフォーマットでデータを追加してみます。



追加したデータを取得すると次のように表示されます。



**TIME型**

時刻を文字列として指定する場合、'HH:MM:SS'が基本となるフォーマットですが、次のような指定方法も許可されています。

フォーマット 入力例

----------------------- -------------------------

'HH:MM:SS' '15:25:07'

'HH:MM' '15:25'

'HHMMSS' '152507'

'MMSS' '2507'

'SS' '07'

'D HH:MM:SS' '2 15:25:07'

'D HH:MM' '2 15:25'

'D HH'' '2 15'

'MMSS'で指定した場合は分と秒になり'SS'で指定した場合は秒となりますが、'HH:MM'のように区切り文字がある場合は時と分になりますので注意して下さい。

'D'は日にちを表します。'D'付きの場合はD×24が時に加算されて表示されます。

時間に区切り文字が使われている場合、0秒から9秒、0分から9分、0時から9時については2桁で表す必要はありません。

フォーマット 入力例

----------------------- -------------------------

'HH:M:S' '15:3:7'

時間は数値として指定することもできます。

フォーマット 入力例

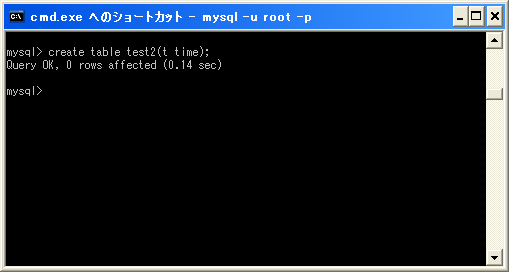
----------------------- -------------------------

HHMMSS 152507

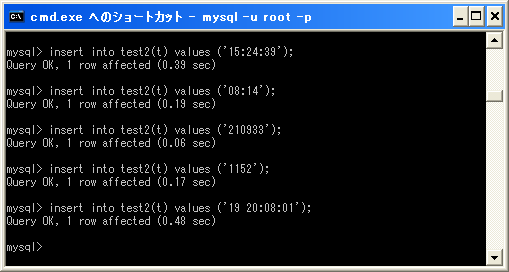
MMSS 2507

SS 07

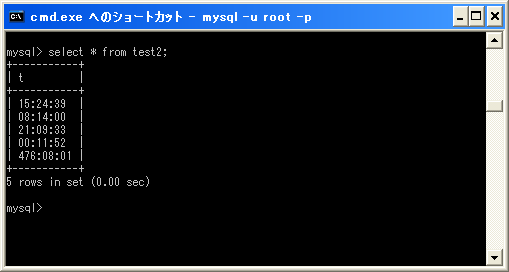
ではTIME型のカラムを持つテーブルを作成します。



では様々なフォーマットでデータを追加してみます。



追加したデータを取得すると次のように表示されます。



**DATETIME型 / TIMESTAMP型**

日付と時刻を文字列として指定する場合、'YYYY-MM-DD HH:MM:SS'が基本となるフォーマットですが、次のような指定方法も許可されています。

フォーマット 入力例

----------------------- -------------------------

'YYYY-MM-DD HH:MM:SS' '2009-10-04 15:25:07'

'YY-MM-DD HH:MM:SS' '09-10-04 15:25:07'

'YYYYMMDDHHMMSS' '20091004152507'

'YYMMDDHHMMSS' '091004152507'

また区切り文字は'-'だけではなく次のような文字を使うこともできます。

フォーマット 入力例

----------------------- -------------------------

'YYYY-MM-DD HH:MM:SS' '2009-10-04 15:25:07'

'YYYY.MM.DD HH+MM+SS' '2009.10.04 15+25+07'

'YYYY/MM/DD HH\*MM\*SS' '2009/10/04 15\*25\*07'

'YYYY@MM@DD HH^MM^SS' '2009@10@04 15^25^07'

日付や時刻に区切り文字が使われている場合、1桁の数値を2桁で表す必要はありません。

フォーマット 入力例

----------------------- -------------------------

'YYYY-M-D H:M:S' '2009-3-7 8:2:9'

数値として指定することもできます。

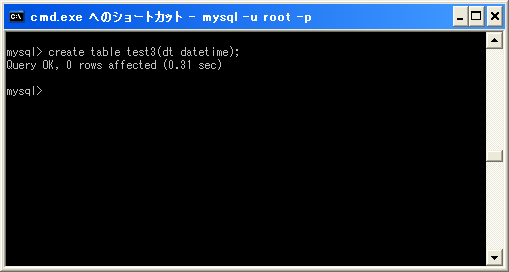
フォーマット 入力例

----------------------- -------------------------

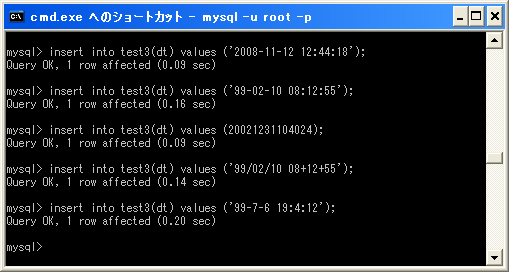
YYYYMMDDHHMMSS 20091004152507

YYMMDDHHMMSS 091004152507

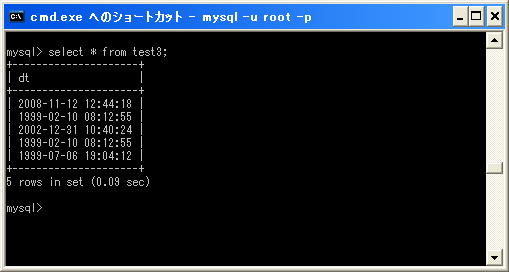
ではDATETIME型のカラムを持つテーブルを作成します。



様々なフォーマットでデータを追加してみます。



追加したデータを取得すると次のように表示されます。



TIMESTAMP型のフォーマットはDATETIME型とほぼ同じですが、使い方などが異なります。TIMESTAMP型についてはまた別途詳しく確認します。

**YEAR型**

年を数値として指定する場合、YYYY が基本となるフォーマットですが、次のような指定方法も許可されています。

フォーマット 入力例

----------------------- -------------------------

YYYY 2009

YY 09

文字列として指定することもできます。

フォーマット 入力例

----------------------- -------------------------

'YYYY' '2001'

'YY' '98'

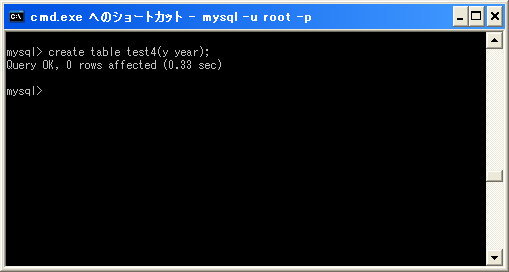
なお年が2桁で指定された場合は次のルールによって4桁の年に変換します。

70-99 の範囲の年の値は 1970-1999

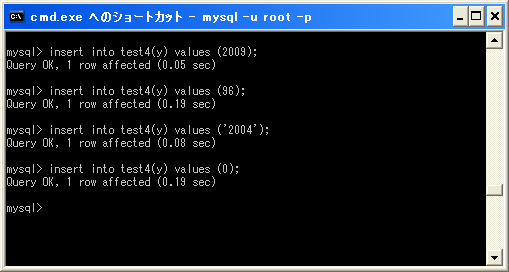
1-69 の範囲の年の値は 2001-2069

DATE型などと異なり0を指定すると2000にはなりませんので注意が必要です。数字の0や文字列の'0'を指定すると0000に変換されます。

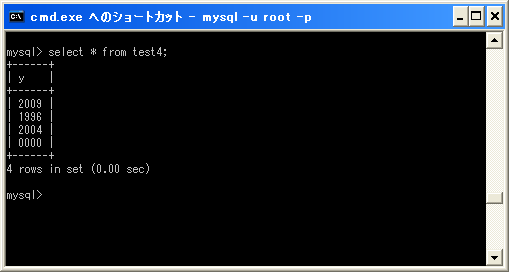
ではYEAR型のカラムを持つテーブルを作成します。



様々なフォーマットでデータを追加してみます。



追加したデータを取得すると次のように表示されます。



**CHAR型とVARCHAR型**

広告

ここでは文字列型の中でもCHAR型とVARCHAR型について確認していきます。

CHAR(M)

固定長文字列

Mは文字数を指定。0から255文字

別名：CHAR(M)はCHARACTER(M)の別名

VARCHAR(M)

可変長文字列

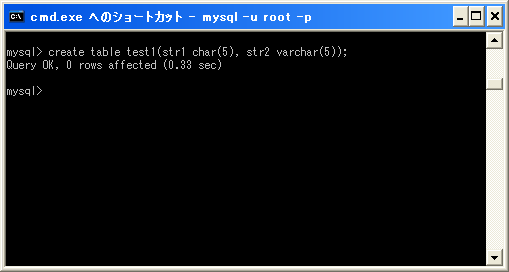
Mはバイト数を指定。0から65535バイト

別名：VARCHAR(M)はCHARACTER VARYING(M)の別名

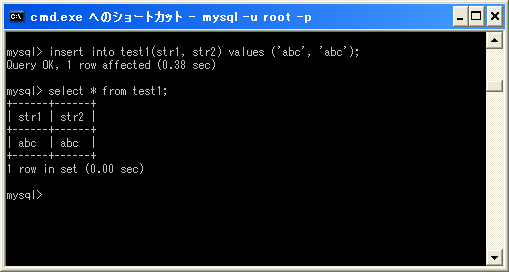
CHAR型は指定した文字数以下の文字を格納した場合には文字列の末尾に空白を必要なだけ付け加えて指定の長さの文字列として格納します。ただし値を取得する場合は末尾にある空白は全て削除された上で取得されますし比較される時も同様です。

それに対してVARCHAR型は末尾に空白を付けるようなことはしません。また現行のバージョンでは末尾に空白がある文字列であっても空白が付いたまま格納されます。取得する時も空白付きで取得しますが、WHERE句で比較する時だけ末尾の空白を削除した上で比較されます。

それではCHAR(5)型のカラムとVARCHAR(5)型のカラムを持つテーブルを作成してみます。

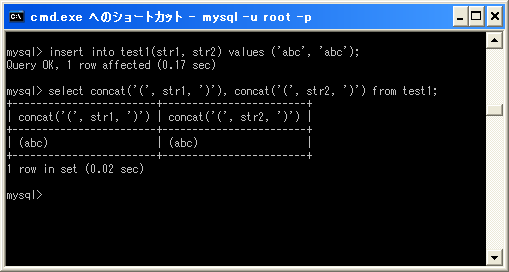


ではデータを追加した後でテーブルのデータを取得してみます。



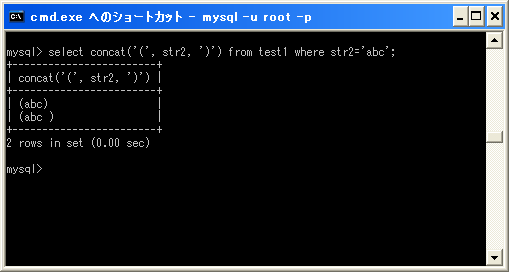
CHAR(5)型は常に5文字の長さになるように空白を使って足りない分を埋めて格納されますが、格納されたデータを取得すると文字列の末尾に付いている空白は削除された上で取得されます。よって今回の場合は同じ結果を取得します。

では今度は各カラムに格納する文字列の最後に空白を加えたデータを追加してみます。



MySQL5.1ではVARCHAR型のカラムに末尾に空白を含む文字列を格納しても末尾の空白は削除されずに格納されます。CHAR型の場合は取得する時に末尾についた空白は削除された上で取得されますが、VARCHAR型の場合は取得する時に末尾についた空白はそのまま取得します。

ただしVARCHAR型のカラムに格納された値であっても、WHERE句で比較する場合は末尾の空白を削除した上で比較されますので注意して下さい。



**文字セットと照合順序**

CHAR型とVARCHAR型はそれぞれ文字セットと照合順序を指定することができます。使い方は次の通りです。

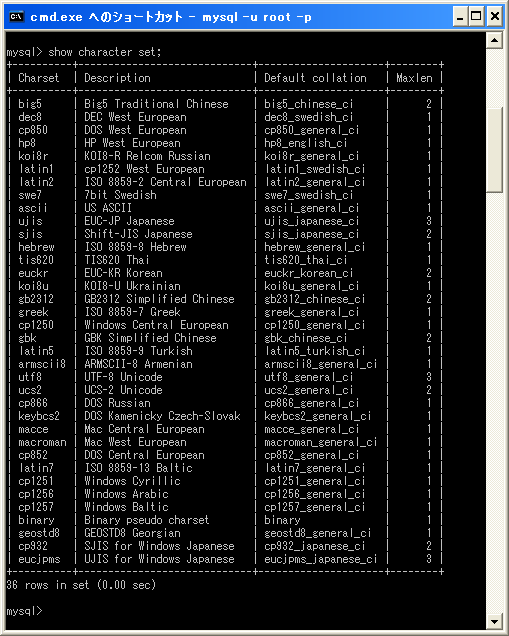
CHAR(M) [CHARACTER SET 文字セット] [COLLATE 照合順序]

VARCHAR(M) [CHARACTER SET 文字セット] [COLLATE 照合順序]

MySQLにはデータベース、テーブル、カラム、文字列リテラルの4つのレベルで文字セットを設定することができます。文字列のデータ型に文字セットを指定した場合、そのデータ型を指定したカラムに対して文字セットを指定することになります。

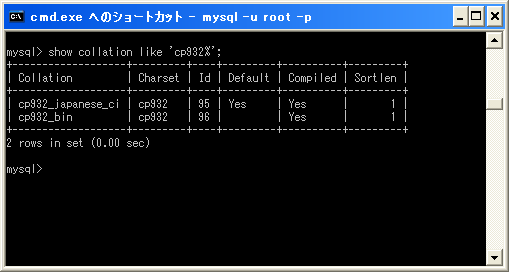
MySQLに設定可能な文字セットは次のSQL文を実行すると取得できます。

mysql> show character set;

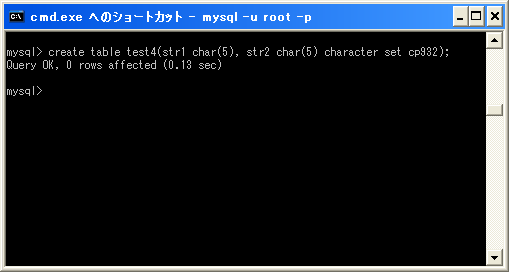


またある文字セットに対して指定可能な照合順序は次のように取得できます。(下記はcp932の対する照合順序です)。

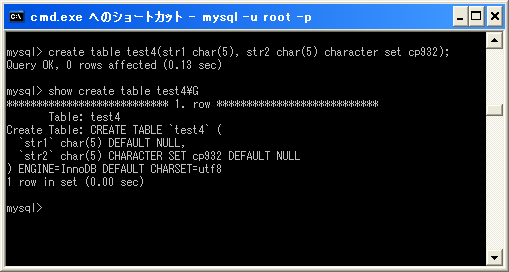
mysql> show collation like 'cp932%';



それではCHAR(5)型のカラムとCHAR(5) CHARACTER SET cp932型のカラムを持つテーブルを作成してみます。



作成したテーブル定義を確認してみます。



カラムに対して文字セットが定義されていることが確認できます。

**ASCII属性/UNICODE属性/BINARY属性**

データ型に対してASCII属性を設定するとCHARACTER SET latin1を設定した場合と同じです。

CHAR(M) ASCII

CHAR(M) CHARACTER SET latin1

データ型に対してUNICODE属性を設定するとCHARACTER SET ucs2 を設定した場合と同じです。

CHAR(M) UNICODE

CHAR(M) CHARACTER SET ucs2

データ型に対してBINARY属性を設定すると比較や照合の時にバイナリデータとして比較されます。つまりCOLLATEがバイナリのものに設定されます。

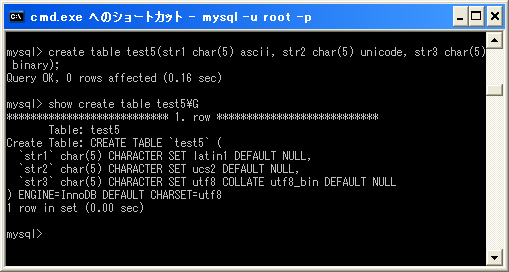
CHAR(M) BINARY

CHAR(M) CHARACTER SET utf8 COLLATE utf8\_bin

※ 上記はデフォルトの文字セットがUTF8の場合です。CHARACTER SETを合わせて指定することもできます。

MySQLでは文字列の値を比較する時に大文字と小文字を区別しません。BINARY属性を設定するとバイナリデータとして比較されますので'A'と'a'は区別されます。

それではCHAR(5) ASCII型のカラムとCHAR(5) UNICODE型とCHAR(5) BINARY型のカラムを持つテーブルを作成しテーブル定義を確認してみます。



各属性を指定した場合に実際にどのようなデータ型として作成されているのか確認できると思います。

**BINARY型とVARBINARY型**

広告

ここでは文字列型の中でもBINARY型とVARBINARY型について確認していきます。

BINARY[(N)]

固定長バイナリバイト文字列

Nはバイトを指定。0から255文字

別名：CHAR BYTE

VARBINARY(M)

可変長バイナリバイト文字列

Mはバイト数を指定。0から65535バイト

BINARY型とVARBINARY型はデータをバイナリデータとして格納します。どちらのデータ型も文字セットなどはなく、送られてきたデータをそのままバイナリデータとして格納します。よって比較や照合もデータの数値をそのまま比較します。

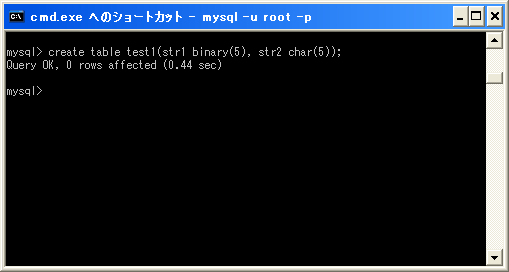
BINARY型に指定したバイト数よりも少ないデータを格納した場合には末尾に0x00を必要なだけ付け指定のバイト数になるように調整されてから格納されます。また取得した時にも削除されずに0x00が付いたまま取得されます。

VARBINARY型は0x00が末尾に付けられることはありません。空白と0x00は異なりますのでデータの末尾に空白が合った場合も特別扱いされることはなく、空白に該当するコードはそのままバイナリデータとして格納されます。

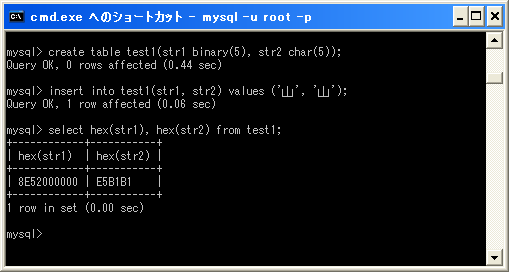
**バイナリデータがどのように格納されるか**

バイナリデータとしてどのように格納されるのかを確認します。クライアントからBINARY型が設定されたカラムに対してデータが追加された場合、追加される文字列をクライアント側で設定されている文字セットでコード化したデータがMySQLサーバ側に送信されます。送られてきたデータはMySQL側のデフォルトの文字セットなどに関係無く変換されずにそのままコードとして格納されます。

例としてBYTE(5)型とCHAR(5)型のカラムを持つテーブルを作成します。



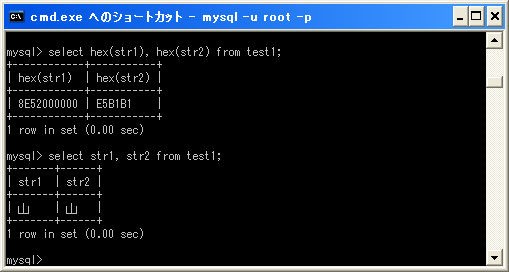
両方のカラムに同じ文字列を指定してデータを追加します。そして追加されたデータを16進数にして取得してみます。



BYTE(5)型のカラムには「8E52」という値が格納されています。今回テストしている環境はクライアント側の文字コードがcp932、MySQLサーバ側がUTF8なのですが、クライアント側の文字セットであるcp932で"山"に対応するコードが「8E52」です。クライアントからは"山"という文字が「8E52」というデータがMySQLサーバへ送信されていますが、BINARY型のカラムに格納する時は、送られたきたデータをそのまま格納します。

それに対してCHAR(5)型のカラムには「E5B1B1」という値が格納されています。この「E5B1B1」はMySQLサーバ側の文字セットであるUTF8で"山"に対応するコードです。CHAR側のカラムに格納する時にクライアント側の文字セットからMySQLサーバ側の文字セットに変換された上で格納されています。

では格納されているデータを取得してみます。



BYTE(5)型のカラムの値は格納されたデータのままクライアントへ返します。送られてきたデータをクライアント側の文字セットで該当する文字に変換すると"山"になります。CHAR(5)型のカラムの値はMySQL側の文字セットからクライアント側の文字セットに変換した上でクライアントへ帰されます。よってこちらも"山"になります。

データを追加したクライアントの文字セットとデータを取得した文字セットがたまたま同じなので文字化けなどは起こりませんが、もし違っていた場合は文字化けする可能性があります。

**BINARY型とBINARY属性の違い**

CHAR型にBINARY属性を付けた場合は、あくまでカラムで使用される文字セットで文字列として保存されたデータを、その格納されている数値を使って比較したり並べ替えを行ったりします。

それに対してBINARY型の場合は、格納されているデータはバイナリデータであって文字列ではありません。その為、比較や並べ替えも格納されているデータの数値を使って行います。

どちらのデータ型が設定されているカラムに対して'a'と'A'を格納した場合、区別して比較されるという点では同じですが根本的な仕組みはまったく異なります。

**BLOB型とTEXT型**

広告

ここでは文字列型の中でもBLOB型とTEXT型について確認していきます。

BLOB型はバイナリデータを扱うデータ型で格納できるデータのサイズを指定しません。ただ最大長は決まっており、最大長の種類によって次の4つが用意されています。

TINYBLOB

最長255 (28 - 1) バイト

BLOB[(M)]

最長65,535 (216 - 1) バイト

MEDIUMBLOB

最長16,777,215 (224 - 1) バイト

LONGBLOB

最長4,294,967,295、または4GB (232 - 1) バイト

TEXT型は文字列データを扱うデータ型で格納できるデータのサイズを指定しません。ただ最大長は決まっており、最大長の種類によって次の4つが用意されています。

TINYTEXT

最長255 (28 - 1) バイト

TEXT[(M)]

最長65,535 (216 - 1) バイト

MEDIUMTEXT

最長16,777,215 (224 - 1) バイト

別名：LONG, LONG VARCHAR

LONGTEXT

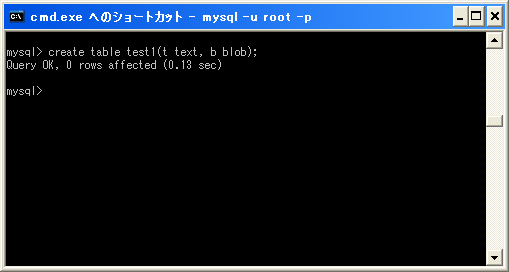
最長4,294,967,295、または4GB (232 - 1) バイト

BLOG型及びTEXT型はそれぞれ巨大なバイナリデータとテキストデータを格納する時に利用されます。

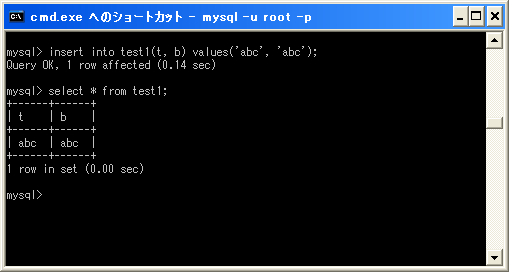
BLOB型はVARBINARY型で格納できるバイト数を指定しなかったものと考えて頂けば結構です。(VARBINARY型については「[BINARY型とVARBINARY型](http://www.dbonline.jp/mysql/type/index5.html)」を参照して下さい)。データはバイナリデータとしてそのまま格納されます。

TEXT型はVARCHAR型で格納できるバイト数を指定しなかったものと考えて頂けば結構です。(VARCHAR型については「[CHAR型とVARCHAR型](http://www.dbonline.jp/mysql/type/index3.html)」を参照して下さい)。データは文字列データとして格納されます。

ではBLOB型のカラムとTEXT型のカラムを持つテーブルを作成してみます。



データを格納した後でデータを取得して表示してみます。



巨大なデータを取り扱えるという点を除けばVARBINARY型とVARCHAR型と同じように扱えます。

**文字セットと照合順序**

TEXT型は文字セットと照合順序を指定することができます。使い方は次の通りです。

TEXT [CHARACTER SET 文字セット] [COLLATE 照合順序]

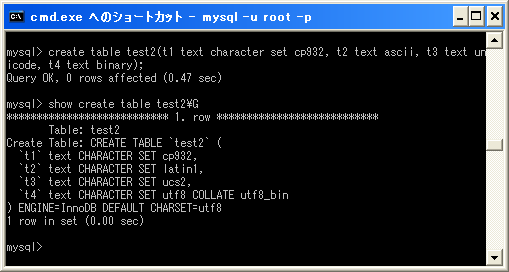
またASCII属性/UNICODE属性/BINARY属性なども設定することができます。

TEXT ASCII

TEXT UNICODE

TEXT BINARY

では各属性を設定したテーブルを作成し、テーブル定義を確認してみます。



基本的にVARCHAR型と同じように使うことができます。

**ENUM型**

広告

ここでは文字列型の中でもENUM型について確認していきます。書式は次の通りです。

ENUM('value1','value2',...)

ENUM型では文字列定数をリストします。このデータ型が設定されたカラムに値を格納する場合、このリストの中の文字列定数の一つを指定します。(例外はあります)。またリストできる値は65,535までです。

例として次のように記述します。

mysql> CREATE TABLE test(col ENUM('red','blue','yellow'));



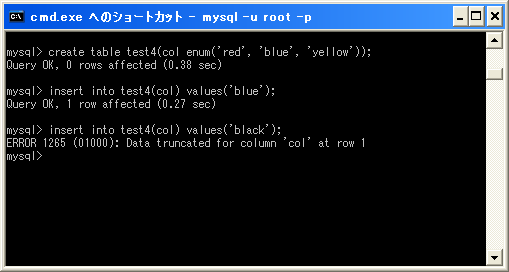
カラムcolにはENUM型が設定してあります。ENUM型には'red','blue','yellow'の3つの値をリストとして記載していますので、このテーブルにデータを追加する場合にはcolカラムには'red','blue','yellow'のいずれか一つを指定して下さい。

mysql> INSERT INTO test(col) VALUES('blue');



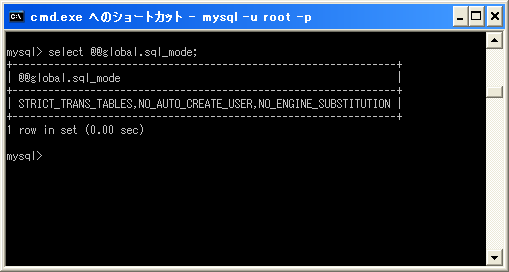
リストに含まれていない値を格納しようとした場合、Strict modeが有効な場合にはエラーとなります。無効の場合には空白が代入されます。

mysql> INSERT INTO test(col) VALUES('black');



実際試してみるとエラーとなりますのでMySQLサーバがStrict modeが有効となっていると考えられます。実際次のようなSQL文を実行して確認できます。

mysql> SELECT @@global.sql\_mode;



「STRICT\_TRANS\_TABLES」が設定されていることが確認できます。これはオプションファイルである「my.ini」に次のような記述があり設定されているためです。

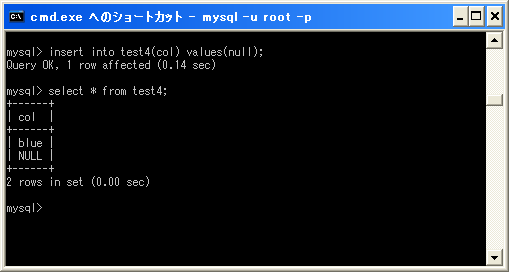
# Set the SQL mode to strict

sql-mode="STRICT\_TRANS\_TABLES,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION"

**NULL**

NULLはリストに含まれていない場合でもデータとして指定することができます。

mysql> INSERT INTO test(col) VALUES(null);



NULLはそのままNULLとして格納されます。

**インデックス**

ENUM型にリストされた文字列定数には順にインデックスが割り当てられます。インデックスは1から開始されます。

ENUM('red','blue','yellow')

'red' インデックス 1

'blue' インデックス 2

'yellow' インデックス 3

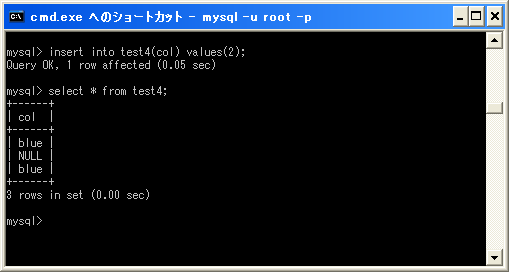
※ NULLがデータとして格納された場合、NULLのインデックスはNULLとなります。

ENUM型が設定されているカラムにデータを追加する場合、文字列の代わりにインデックスを指定することができます。

mysql> CREATE TABLE test(col ENUM('red','blue','yellow'));

mysql> INSERT INTO test(col) VALUES(2);

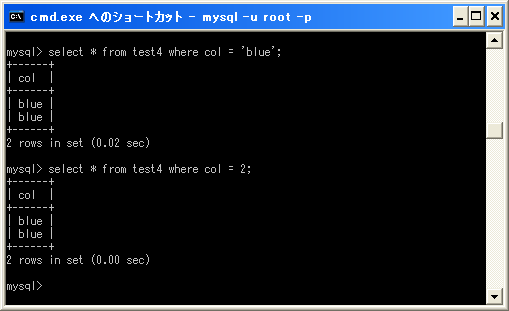
上記の場合、インデックス2に該当する'blue'が格納されます。



またWHERE句の値としてインデックスを指定することもできます。つまり次の2つのSELECT文は同じ結果を取得できます。

SELECT \* FROM test WHERE col = 'blue';

SELECT \* FROM test WHERE col = 2;



**文字セットと照合順序**

ENUM型は文字セットと照合順序を指定することができます。使い方は次の通りです。

ENUM('value1','value2',...) [CHARACTER SET 文字セット] [COLLATE 照合順序]

文字セットと照合順序の設定についてはCHAR型などと同じです。詳しくは「[CHAR型とVARCHAR型](http://www.dbonline.jp/mysql/type/index3.html)」を参照して下さい。

**SET型**

広告

ここでは文字列型の中でもSET型について確認していきます。書式は次の通りです。

SET('value1','value2',...)

SET型では文字列定数をリストします。このデータ型が設定されたカラムに値を格納する場合、このリストの中の0個以上の定数を指定します。(2個以上指定する場合はカンマで区切ります)。またリストできる値は64までです。

ENUM型とSET型は似ていますが、ENUM型は格納可能な値の一覧を定義し一つ選択しているのに対して、SET型は指定可能な値の組み合わせを定義し、データとして格納するのは組み合わせを指定するものです。

例として次のように記述します。

mysql> CREATE TABLE test(col SET('red','blue','yellow'));



カラムcolにはSET型が設定してあります。SET型には'red','blue','yellow'の3つの値をリストとして記載していますので、このテーブルにデータを追加する場合にはcolカラムには'red','blue','yellow'の0個以上の組み合わせたものを指定します。

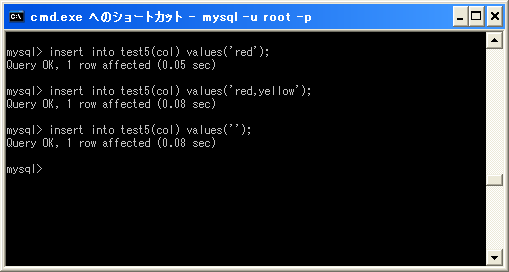
INSERT INTO test(col) VALUES('red');

INSERT INTO test(col) VALUES('blue');

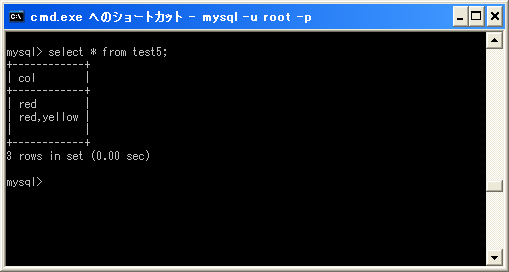
INSERT INTO test(col) VALUES('red,yellow');

INSERT INTO test(col) VALUES('red,blue,yellow');

INSERT INTO test(col) VALUES('');

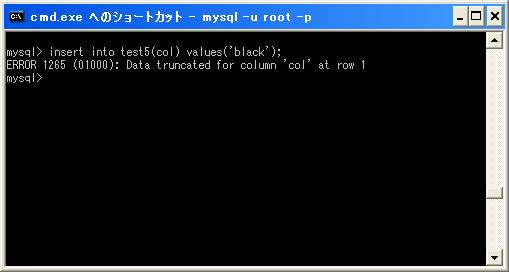


データを格納し、格納されたデータを取得してみます。



リストに含まれていない値を格納しようとした場合、Strict modeが有効な場合にはエラーとなります。無効の場合には空白が代入されます。

mysql> INSERT INTO test(col) VALUES('black');

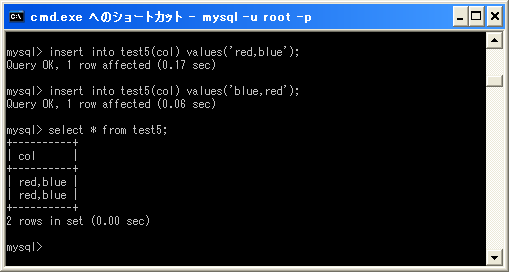


実際試してみるとエラーとなりますのでMySQLサーバがStrict modeが有効となっています。Strict modeについてはENUM型のページで記載していますので、詳しくは「[ENUM型](http://www.dbonline.jp/mysql/type/index8.html)」を参照して下さい。

複数の値を指定したデータは、記述されて順番に関係なく、SET型のリストの中に記述された順序を使ってデータが格納されます。

mysql> INSERT INTO test(col) VALUES('red,blue');

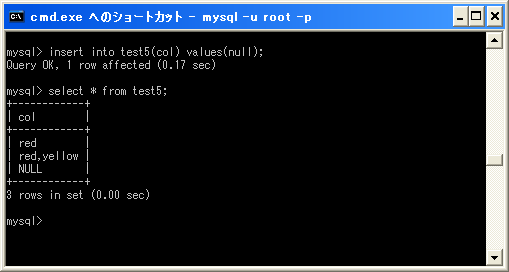
mysql> INSERT INTO test(col) VALUES('blue,red');



**NULL**

NULLはリストに含まれていない場合でもデータとして指定することができます。

mysql> INSERT INTO test(col) VALUES(null);



NULLはそのままNULLとして格納されます。

**ビットを使った指定**

SET型にリストされた文字列定数には順に次のようにビットが割り当てられます。

SET('red','blue','yellow')

'red' 0001 (=1)

'blue' 0010 (=2)

'yellow' 0100 (=4)

SET型が設定されているカラムにデータを追加する場合、文字列の代わりにビットを使って指定することができます。

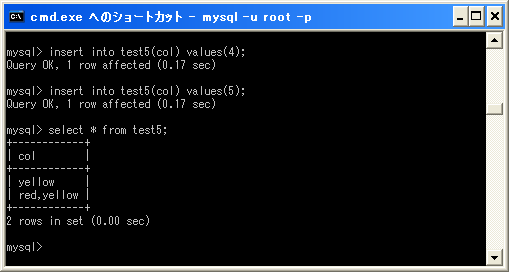
mysql> INSERT INTO test(col) VALUES(4);

上記の場合、ビット4に該当する'yellow'が格納されます。

複数の値を指定する場合は、各値のビットをORした結果を使って指定することができます。

mysql> INSERT INTO test(col) VALUES(5);

上記の場合、5(=0101)ですので該当する'red,yellow'が格納されます。



**文字セットと照合順序**

SET型は文字セットと照合順序を指定することができます。使い方は次の通りです。

SET('value1','value2',...) [CHARACTER SET 文字セット] [COLLATE 照合順序]

文字セットと照合順序の設定についてはCHAR型などと同じです。詳しくは「[CHAR型とVARCHAR型](http://www.dbonline.jp/mysql/type/index3.html)」を参照して下さい。

**MySQLのファイル構造**

広告

MySQLでデータベースを作成したり、データベースにテーブルを作成した時に作成されたデータがどのように格納されるのかについて解説していきます。

* **MySQLのファイル構造**
* [データベースのファイル構造](http://www.dbonline.jp/mysql/storage/index1.html)
* [MyISAM型テーブル用のファイル](http://www.dbonline.jp/mysql/storage/index2.html)
* [InnoDB型テーブル用のファイル](http://www.dbonline.jp/mysql/storage/index3.html)

**データベースのファイル構造**

広告

まずはデータベースを作成した時のファイル構造を確認していきます。

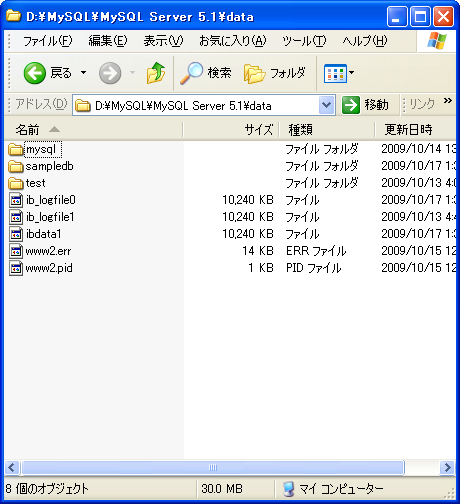
データベースを作成した時にデータが格納される場所がデータディレクトリです。データディレクトリの実際の場所はオプションファイルであるmy.iniファイル内に記述されています。

#Path to the database root

datadir="d:/MySQL/MySQL Server 5.1/data/"

上記のように「datadir=パス」という形で記載されています。(上記はデフォルトで設定された値から変更してあります。変更する場合は他に行わなければならない作業があります。手順については「[データディレクトリの変更](http://www.dbonline.jp/mysqlinstall/install/index6.html)」を参照して下さい)。

ではデータディレクトリで指定された場所について見てみます。



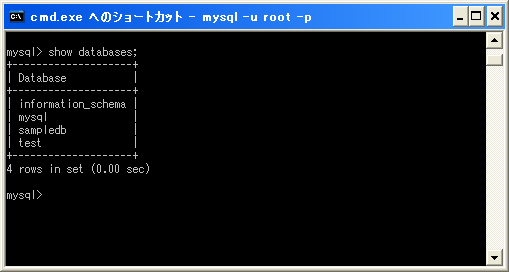
MySQLではデータベース毎にディレクトリが1つ作成されます。現在「mysql」「test」「sampledb」と3つのディレクトリが作成されていますので同じ名前の3つのデータベースが存在していることになります。

「mysql」データベースはMySQLサーバがデータベースに関する様々な情報を管理するために使用しているデータベースです。「test」データベースは書き込みのテスト用などにデフォルトで作成されているものです。「sampledb」は自分で作成したデータベースです。

**データベースの作成**

ではMySQLサーバに接続してデータベースを実際に作成してみます。まず次のように入力して現在のデータベース一覧を確認します。

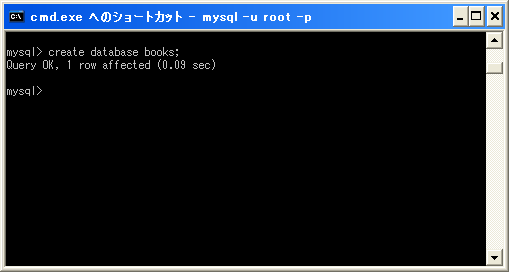
mysql> show databases;



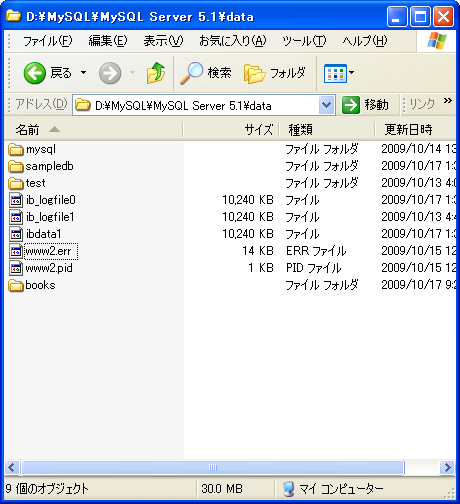
先ほどの3つのデータベースに加えて「information\_schema」というデータベースがあります。このデータベースもMySQLが使用しているものでMySQLサーバーが保持する他のすべてのデータベースに関する情報を保存しています。このデータベースは他のデータベースとは異なりディレクトリを作成しません。

次に新しいデータベースを作成してみます。次のように入力して下さい。

mysql> create database books;

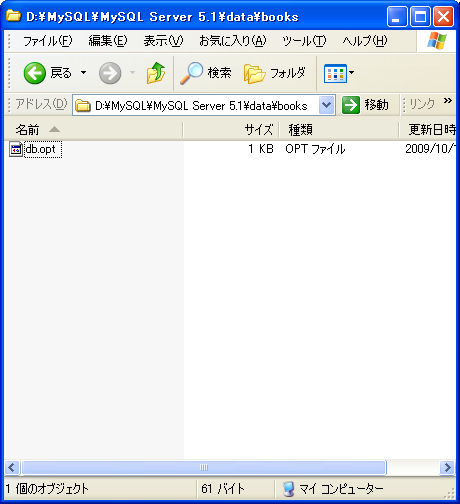


新しいデータベース「books」が作成されました。では改めてデータディレクトリを見てみます。



新しいディレクトリ「books」が作成されていることが確認できます。

作成されたディレクトリの中を確認してみると次のように「db.opt」ファイルが作成されています。



この「db.opt」ファイルには作成したデータベースに関する特徴などが保存されています。このファイルはテキストファイルなのでテキストエディタで中身を確認してみます。

default-character-set=utf8

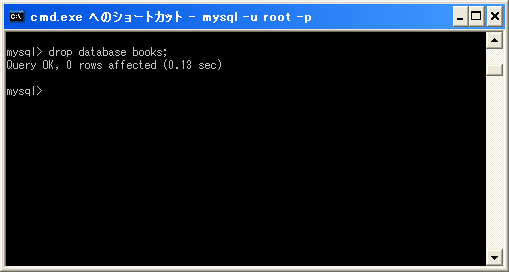
default-collation=utf8\_general\_ci

データベースのデフォルト文字セットと照合順序に関する情報が格納されていました。

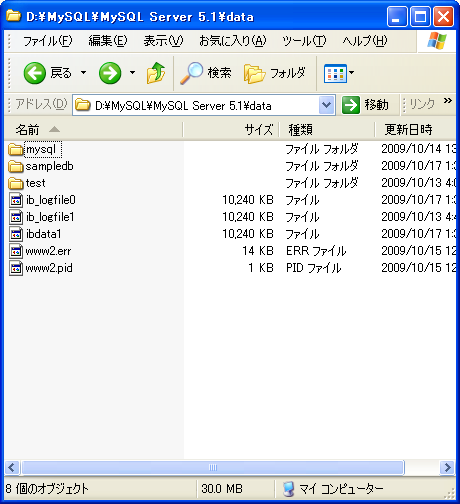
**データベースの削除**

先ほど作成したデータベースを削除してみます。次のように入力して下さい。

mysql> drop database books;



データベース「books」が削除されました。では改めてデータディレクトリを見てみます。



ディレクトリ「books」が削除されていることが確認できます。

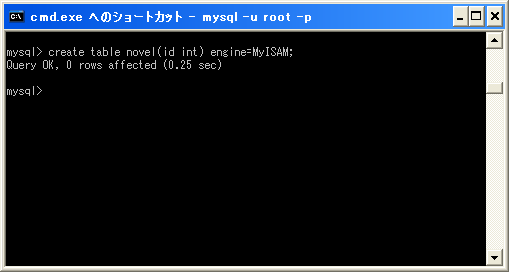
**MyISAM型テーブル用のファイル**

広告

データベースを作成した後で、テーブルを作成します。この時、ストレージエンジンとしてMyISAM型かInnoDB型かでテーブルに関するデータをどのように保管するかが異なります。まずはMyISAM型のテーブルで試してみます。

「books」データベースの中に「novel」テーブルを作成します。

mysql> create table novel(id int) engine=MyISAM;



ではデータディレクトリの中の「books」ディレクトリの中を確認して下さい。



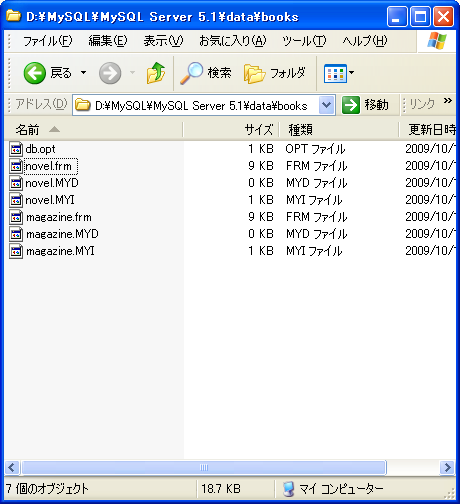
作成したテーブル名の名前を持つ3つのファイルが作成されています。この3つのファイルは次のような目的で使用されます。

table\_name.frm テーブル定義の保存

table\_nam.MYD テーブルデータの保存

table\_nam.MYI インデックスの保存

テーブル毎にこれら3つのファイルが作成されます。ではもう一つ「magazine」テーブルを作成してみます。

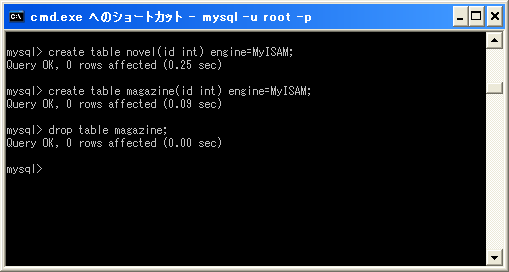


新しいテーブル用のファイルが3つ作成されました。

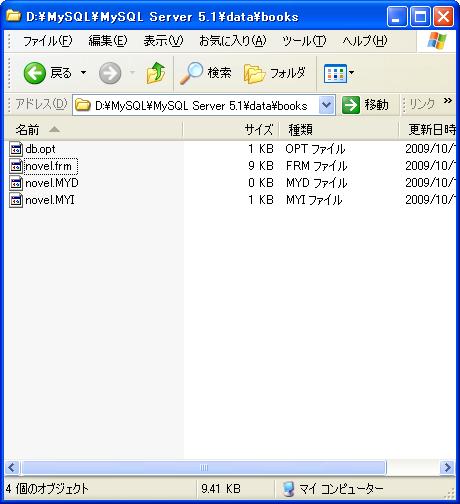
**テーブルの削除**

先ほど作成したテーブルを削除してみます。次のように入力して下さい。

mysql> drop table magazine;



テーブル「magazine」が削除されました。では改めてデータディレクトリの中の「books」ディレクトリの中を見てみます。



「magazine」テーブル用の3つのファイルが削除されていることが確認できます。

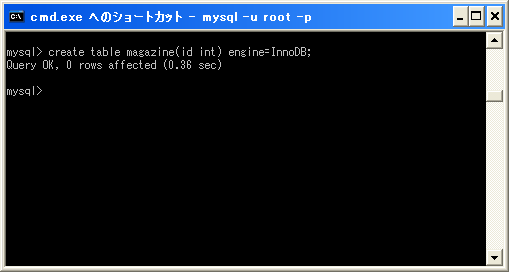
**InnoDB型テーブル用のファイル**

広告

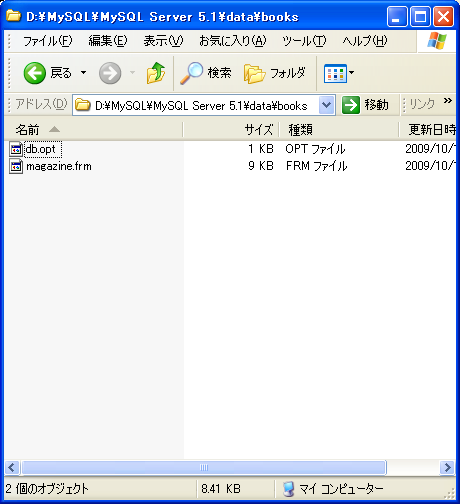
前のページMyISAM型のテーブルについて確認しました。今度はInnoDB型のテーブルで試してみます。

「books」データベースの中に「magazine」テーブルを作成します。

mysql> create table magazine(id int) engine=InnoDB;



ではデータディレクトリの中の「books」ディレクトリの中を確認して下さい。



「テーブル名.frm」というファイルが1つ作成されています。このファイルは次のような目的で使用されます。

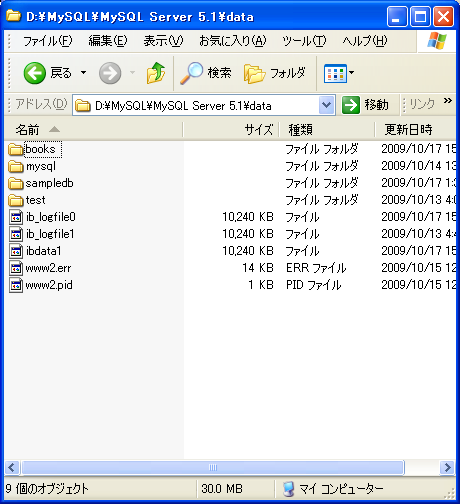
table\_name.frm テーブル定義の保存

テーブルに含まれるデータやインデックスについては全テーブルで共有のファイルに対して書き込まれていきます。デフォルトの設定ではデータベース毎のディレクトリではなくデータディレクトリの中に次のファイルを作成します。

ibdata1 テーブルデータやインデックスの保存

ib\_logfile0 トランザクションログファイル

ib\_logfile1 トランザクションログファイル

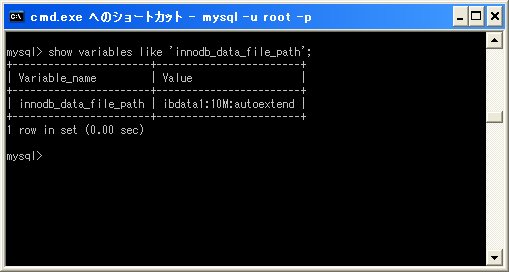


このようにテーブル毎に作成されるテーブル定義ファイル以外はInnoDB型のテーブルのデータは共通のファイルを使って管理されます。

**データファイル**

データを保存するファイルはデフォルトの設定では1つとなっており、10MBのサイズです。多くのデータが格納されて容量が足りなくなると自動で拡張されるように設定されています。デフォルトの設定は次のように入力することで確認できます。

mysql> show variables like 'innodb\_data\_file\_path';



取得した値は次のようになっています。

+-----------------------+------------------------+

| Variable\_name | Value |

+-----------------------+------------------------+

| innodb\_data\_file\_path | ibdata1:10M:autoextend |

+-----------------------+------------------------+

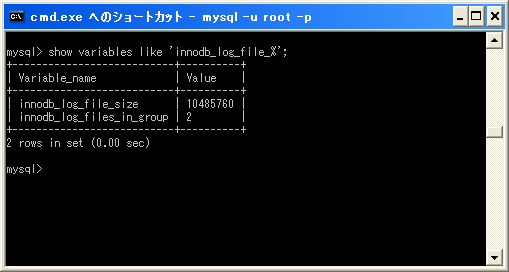
1つのファイルで管理するのではなく複数のファイルを指定してデータを管理することもできます。またファイル毎に拡張可能な最大サイズなども指定することもできます。この値を変更することは可能ですが、単に値を変更すればいいだけではありません。詳しい手順についてはまた別のページで確認します。

**トランザクションログファイル**

トランザクションログファイルは複数のファイルを使ってローテーションしながら記録していきます。

次のように入力することで現在の設定を確認できます。

mysql> show variables like 'innodb\_log\_file\_%';



取得した値は次のようになっています。

+---------------------------+----------+

| Variable\_name | Value |

+---------------------------+----------+

| innodb\_log\_file\_size | 10485760 |

| innodb\_log\_files\_in\_group | 2 |

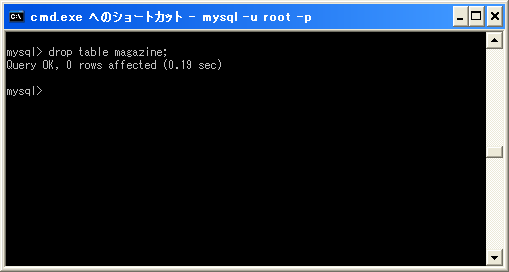
+---------------------------+----------+

デフォルトの設定では1つのファイルサイズが10MB、ファイル数は2つでローテーションを行います。この値は変更することが可能ですが、具体的な手順についてはまた別のページで解説します。

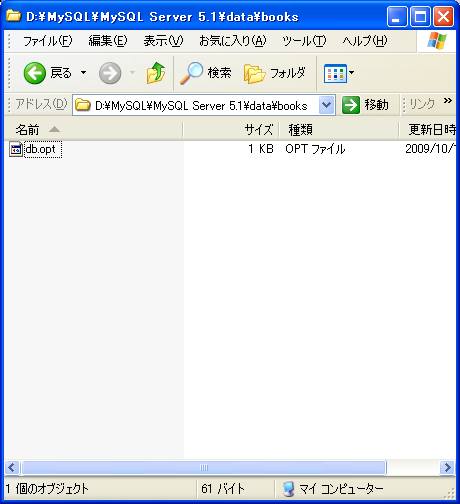
**テーブルの削除**

先ほど作成したテーブルを削除してみます。次のように入力して下さい。

mysql> drop table magazine;



テーブル「magazine」が削除されました。では改めてデータディレクトリの中の「books」ディレクトリの中を見てみます。



「magazine」テーブル用の1つのファイルが削除されていることが確認できます。

ただこの場合でもInnoDBで共有して使っているデータファイルとトランザクションログファイルは削除されません。



**データベースの作成**

広告

MySQLではデータベースを単位として管理されます。これから作成するテーブルや、データにアクセスできるユーザーはデータベースを単位として管理されます。ここではMySQLでデータベースの作成や削除をおこなう方法について確認します。

* **データベースの作成**
* [データベースの作成 (CREATE DATABASE文)](http://www.dbonline.jp/mysql/database/index1.html)
* [データベースに関する情報の取得 (SHOW DATABASES文, SHOW CREATE DATABASE文, INFORMATION\_SCHEMAデータベース)](http://www.dbonline.jp/mysql/database/index2.html)
* [データベースへ接続 (USE文)](http://www.dbonline.jp/mysql/database/index3.html)
* [文字セットと照合順序を指定してデータベースを作成](http://www.dbonline.jp/mysql/database/index4.html)
* [データベースを削除 (DROP DATABASE文)](http://www.dbonline.jp/mysql/database/index5.html)
* [データベース属性の変更 (ALTER DATABASE文)](http://www.dbonline.jp/mysql/database/index6.html)

**データベースの作成**

広告

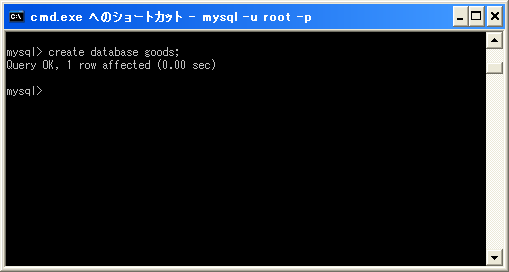
データベースを作成します。CREATE DATABASE文を使います。基本書式は次の通りです。

CREATE DATABASE db\_name;

指定した名前のデータベースを作成します。データベース名は64バイト以内で指定し、Windowsの場合は大文字と小文字は区別されません。

実際に作成してみます。次のように入力して下さい。

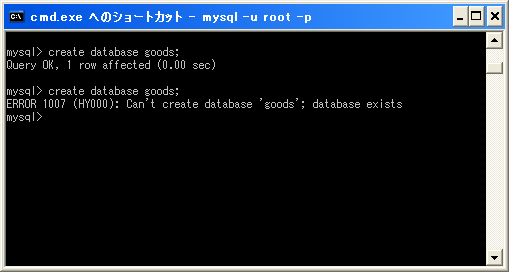
mysql> create database goods;



「goods」データベースが作成されます。

**同じデータベース名を指定した場合**

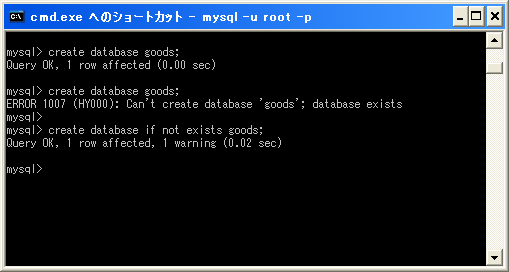
既に存在するデータベース名を指定して新しいデータベースを作成しようとするとエラーが発生します。



既に存在するデータベース名を指定した場合でもエラーが発生しないようにするには「IF NOT EXISTS」を指定します。書式は次の通りです。

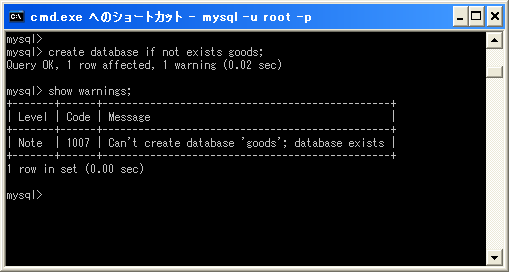
CREATE DATABASE IF NOT EXISTS db\_name;

この書式を使った場合、既に存在するデータベース名を指定してデータベースを作成しようとしてもエラーにはなりません。ただし、エラーにならないというだけで新しいデータベースが作成されるわけではありません。



実際に試してみるとエラーとはなりませんでしたが警告が1つ発生しています。次のように入力することで前回のコマンドでの警告を表示することができます。

mysql> show warnings;



エラーが発生した時と同じメッセージが表示されました。

**データベースに関する情報の取得**

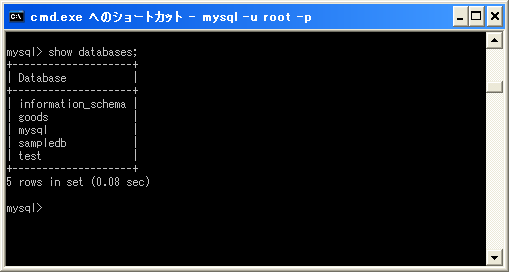
広告

作成したデータベースに関する情報を取得する方法を確認します。

**データベースの一覧**

まずは現在作成されているデータベースの一覧を取得する方法です。SHOW DATABASES文を使います。書式は次の通りです。

SHOW DATABASES;



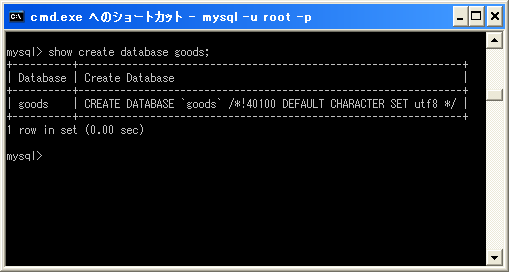
作成済みのデータベースの一覧が表示されます。

なお「information\_schema」「mysql」はMySQL自身が使用しているデータベースです。また「test」はデフォルトで作成されているデータベースでテスト用として使って頂けます。この3つを除いたデータベースがユーザーが作成したデータベースとなります。

**指定した名前のデータベースを作成するためのCREATE DATABASE文を取得**

次にデータベース名を指定し、そのデータベースを作成するためCREATE DATABASE文を表示する方法です。SHOW CREATE DATABASE文を使います。書式は次の通りです。

SHOW CREATE DATABASE db\_name;



照合順序について記載がない場合は、設定されている文字セットのデフォルトの照合順序が使用さえることを表しています。

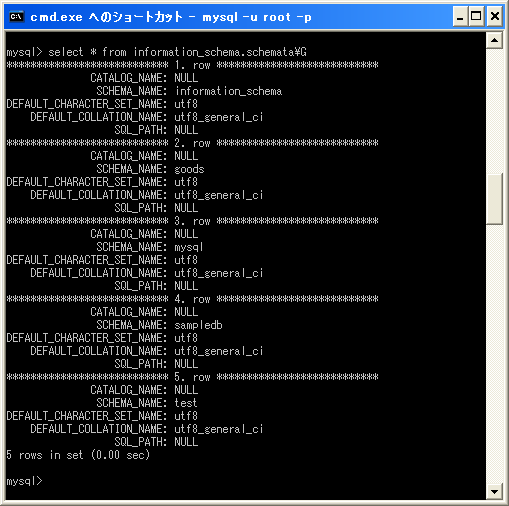
**INFORMATION\_SCHEMAデータベースの参照**

INFORMATION\_SCHEMAデータベースはMySQLに存在するデータベースやテーブルなどの情報を参照するために利用できるものです。データベースに関してはSCHEMATAテーブルに格納されており、情報を参照するには次のように入力して下さい。

SELECT \* FROM INFORMATION\_SCHEMA.SCHEMATA;

SCHEMATAテーブルに含まれるSCHEMA\_NAMEカラムにデータベース名、DEFAULT\_CHARACTER\_SET\_NAMEカラムに文字セット名、DEFAULT\_COLLATION\_NAMEカラムに照合順序が格納されています。

では実際に試してみます。



現在存在しているデータベース毎に1つのデータが存在し、各データベースの文字セットなどが確認できます。

**データベースへ接続**

広告

存在しているデータベースへ接続する方法を確認します。USE文を使います。書式は次の通りです。

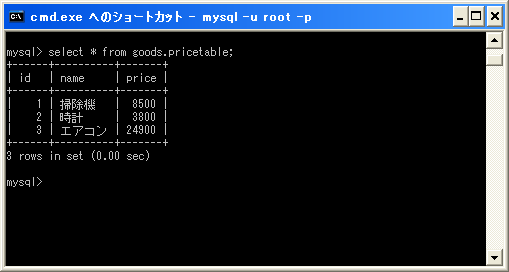
USE db\_name;

例えば「goods」データベースに接続するには次のように入力します。

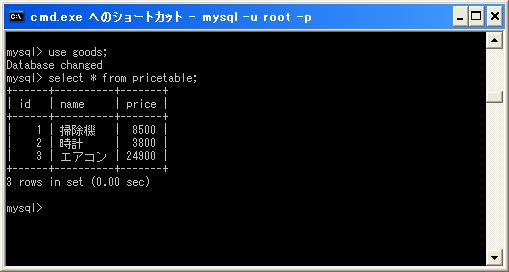
mysql> use goods;

データベース名で指定したデータベースに接続します。ここで接続というのは、接続以降に行われる操作の時にデータベースが省略された場合にデフォルトで使用されるデータベースを設定するということです。

例として「sampledb」データベースの中の「pricetable」テーブルのデータを取得してみます。データベースに接続していない場合には「データベース名.テーブル名」という形式でテーブルを指定する必要があります。



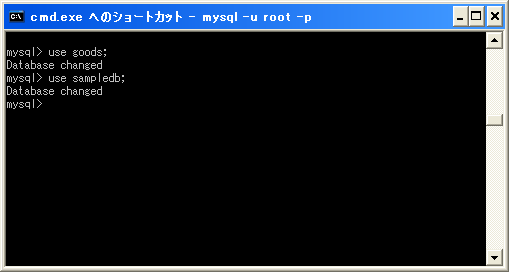
USE文を使ってデータベースへ接続していれば、データベース名を省略しテーブル名の指定だけでデータを取得できます。



なおデータベースに接続している場合でも、「データベース名.テーブル名」の形式で指定すれば他のデータベースのテーブルからでもデータを取得することは可能です。

**他のデータベースへ接続する**

データベースに接続している状態でも、再度USE文を使用すれば他のデータベースへ接続しなおすことができます。



**MySQLコマンドラインツールのコマンドで接続する**

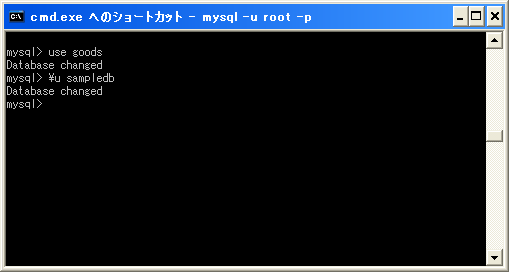
MySQLコマンドラインツールを使ってMySQLサーバへ接続している場合、useコマンドを使用することでデータベースへ接続することができまう。書式は次の通りです。

USE db\_name

¥u db\_name

USE文とUSEコマンドは末尾にセミコロンがあるかどうかの違いだけです。

では実際に試してみます。



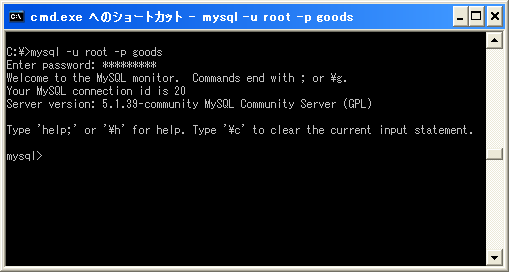
**MySQLコマンドラインツールの引数**

MySQLコマンドラインツールを使用している場合は、MySQLコマンドラインツールを起動する時に、接続したいデータベースも指定することができます。

mysql -u root -p db\_name

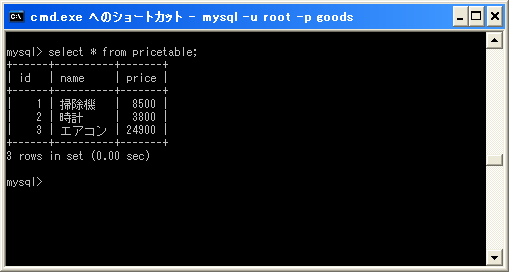
実際に試してみます。

mysql -u root -p goods



MySQLコマンドラインツールを起動しMySQLサーバへ接続します。この時、起動時に指定したデータベースに接続します。

では「goods」データベースに含まれる「pricetable」テーブルのデータを取得してみます。



問題なくデータを取得できていることから、現在「goods」データベースに接続が行われていることが確認できます。

**接続しているデータベースの確認**

現在接続しているデータベースを確認するにはdatabase関数を使用します。次のように入力して下さい。

mysql> select database();



**文字セットと照合順序を指定してデータベースを作成**

広告

文字セットと照合順序を指定してデータベースを作成する方法を確認します。書式は次の通りです。

CREATE DATABASE db\_name

CHARACTER SET charset\_name COLLATE collation\_name;

文字セットを指定しなかった場合には、MySQLサーバに設定されているデフォルト文字セットが適用されます。

**文字セットの指定**

では実際に試してみます。まずは文字セットだけを設定します。(MySQLで設定可能な文字セットについては「[文字セットと照合順序](http://www.dbonline.jp/mysql/ini/index5.html)」を参照して下さい)。

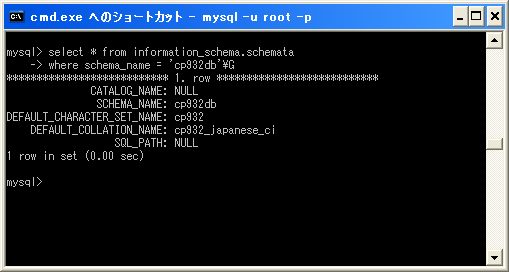
mysql> create database cp932db character set cp932;



INFORMATION\_SCHEMA.SCHEMATAテーブルを参照してみます。

mysql> select \* from information\_schema.schemata

-> where schema\_name = 'cp932db'¥G

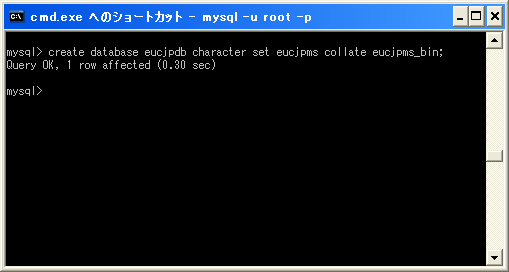


デフォルトの文字セットがcp932に設定されていることが確認できます。なお照合順序を指定しなかった場合は指定した文字セットのデフォルトの照合順序に設定されます。

**文字セットと照合順序の指定**

今度は文字セットに加えて照合順序も設定します。

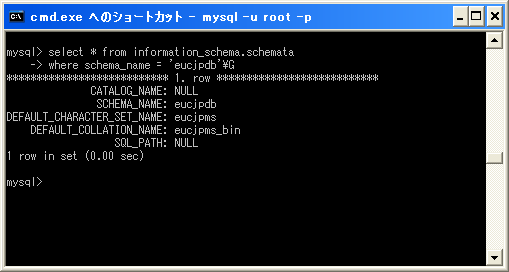
mysql> create database eucjpdb character set eucjpms collate eucjpms\_bin;



INFORMATION\_SCHEMA.SCHEMATAテーブルを参照してみます。

mysql> select \* from information\_schema.schemata

-> where schema\_name = 'eucjpdb'¥G



デフォルトの文字セットがeucjpmsに設定されていることが確認できます。また照合順序もeucjpmsのデフォルト照合順序のeucjpms\_japanese\_ciではなく指定したeucjpms\_binに設定されていることが確認できます。

**データベースを削除**

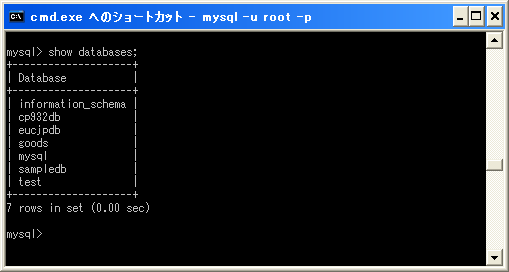
広告

データベースを作成します。DROP DATABASE文を使います。基本書式は次の通りです。

DROP DATABASE db\_name;

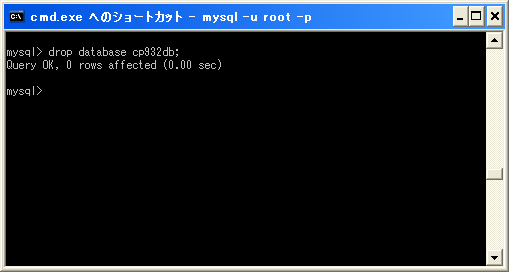
指定した名前のデータベースを削除します。

実際に削除してみます。まずは存在するデータベースの一覧を確認します。

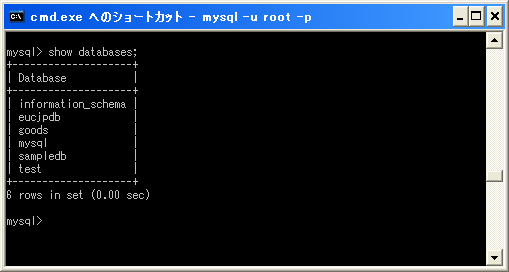


この中から「cp932」データベースを削除します。次のように入力して下さい。

mysql> drop database cp932db;



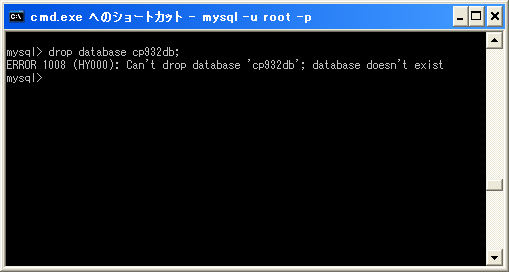
指定したデータベースが削除されました。では再度データベースの一覧を確認します。



一覧からも削除されていることが確認できます。

**存在しないデータベースを指定した場合**

存在しないデータベース名を指定して削除しようとするとエラーとなります。

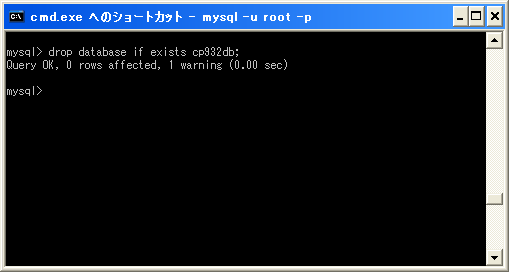


存在しないデータベース名を指定した場合でもエラーが発生しないようにするには「IF EXISTS」を指定します。書式は次の通りです。

DROP DATABASE IF EXISTS db\_name;

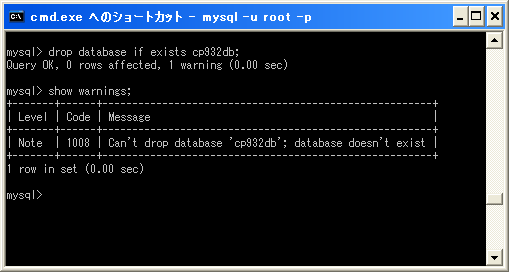
では実際に試してみます。次のように入力して下さい。

mysql> drop database if exists cp932db;



実際に試してみるとエラーとはなりませんでしたが警告が1つ発生しています。次のように入力することで前回のコマンドでの警告を表示することができます。

mysql> show warnings;



エラーが発生した時と同じメッセージが表示されました。

**データベース属性の変更**

広告

データベースの属性を変更する方法を確認します。ALTER DATABASE文を使います。基本書式は次の通りです。

ALTER DATABASE db\_name

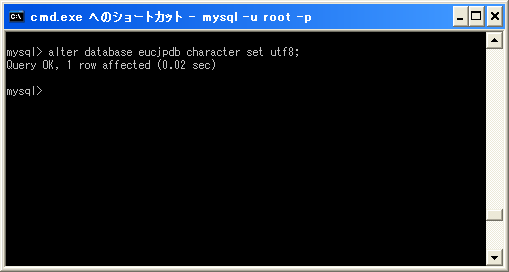
CHARACTER SET charset\_name COLLATE collation\_name;

既に作成済みのデータベース名を指定し、属性の変更を行えます。現在のバージョンで変更可能なのはデフォルト文字セットと照合順序の変更です。

なおデータベースにおけるデフォルト文字セットというのは、データベースにテーブルを作成した時、文字セットを指定しなかった場合に設定される文字セットです。ALTER DATABASEを使ってデフォルト文字セットを変更したとしても、既に存在するテーブルの文字セットなどが変更されるわけではなりません。あくまで変更以降に作成されるテーブルのデフォルトの文字セットに影響があるだけです。

実際に変更してみます。次のように入力して下さい。

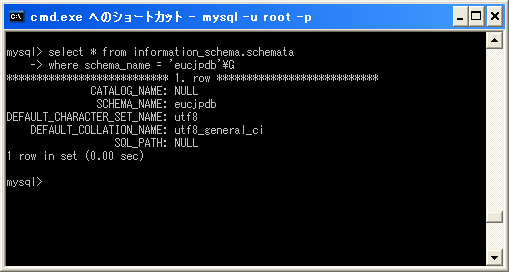
mysql> alter database eucjpdb character set utf8;



「eucjpdb」データベースのデフォルト文字セットが変更されました。ではINFORMATION\_SCHEMA.SCHEMATAテーブルを参照して確認してみます。

mysql> select \* from information\_schema.schemata

-> where schema\_name = 'eucjpdb'¥G



文字セットが指定したutf8に変更されていることが確認できます。

今回は文字セットだけを変更しましたが、同時に照合順序を変更することもできます。今回のように文字セットだけを変更した場合は、文字セットのデフォルトの照合順序が設定されます。

**ユーザーの作成**

広告

MySQLへ接続しデータを追加したり参照したりするにはMySQLのユーザーが必要です。MySQLユーザーはユーザー名、パスワード、ホストの3つの組み合わせから成り立っていて、ユーザー毎にどのようなことが行えるのかの権限を細かく設定することも可能です。

ここではユーザーの作成方法などについて確認して行きます。

* **ユーザーの作成**
* [ユーザーの作成 (CREATE USER文)](http://www.dbonline.jp/mysql/user/index1.html)
* [パスワードの設定 (SET PASSWORD文)](http://www.dbonline.jp/mysql/user/index2.html)
* [ユーザーの変更 (RENAME USER文)](http://www.dbonline.jp/mysql/user/index3.html)
* [ユーザーの削除 (DROP USER文)](http://www.dbonline.jp/mysql/user/index4.html)
* [権限の種類と設定されている権限の確認 (SHOW GRANTS文)](http://www.dbonline.jp/mysql/user/index5.html)
* [権限の設定 (GRANT文)](http://www.dbonline.jp/mysql/user/index6.html)
* [権限設定のオプション](http://www.dbonline.jp/mysql/user/index7.html)
* [権限の削除 (REVOKE文)](http://www.dbonline.jp/mysql/user/index8.html)

**ユーザーの作成**

広告

MySQLコマンドラインツールなどを利用してMySQLサーバに接続し、様々な操作を行うにはユーザーアカウントが必要です。MySQLをインストールした直後には「root」ユーザーしか作成されていません。ここでは新しいユーザーを作成する方法を確認します。

ユーザーを作成するにはCREATE USER文を使います。基本書式は次の通りです。

CREATE USER user;

CREATE USER user IDENTIFIED BY [PASSWORD] 'password';

1つ目の構文は指定したユーザーをパスワード無しで作成します。パスワードは後からでも設定できますが、パスワード無しのアカウントはセキュリティ的に問題がありますので2つ目の構文を使ってユーザーの作成時にパスワードを設定するようにして下さい。

2つ目の構文は指定したユーザーを作成しパスワードを設定します。ユーザーの指定方法は後で説明します。パスワードは文字列なのでシングルクオーテーションで囲って指定します。

複数のユーザーをまとめて追加する場合は、次の書式を使用して下さい。

CREATE USER user1 IDENTIFIED BY [PASSWORD] 'password1',

user2 IDENTIFIED BY [PASSWORD] 'password2',

user3 IDENTIFIED BY [PASSWORD] 'password3', ...;

**ユーザーの書式**

ユーザーは次の書式で指定します。

user\_name@host\_name

ユーザーはユーザー名とMySQLへ接続するホスト名の組み合わせで指定します。登録されたユーザー名を持つユーザーが存在しても、そのユーザー名で接続できるのは同時に登録されているホストからしか行えません。また同じユーザー名であっても使用しているホスト毎に異なる権限を設定することもできます。

ユーザー名は最大16文字です。

ユーザー名とホスト名は特別な文字(例えば@など)を含まない限りは引用符で囲う必要はありませんが、特別な文字を含む場合やホスト名にワイルドカードを使用する場合はシングルクオーテーションで囲って指定して下さい。(引用符で囲む場合はユーザー名とホスト名は別々に囲って下さい)。

'user\_name'@'host\_name'

ホスト名はホスト名の他にIPアドレスやlocalhostを指定することができます。

'username'@'host.example.com'

'username'@'192.168.128.1'

'username'@'localhost'

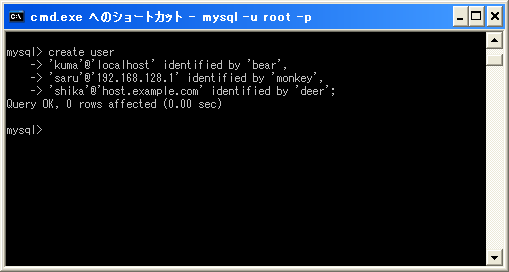
では実際にユーザーを作成してみます。

mysql> create user

-> 'kuma'@'localhost' identified by 'bear',

-> 'saru'@'192.168.128.1' identified by 'monkey',

-> 'shika'@'host.example.com' identified by 'deer';

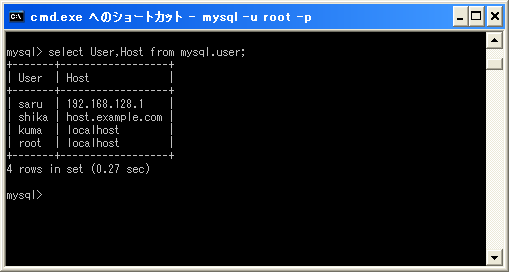


3つのユーザーを作成しました。

それではユーザーの一覧を取得してみます。ユーザーの情報は「mysql」データベースの中の「user」テーブルに格納されています。このテーブルにはユーザー毎に1つのデータが作成されており、ユーザー名やホスト名だけではなくパスワードや権限などの情報が格納されています。

今回は「User」カラムと「ホスト」カラムの値を取得します。

mysql> select User,Host from mysql.user;



もともと存在していた「root」ユーザーに加えて追加した3つのユーザーが存在していることが確認できます。

**ホスト名のワイルドカード指定**

ホスト名にはワイルドカードである'\_'と'%'が使用できます。'\_'は任意の一文字を表し'%'は任意の長さの文字列に一致します。

ユーザー名ではワイルドカードを使用できません。

例えば同じドメインに所属する任意のホストからアクセス可能なユーザーの指定は次のように記述できます。

'username'@'%.example.com'

同じCクラス(192.168.128.0/24)の全てのホストにからアクセス可能なユーザーの指定は次のように記述できます。

'username'@'192.168.128.%'

またホスト名を省略した場合は次のように記述したものとして扱われます。

'username'@'%'

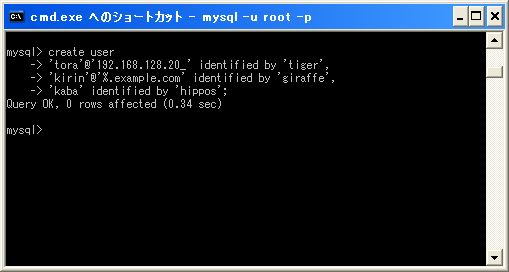
では実際にユーザーを作成してみます。

mysql> create user

-> 'tora'@'192.168.128.20\_' identified by 'tiger',

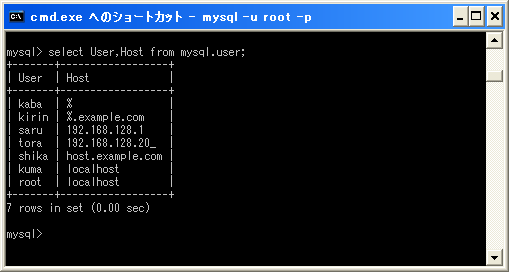
-> 'kirin'@'%.example.com' identified by 'giraffe',

-> 'kaba' identified by 'hippos';



改めてユーザー情報を取得してみます。

mysql> select User,Host from mysql.user;

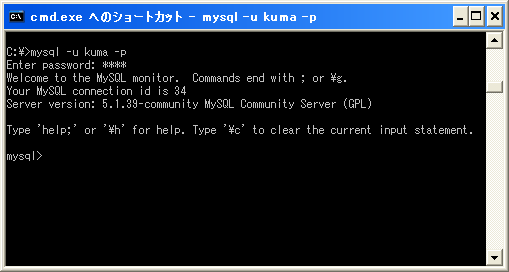


ワイルドカードはそのまま格納され、ホスト名を省略した場合は'ユーザー名@%'の形式で格納されていることが確認できます。

**作成したユーザーでMySQLサーバへ接続**

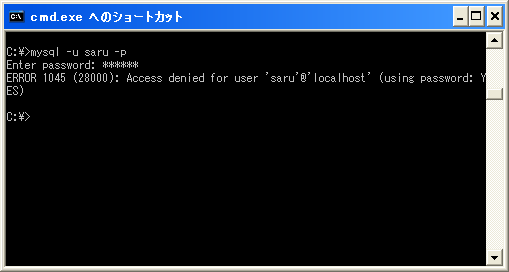
それではMySQLコマンドラインツールを使い、新しく作成したユーザーでMySQLサーバへ接続してみます。

ローカルホストからアクセスが許可されている「kuma」ユーザーを使って接続します。



問題なく接続できました。

では今度はホスト「192.168.128.1」からアクセスが許可されている「saru」ユーザーを使って接続してみます。



ローカルホストから接続しようとしましたのでユーザーは「'saru'@'localhost'」と認識されます。ユーザーとして追加されているのは「'saru'@'192.168.128.1'」ですのでアクセスが拒否されました。

**パスワードの設定**

広告

作成済みのユーザーのパスワードを再設定するにはSET PASSWORD文を使います。基本書式は次の通りです。

SET PASSWORD = PASSWORD('some password')

SET PASSWORD FOR user = PASSWORD('some password')

1つ目の構文は現在接続しているユーザーのパスワードを設定します。2つ目の構文は指定したユーザーのパスワードを設定します。

パスワードは指定した文字列をPASSWORD関数を使って暗号化した値を格納します。(PASSWORD関数については「[PASSWORD関数](http://www.dbonline.jp/mysql/function/index2.html)」を参照して下さい)。

実際には次のように入力します。

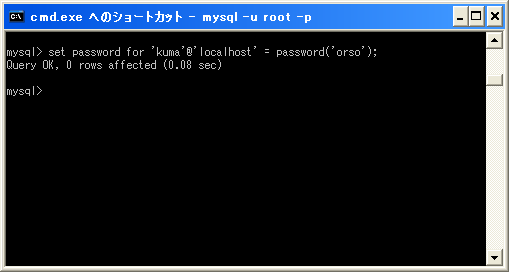
set password for 'username'@'192.168.128.1' = password('pass');

上記では「'username'@'192.168.128.1'」のパスワードを'pass'に設定しています。ユーザーはユーザー名ではなくユーザー名@ホスト名の形式でユーザーを指定する点に注意して下さい。

**サンプル**

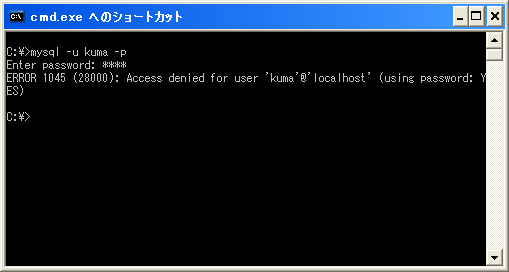
では実際にパスワードを変更してみます。

mysql> set password for 'kuma'@'localhost' = password('orso');

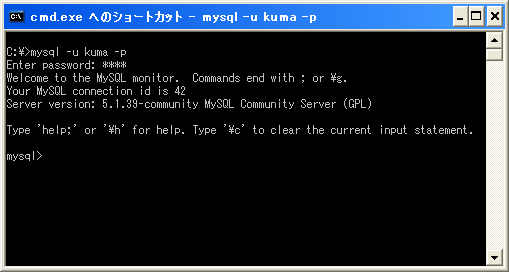


パスワードの変更が完了しました。

それでは実際に試してみます。まず変更前のパスワードを指定してみます。



接続に失敗しました。今度は新しく設定したパスワードを指定してみます。



新しく設定したパスワードを指定すると無事接続に成功しました。

**ユーザーの変更**

広告

作成済みのユーザーのパスワードを再設定するにはRENAME USER文を使います。基本書式は次の通りです。

RENAME USER old\_user TO new\_user;

old\_userで指定したユーザーをnew\_userに変更します。

実際には次のように入力します。

rename user 'olduser'@'localhost' to 'newuser'@'loclahost';

変更前のユーザーが存在しなかったり、変更後に指定したユーザーが既に存在していた場合はエラーとなります。またユーザー名などは変わりますが変更前のユーザーが持っていた権限はそのまま新しいユーザーに引き継がれます。

なお複数のユーザーをまとめて変更する場合は次の書式を使用して下さい。

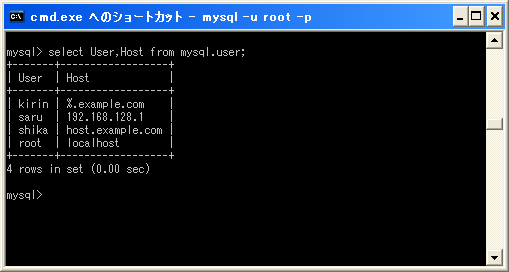
RENAME USER old\_user1 TO new\_user1,

old\_user2 TO new\_user2,

old\_user3 TO new\_user3, ...;

**サンプル**

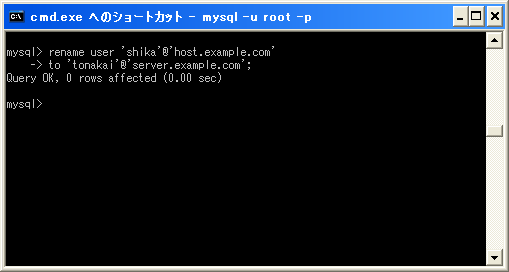
では実際にユーザーを変更してみます。現在存在しているユーザーの一覧は次の通りです。



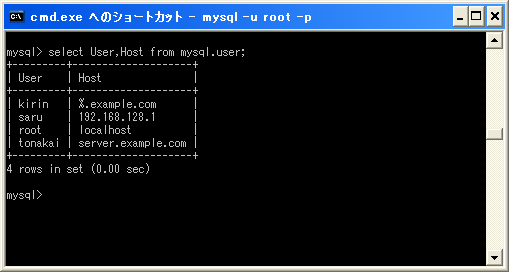
'shika'@'host.example.com'を'tonakai'@'server.example.com'に変更します。

mysql> rename user 'shika'@'host.example.com'

-> to 'tonakai'@'server.example.com';



再度ユーザー一覧を確認してみます。



ユーザーが変更されていることが確認できました。

**ユーザーの削除**

広告

作成済みのユーザーを削除するにはDROP USER文を使います。基本書式は次の通りです。

DROP USER user;

userで指定したユーザーを削除します。

実際には次のように入力します。

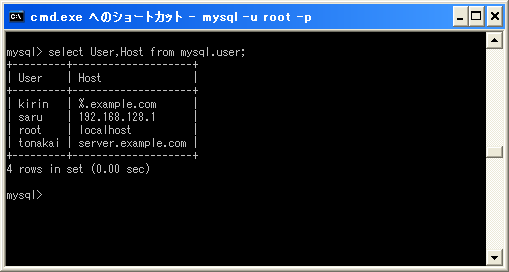
drop user 'user'@'localhost';

なお複数のユーザーをまとめて変更する場合は次の書式を使用して下さい。

DROP USER user1, user2, user3, ...;

**サンプル**

では実際にユーザーを変更してみます。現在存在しているユーザーの一覧は次の通りです。

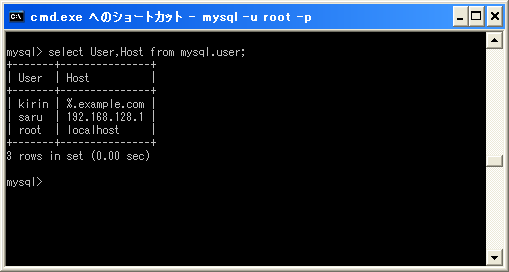


'tonakai'@'server.example.com'を削除します。

mysql> drop user 'tonakai'@'server.example.com';



再度ユーザー一覧を確認してみます。



ユーザーが削除されていることが確認できました。

**権限の種類と設定されている権限の確認**

広告

作成したユーザーがMySQLに接続し、データベースやテーブルを作成したり、テーブルからデータを取得するにはその操作に対する権限が設定されている必要があります。ここではまずMySQLにおける権限の種類を確認し、そしてユーザーにどの権限が設定されているかどうか確認する方法を見ていきます。

**権限のレベル**

権限は大きく分けて次の5つのレベルが用意されています。

グローバル レベル(G)

グローバルレベル権限は全てのデータベースに適用される権限です。この権限は

mysql.user テーブルに格納されます。

データベース レベル(D)

データベースレベル権限は特定のデータベース内の全てのオブジェクトに適用さ

れる権限です。この権限は mysql.db と mysql.host テーブル内に格納されます。

テーブル レベル(T)

テーブルレベル権限は特定のテーブル内の全てのカラムに適用される権限です。

この権限は mysql.tables\_priv テーブル内に格納されています。

カラム レベル(C)

カラムレベル権限は特定テーブル内の単一カラムに適用される権限です。この権

限は mysql.columns\_priv テーブル内に格納されています。

ルーチン レベル

CREATE ROUTINE、ALTER ROUTINE、EXECUTE、そして GRANT 権限はストアド ルー

チンに適用される権限です。(ファンクションとプロシージャ)それらは、グロー

バルとデータベース レベルで供与されます。また、CREATE ROUTINE 以外は、こ

れらの権限は各ルーチンに対してルーチン レベルで適用する事ができ、

mysql.procs\_priv テーブル内に格納されます。

グローバルレベルで権限は全てのデータベースに対する権限となります。それに対してデータベースレベルはデータベースを指定して権限を与える場合、テーブルレベルは特定のデータベースのさらに特定のテーブルに対する権限を与える場合、カラムラベルは特定のテーブルの中にある特定のカラムだけに対する権限を与える場合です。

例えば特定のデータベースにだけテーブルを作成する権限を与えたり、特定のテーブルのさらに特定のカラムの値だけを参照する権限を与えることなどができます。

**権限の種類**

次に与えることができる権限の種類です。

|  |  |  |
| --- | --- | --- |
| **権限** | **レベル** | **意味** |
| ALL [PRIVILEGES] | GDT | GRANT OPTION 以外の全てのシンプルな権限を設定します。 |
| ALTER | GDT | ALTER TABLE の使用を可能にします。 |
| ALTER ROUTINE | GD | ストアド ルーチンの変更、ドロップを可能にします。 |
| CREATE | GDT | CREATE TABLE の使用を可能にします。 |
| CREATE ROUTINE | GD | ストアド ルーチンの作成を可能にします。 |
| CREATE TEMPORARY TABLES | GD | CREATE TEMPORARY TABLE の使用を可能にします。 |
| CREATE USER | G | CREATE USER、DROP USER、RENAME USER、そして REVOKE ALL PRIVILEGES の使用を可能にします。 |
| CREATE VIEW | GDT | CREATE VIEW の使用を可能にします。 |
| DELETE | GDT | DELETE の使用を可能にします。 |
| DROP | GDT | DROP TABLE の使用を可能にします。 |
| EVENT | GD | イベントスケジューラがイベントを作成するのを可能にします。 |
| EXECUTE | G | ユーザがストアドルーチンを起動させるのを可能にします。 |
| FILE | G | SELECT ... INTO OUTFILE と LOAD DATA INFILE の使用を可能にします。 |
| INDEX | GDT | CREATE INDEX と DROP INDEX の使用を可能にします。 |
| INSERT | GDTC | INSERTの使用を可能にします。 |
| LOCK TABLES | GD | SELECT 権限を持つテーブル上の LOCK TABLES の使用を可能にします。 |
| PROCESS | G | SHOW FULL PROCESSLIST の使用を可能にします。 |
| REFERENCES | GDTC | インプリメントされていません。 |
| RELOAD | G | FLUSH の使用を可能にします。 |
| REPLICATION CLIENT | G | ユーザがスレーブとマスタの場所を問い合わせる事を可能にします。 |
| REPLICATION SLAVE | G | 複製スレーブが必要とします。 (マスタからバイナリ ログ イベントを読み込む為) |
| SELECT | GDTC | SELECT の使用を可能にします。 |
| SHOW DATABASES | G | SHOW DATABASES は全てのデータベースを表示します。 |
| SHOW VIEW | GD | SHOW CREATE VIEWの使用を可能にします。 |
| SHUTDOWN | G | mysqladmin shutdown の使用を可能にします。 |
| SUPER | G | CHANGE MASTER、KILL、PURGE MASTER LOGS、そして SET GLOBAL ステートメントの使用を可能にし、 the mysqladmin debug コマンドはmax\_connections が達成していても接続を (一回) 許可します。 |
| TRIGGER | GDT | ユーザがトリガを作成、ドロップするのを可能にします。 |
| UPDATE | GDTC | UPDATE の使用を可能にします。 |
| USAGE | G | 「権限が無い」 の同義語です。 |
| GRANT OPTION | T | 権限を与えるのを可能にします。 |

権限の種類によって与える対象のレベルが決まっています。(G：グローバル、D：データベース、T：テーブル、C：カラムです)。

例えば「SELECT」権限ではグローバルレベル単位からカラム単位まで任意のレベルで設定することができますが、「CREATE USER」権限はグローバルレベルでしか与えることができません。つまり個別のデータベースにのみ有効なユーザーなどは作成できないということです。それに対してデータの取得は全データベースに対する権限を与えることもできますし、カラムを指定して与えることもできます。

**設定されている権限の確認**

ではユーザー毎にどのような権限が設定されているのかを確認してみます。権限を確認するにはSHOW GRANTS文を使います。書式は次の通りです。

SHOW GRANTS FOR user

この構文を実行すると、指定したユーザーに設定されている権限と同じ権限を設定するための必要なGRANT文を表示します。

実際には次のように入力します。

show grants for 'user'@'localhost';

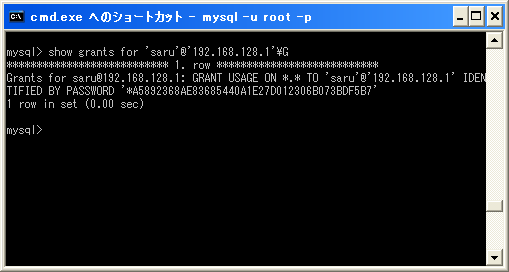
また現在接続しているユーザーの権限について確認するには次のように入力することもできます。

show grants;

show grants for current\_user();

では実際に試してみます。

mysql> show grants for 'saru'@'192.168.128.1'¥G



表示されたGRANT文は次の通りです。

Grants for saru@192.168.128.1:

GRANT USAGE ON \*.\* TO 'saru'@'192.168.128.1' IDENTIFIED BY PASSWORD '\*\*\*\*'

GRANT文の詳細は次のページで確認しますが、上記は全てのデータベースに対して権限が無い状態に設定されています。よって現在では権限が必要な操作を何もできない状態です。CREATE USER文によって作成されただけのユーザーはこのように何も権限が設定されていません。

**権限の設定**

広告

作成したユーザーに対して権限を設定するにはGRANT文を使用します。書式は次の通りです。

GRANT priv\_type [(column\_list)] [, priv\_type [(column\_list)]] ...

ON [object\_type] {tbl\_name | \* | \*.\* | db\_name.\*}

TO user [IDENTIFIED BY [PASSWORD] 'password']

[, user [IDENTIFIED BY [PASSWORD] 'password']] ...

[REQUIRE

NONE |

[{SSL| X509}]

[CIPHER 'cipher' [AND]]

[ISSUER 'issuer' [AND]]

[SUBJECT 'subject']]

[WITH with\_option [with\_option] ...]

object\_type =

TABLE

| FUNCTION

| PROCEDURE

with\_option =

GRANT OPTION

| MAX\_QUERIES\_PER\_HOUR count

| MAX\_UPDATES\_PER\_HOUR count

| MAX\_CONNECTIONS\_PER\_HOUR count

| MAX\_USER\_CONNECTIONS count

複雑な文なのですが、基本書式は次のようになります。

GRANT 権限 ON レベル TO user;

※権限のレベル及び種類については「[権限の種類と設定されている権限の確認](http://www.dbonline.jp/mysql/user/index5.html)」を参照して下さい。

各レベルに応じて権限を設定するには、それぞれ次のように記述します。

グローバルレベル:

GRANT 権限 ON \*.\* TO user;

データベースレベル:

GRANT 権限 ON db\_name.\* TO user;

テーブルレベル:

GRANT 権限 ON db\_name.table\_name TO user;

カラムレベル:

GRANT 権限 (カラム1, カラム2, ...) ON db\_name.table\_name TO user;

例えばグローバルレベルでCREATE権限を設定するには次のように入力します。

grant create on \*.\* to 'user'@'localhost';

「test」デーベースの「goods」テーブルに対してSELECT権限とUPDATE権限を設定するには次のように入力します。

grant select,update on test.goods to 'user'@'localhost';

このようにどのレベルに対して何の権限を与えるのかを指定することができます。

**ALL権限**

ALL権限はGRANT OPTION権限以外の全ての権限を与える場合に使用します。

GRANT ALL ON レベル TO user;

実際にどのような権限が与えられるのかは合わせて指定したレベルによって異なります。例えばグローバルレベルを指定した場合には、グローバルレベルで指定可能な権限の全てが設定されますし、テーブルレベルを指定した場合には、テーブルレベルで指定可能な全ての権限が設定されることになります。

**USAGE権限**

USAGE権限はグローバルレベルでのみ指定が可能な権限で、この権限を設定すると何も権限が無いことになります。

GRANT USAGE ON \*.\* TO user;

CREATE USER文でユーザーを作成した直後はこの状態となります。

**サンプル**

では実際に試してみます。まず対象となるユーザーの現在の権限を確認します。

mysql> show grants for 'saru'@'192.168.128.1'¥G



現時点での権限は次のように表示されます。

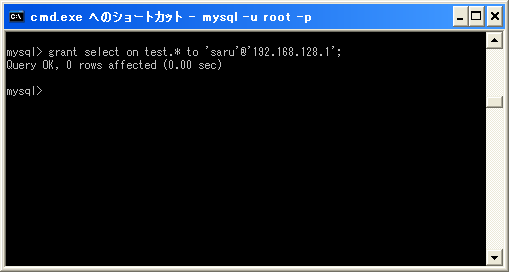
GRANT USAGE ON \*.\* TO 'saru'@'192.168.128.1' IDENTIFIED BY PASSWORD '\*\*\*\*'

グローバルレベルに対するUSAGE権限が設定されているだけとなっています。つまり現時点では権限は何も設定されていません。

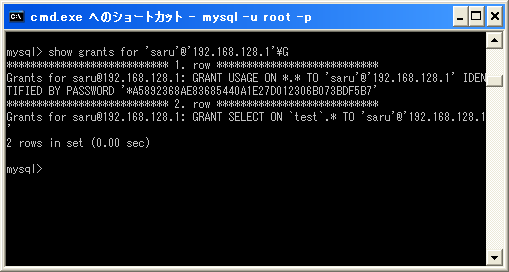
---

ではまずtestデータベース対するSELECT権限を設定します。

mysql> grant select on test.\* to 'saru'@'192.168.128.1';



再度権限を確認します。



現時点での権限は次のように表示されます。

GRANT USAGE ON \*.\* TO 'saru'@'192.168.128.1' IDENTIFIED BY PASSWORD '\*\*\*\*'

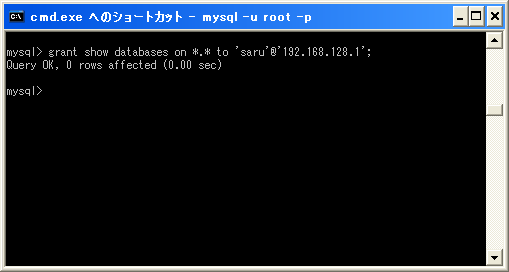
GRANT SELECT ON `test`.\* TO 'saru'@'192.168.128.1'

このユーザーの権限と同じ権限を与えるには、まずグローバルレベルに対するUSAGE権限を与えた上で、データベースレベルでSELECT権限を与えることによって可能になることを表しています。

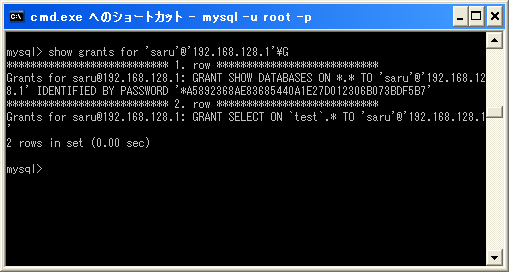
---

今度はグローバルレベルに対してSHOW DATABASES権限を設定します。

mysql> grant show databases on \*.\* to 'saru'@'192.168.128.1';



再度権限を確認します。



現時点での権限は次のように表示されます。

GRANT SHOW DATABASES ON \*.\* TO 'saru'@'192.168.128.1' IDENTIFIED BY PASSWORD '\*\*\*\*'

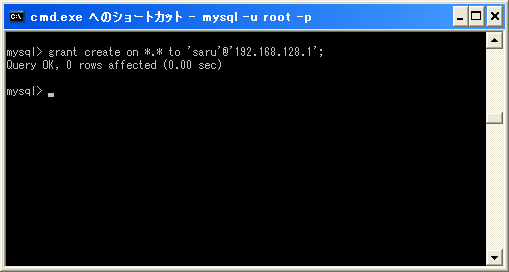
GRANT SELECT ON `test`.\* TO 'saru'@'192.168.128.1'

このユーザーの権限と同じ権限を与えるには、まずグローバルレベルに対するSHOW DATABASES権限を与えた上で、データベースレベルでSELECT権限を与えることによって可能になることを表しています。

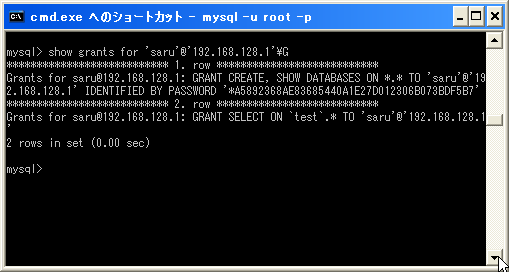
---

最後にグローバルレベルに対してCREATE権限を設定します。

mysql> grant create on \*.\* to 'saru'@'192.168.128.1';



再度権限を確認します。



現時点での権限は次のように表示されます。

GRANT CREATE, SHOW DATABASES ON \*.\* TO 'saru'@'192.168.128.1' IDENTIFIED BY PASSWORD '\*\*\*\*'

GRANT SELECT ON `test`.\* TO 'saru'@'192.168.128.1'

このユーザーの権限と同じ権限を与えるには、まずグローバルレベルに対するCREATE権限とSHOW DATABASES権限を与えた上で、データベースレベルでSELECT権限を与えることによって可能になることを表しています。

このように権限を設定していくと、USAGE権限から他の権限に変わる場合を除いて今までの権限に追加されていくことが確認できます。なお一度設定した権限に対して改めてUSAGE権限を設定しても権限はなくなりません。権限を削除するにはREVOKE文を使用します。

**権限設定のオプション**

広告

GRANT文を使って権限を設定する場合のオプションについて確認します。

**GRANT OPTION**

1つ目はGRANT OPTIONです。このオプションを設定すると他のユーザーに対する権限を設定することができるようになります。書式は次の通りです。

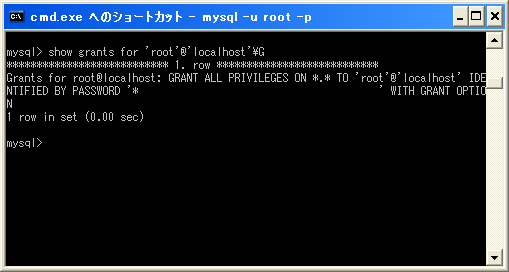
GRANT 権限 ON レベル TO user WITH GRANT OPTION;

GRANT OPTIONはレベル毎に設定することが可能です。そして他のユーザーに権限を与えることができる権限は、自分が与えられている権限だけです。

例えばデータベースレベルでSELECT権限が設定されているユーザーに、データベースレベルでWITH GRANT OPTIONを設定すると、他のユーザーに対してデータベースレベルでのSELECT権限を設定することが可能になります。

例として管理者である'root'@'localhost'ユーザーに設定されている権限を見てみます。

mysql> show grants for 'root'@'localhost'¥G



現時点での権限は次のように表示されます。

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'localhost'

IDENTIFIED BY PASSWORD '\*\*\*\*' WITH GRANT OPTION

'root'@'localhost'ユーザーはグローバルレベルに対して全ての権利が与えられており、そしてGRANT OPTIONが付いていますので全ての権限に関して他のユーザーに設定することが可能です。

**MAX\_QUERIES\_PER\_HOUR, MAX\_UPDATES\_PER\_HOUR, MAX\_CONNECTIONS\_PER\_HOUR**

2つ目はMAX\_QUERIES\_PER\_HOUR count、MAX\_UPDATES\_PER\_HOUR count、そして MAX\_CONNECTIONS\_PER\_HOURです。これらのオプションは1時間ごとに行う処理数の制限を行います。書式は次の通りです。

GRANT 権限 ON レベル TO user WITH MAX\_QUERIES\_PER\_HOUR count;

GRANT 権限 ON レベル TO user WITH MAX\_UPDATES\_PER\_HOUR count;

GRANT 権限 ON レベル TO user WITH MAX\_CONNECTIONS\_PER\_HOUR count;

MAX\_QUERIES\_PER\_HOURは1時間の間にユーザが行う事ができるクエリ数を設定します。MAX\_UPDATES\_PER\_HOURは1時間の間にユーザが行う事ができる更新数を設定します。MAX\_CONNECTIONS\_PER\_HOURは1時間の間にユーザがログインできる回数を設定します。

**MAX\_USER\_CONNECTIONS**

3つ目はMAX\_USER\_CONNECTIONSです。このオプションは同時接続数の最大数を設定します。書式は次の通りです。

GRANT 権限 ON レベル TO user WITH MAX\_USER\_CONNECTIONS count;

この値に0を設定した場合は制限がなくなります。

**権限の削除**

広告

ユーザーに対して設定した権限を削除するにはREVOKE文を使います。書式は次の通りです。

REVOKE priv\_type [(column\_list)] [, priv\_type [(column\_list)]] ...

ON [object\_type] {tbl\_name | \* | \*.\* | db\_name.\*}

FROM user [, user] ...

複雑な文なのですが、基本書式は次のようになります。

REVOKE 権限 ON レベル FROM user;

権限やレベルの指定方法は権限を設定する時に使用するGRANT文と同じです。(詳しくは「[権限の設定](http://www.dbonline.jp/mysql/user/index6.html)」を参照して下さい)。

例えばグローバルレベルのSELECT権限を削除するには次のように入力します。

revoke select on \*.\* from 'user'@'localhost';;

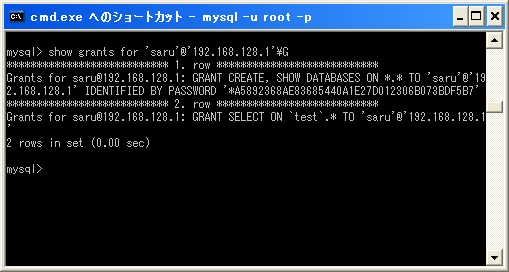
なお特定のユーザーから全てのレベルの全ての権限を削除するには次の書式を利用できます。

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...

権限を全て削除してもユーザーそのものは削除されません。ユーザーを削除するにはDROP USER文を使用して下さい。

**サンプル**

では実際に試してみます。まず対象となるユーザーの現在の権限を確認します。



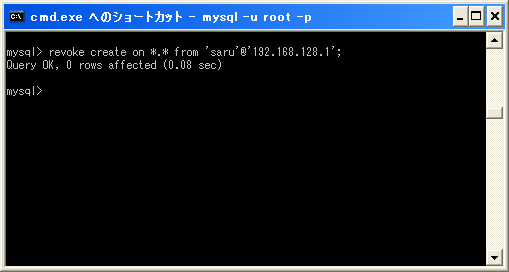
現時点での権限は次のように表示されます。

GRANT CREATE, SHOW DATABASES ON \*.\* TO 'saru'@'192.168.128.1' IDENTIFIED BY PASSWORD '\*\*'

GRANT SELECT ON `test`.\* TO 'saru'@'192.168.128.1'

ではグローバルレベルのCREATE権限を削除します。次のように入力します。

mysql> revoke create on \*.\* from 'saru'@'192.168.128.1';



では改めて権限を確認してみます。



現時点での権限は次のように表示されます。

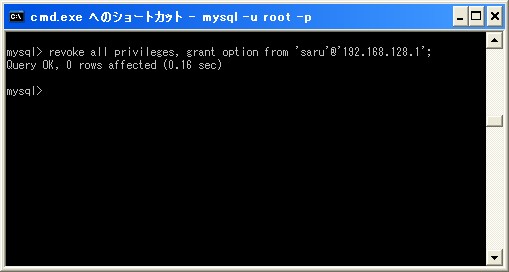
GRANT SHOW DATABASES ON \*.\* TO 'saru'@'192.168.128.1' IDENTIFIED BY PASSWORD '\*\*'

GRANT SELECT ON `test`.\* TO 'saru'@'192.168.128.1'

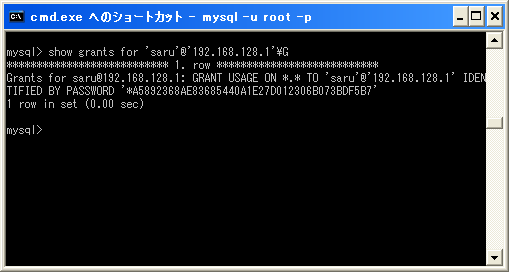
グローバルレベルのCREATE権限が削除されていることが確認できます。

次に全ての権限を削除します。次のように入力します。

mysql> revoke all privileges, grant option from 'saru'@'192.168.128.1';



では改めて権限を確認してみます。



現時点での権限は次のように表示されます。

GRANT USAGE ON \*.\* TO 'saru'@'192.168.128.1' IDENTIFIED BY PASSWORD '\*\*'

グローバルレベルに対してUSAGEが設定されており全ての権限が無くなっていることが確認できます。

**テーブルの作成**

広告

MySQLにてテーブルを作成したり削除したりする方法を解説します。作成されるテーブルに含まれるカラムに対して、デフォルトの値を設定したりNULLの値を格納できるかどうかなどの条件を設定することができます。

* **テーブルの作成**
* [テーブルの作成 (CREATE TABLE文)](http://www.dbonline.jp/mysql/table/index1.html)
* [テーブルに関する情報の取得 (SHOW TABLES文, SHOW CREATE TABLE, SHOW TABLE STATUS文)](http://www.dbonline.jp/mysql/table/index2.html)
* [カラムに関する情報の取得 (SHOW COLUMNS文)](http://www.dbonline.jp/mysql/table/index3.html)
* [テーブルの削除 (DROP TABLE文)](http://www.dbonline.jp/mysql/table/index4.html)
* [NULLの許可 (NOT NULL)](http://www.dbonline.jp/mysql/table/index5.html)
* [デフォルト値の設定 (DEFAULT)](http://www.dbonline.jp/mysql/table/index6.html)
* [AUTO\_INCREMENTの設定](http://www.dbonline.jp/mysql/table/index7.html)
* [プライマリーキー制約 (PRIMARY KEY)](http://www.dbonline.jp/mysql/table/index8.html)
* [ユニークキー制約 (UNIQUE)](http://www.dbonline.jp/mysql/table/index9.html)
* [インデックスの作成 (INDEX)](http://www.dbonline.jp/mysql/table/index10.html)
* [外部キー制約 (FOREIGN KEY)](http://www.dbonline.jp/mysql/table/index11.html)
* [親テーブルの更新/削除の時の挙動 (ON DELETE句, ON UPDATE句)](http://www.dbonline.jp/mysql/table/index12.html)
* [ストレージエンジンの設定 (ENGINE)](http://www.dbonline.jp/mysql/table/index13.html)
* [文字セットと照合順序の設定 (CHARACTER SET, COLLATE)](http://www.dbonline.jp/mysql/table/index14.html)
* [別のテーブルから新しいテーブルを作成](http://www.dbonline.jp/mysql/table/index15.html)
* [別のテーブルと同じ定義のテーブルを作成](http://www.dbonline.jp/mysql/table/index16.html)
* [テーブル属性の変更 (ALTER TABLE文)](http://www.dbonline.jp/mysql/table/index17.html)
* [テーブル名の変更 (ALTER TABLE RENAME文)](http://www.dbonline.jp/mysql/table/index18.html)
* [カラムの名前と定義の変更 (ALTER TABLE CHANGE, ALTER TABLE MODIFY文)](http://www.dbonline.jp/mysql/table/index19.html)
* [カラムの追加と削除 (ALTER TABLE ADD COLUMN, ALTER TABLE DROP COLUMN文)](http://www.dbonline.jp/mysql/table/index20.html)