MySQL Tutorial

Tutorialspoint.com

MySQL is the most popular Open Source Relational SQL database management system.

MySQL is one of the best RDBMS being used for developing web based software applications. This tutorial gives an initial push to start you with MySQL. For more detail kindly check tutorialspoint.com/mysql

What is Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching, and replicating the data it holds.

Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

So now a days we use relational database management systems (RDBMS) to store and manager huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.

A **Relational DataBase Management System (RDBMS)** is a software that:

- Enables you to implement a database with tables, columns, and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology:

Before we proceed to explain MySQL database system, lets revise few definitions related to database.

- **Database:** A database is a collection of tables, with related data.
- **Table:** A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column:** One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row:** A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- **Redundancy:** Storing data twice, redundantly to make the system faster.
- **Primary Key:** A primary key is unique. A key value can not occur twice in one table. With a key you can find at most one row.
- **Foreign Key:** A foreign key is the linking pin between two tables.
- **Compound Key:** A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index:** An index in a database resembles an index at the back of a book.
- **Referential Integrity:** Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Database:

MySQL is a fast, easy-to-use RDBMS used being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons.

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

Before You Begin:

Before you begin this tutorial you should have a basic knowledge of the information covered in our PHP and HTML tutorials.

This tutorial focuses heavily on using MySQL in a PHP environment. Many examples given in this tutorial will be useful for PHP Programmers.

We recommend you check our PHP Tutorial for your reference.

Downloading MySQL:

All downloads for MySQL are located at MySQL Downloads. Pick the version number for *MySQL Community Server* you want and, as exactly as possible, the platform you want.

Installing MySQL on Linux/Unix

The recommended way to install MySQL on a Linux system is via RPM. MySQL AB makes the following RPMs available for download on its web site:

- **MySQL -** The MySQL database server, which manages databases and tables, controls user access, and processes SQL queries.
- **MySQL-client -** MySQL client programs, which makes it possible to connect to, and interact with, the server.
- **MySQL-devel -** Libraries and header files that come in handy when compiling other programs that use MySQL.
- **MySQL-shared -** Shared libraries for the MySQL client
- **MySQL-bench -** Benchmark and performance testing tools for the MySQL database server.

The MySQL RPMs listed here are all built on a SuSE Linux system, but they.ll usually work on other Linux variants with no difficulty.

Now follow the following steps to proceed for installation:

1. Login to the system using **root** user.
2. Switch to the directory containing the RPMs:
3. Install the MySQL database server by executing the following command. Remember to replace the filename in italics with the file name of your RPM.

```
[root@host]# rpm -i MySQL-5.0.9-0.i386.rpm
```

Above command takes care of installing MySQL server, creating a user of MySQL, creating necessary configuration and starting MySQL server automatically.

You can find all the MySQL related binaries in /usr/bin and /usr/sbin. All the tables and databases will be created in /var/lib/mysql directory.

4. This is optional but recommended step to install the remaining RPMs in the same manner:

```
[root@host]# rpm -i MySQL-client-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-devel-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-shared-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-bench-5.0.9-0.i386.rpm
```

Installing MySQL on Windows:

Default installation on any version of Windows is now much easier than it used to be, as MySQL now comes neatly packaged with an installer. Simply download the installer package, unzip it anywhere, and run setup.exe.

Default installer setup.exe will walk you through the trivial process and by default will install everything under C:\mysql.

Test the server by firing it up from the command prompt the first time. Go to the location of the mysqld server, which is probably C:\mysql\bin, and type:

```
mysqld.exe --console
```

NOTE: If you are on NT then you will have to use mysqld-nt.exe instead of mysqld.exe

If all went well, you will see some messages about startup and InnoDB. If not, you may have a permissions issue. Make sure that the directory that holds your data is accessible to whatever user (probably mysql) the database processes run under.

MySQL will not add itself to the start menu, and there is no particularly nice GUI way to stop the server either. Therefore, if you tend to start the server by double clicking the mysqld executable, you should remember to halt the process by hand by using mysqladmin, Task List, Task Manager, or other Windows-specific means.

Verifying MySQL Installation:

After MySQL has been successfully installed, the base tables have been initialized, and the server has been started, you can verify that all is working as it should via some simple tests.

Use the mysqladmin Utility to Obtain Server Status:

Use **mysqladmin** binary to check server version. This binary would be available in /usr/bin on linux and in C:\mysql\bin on windows.

```
[root@host]# mysqladmin --version
```

It will produce following result on Linux. It may vary depending on your installation:

```
mysqladmin  Ver 8.23 Distrib 5.0.9-0, for redhat-linux-gnu on i386
```

If you do not get such message then there may be some problem in your installation and you would need some help to fix it.

Execute simple SQL commands using MySQL Client:

You can connect to your MySQL server by using MySQL client using **mysql** command. At this momemnt you do not need to give any password as by default it will be set to blank.

So just use following command

```
[root@host]# mysql
```

It should be rewarded with a mysql> prompt. Now you are connected to the MySQL server and you can execute all the SQL command at mysql> prompt as follows.

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.13 sec)
```

Post-installation Steps:

MySQL ships with a blank password for the root MySQL user. As soon as you have successfully installed the database and client you need to set a root password as follows:

```
[root@host]# mysqladmin -u root password "new_password";
```

Now to make a connection to your MySQL server you would have to use following command:

```
[root@host]# mysql -u root -p
Enter password:*******
```

Unix users will also want to put your MySQL directory in your PATH, so you won't have to keep typing out the full path every time you want to use the command-line client. For bash, it would be something like:

```
export PATH=$PATH:/usr/bin:/usr/sbin
```

Running MySQL at boot time:

If you want to run MySQL server at boot time then make sure you have following entry in /etc/rc.local file

```
/etc/init.d/mysqld start
```

Also,you should have mysqld binary in /etc/init.d/ directory.

Running and Shutting down MySQL Server:

First check if your MySQL server is running or not. You can use following command to check this:

```
ps -ef | grep mysqld
```

If your MySql is running then you will see **mysqld** process listed out in your result. If server is not running then you can start it by using following command:

```
root@hoat# cd /usr/bin
./safe_mysqld &
```

Now if you want to shutdown an already running MySQL server then you can do it by using following command:

```
root@hoat# cd /usr/bin
./mysqladmin -u root -p shutdown
Enter password: ******
```

Setting Up a MySQL User Accounts:

For adding a new user to MySQL you just need to add a new entry to **user** table in database **mysql**.

Below is an example of adding new user **guest** with SELECT, INSERT and UPDATE privileges with the password **guest123** the SQL query is :

```
root@host# mysql -u root -p
Enter password:*******
mysql> use mysql;
Database changed

mysql> INSERT INTO user
          (host, user, password,
           select_priv, insert_priv, update_priv)
           VALUES ('localhost', 'guest',
           PASSWORD('guest123'), 'Y', 'Y', 'Y');
Query OK, 1 row affected (0.20 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT host, user, password FROM user WHERE user = 'guest';
+-----------+---------+-------------------+
| host      | user    | password          |
+-----------+---------+-------------------+
| localhost | guest | 6f8c114b58f2ce9e |
+-----------+---------+-------------------+
1 row in set (0.00 sec)
```

When adding a new user remember to encrypt the new password using PASSWORD() function provided by MySQL. As you can see in the above example the password mypass is encrypted to 6f8c114b58f2ce9e.

Notice the FLUSH PRIVILEGES statement. This tells the server to reload the grant tables. If you don't use it then you won't be able to connect to mysql using the new user account at least until the server is rebooted.

You can also specify other privileges to a new user by setting the values of following columns in user table to 'Y' when executing the INSERT query or you can update them later using UPDATE query.

- Select_priv
- Insert_priv
- Update_priv
- Delete_priv
- Create_priv
- Drop_priv
- Reload_priv
- Shutdown_priv
- Process_priv
- File_priv
- Grant_priv
- References_priv
- Index_priv
- Alter_priv

Another way of adding user account is by using GRANT SQL command; Following example will ada user **zara** with password **zara123** for a particular database called **TUTORIALS**.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use mysql;
Database changed

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    -> ON TUTORIALS.*
    -> TO 'zara'@'localhost'
    -> IDENTIFIED BY 'zara123';
```

This will also create an entry in mysql database table called **user**.

NOTE: MySQL does not terminate a command until you give a semi colon (;) at the end of SQL command.

The /etc/my.cnf File Configuration:

Most of the cases you should not touch this file. By default it will have following entries:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

[mysql.server]
user=mysql
basedir=/var/lib

[safe_mysqld]
err-log=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Here you can specify a different directory for error log, otherwise you should not change any entry in this table.

Administrative MySQL Command:

Here is the list of important MySQL command which you will use time to time to work with MySQL database:

- **USE *Databasename*** : This will be used to select a particular database in MySQL workarea.
- **SHOW DATABASES:** Lists the databases that are accessible by the MySQL DBMS.
- **SHOW TABLES:** Shows the tables in the database once a database has been selected with the use command.
- **SHOW COLUMNS FROM *tablename:*** Shows the attributes, types of attributes, key information, whether NULL is permitted, defaults, and other information for a table.
- **SHOW INDEX FROM *tablename*:** Presents the details of all indexes on the table, including the PRIMARY KEY.
- **SHOW TABLE STATUS LIKE *tablename*\G:** Reports details of the MySQL DBMS performance and statistics.

MySQL PHP Syntax

MySQL works very well in combination of various programming languages like PERL, C, C++, JAVA and PHP. Out of these languages, PHP is the most popular one because of its web application development capabilities.

This tutorial focuses heavily on using MySQL in a PHP environment. If you are interested in MySQL with PERL then you can look into PERL and MySQL Tutorial.

PHP provides various functions to access MySQL database and to manipulate data records inside MySQL database. You would require to call PHP functions in the same way you call any other PHP function.

The PHP functions for use with MySQL have the following general format:

```
mysql_function(value,value,...);
```

The second part of the function name is specific to the function, usually a word that describes what the function does. The following are two of the functions which we will use in our tutorial

```
mysqli_connect($connect);
mysqli_query($connect,"SQL statement");
```

Following example shows a generic sysntax of PHP to call any MySQL function.

```
<html>
<head>
<title>PHP with MySQL</title>
</head>
<body>
<?php
   $retval = mysql_function(value, [value,...]);
   if( !$retval )
   {
       die ( "Error: a related error message" );
```

```
   }
   // Otherwise MySQL  or PHP Statements
?>
</body>
</html>
```

Starting from next chapter we will see all the important MySQL functionality along with PHP.

MySQL Database Connection

You can establish MySQL database using **mysql** binary at command prompt.

Example:

Here is a simple example to connect to MySQL server from command prompt:

```
[root@host]# mysql -u root -p
Enter password:******
```

This will give you mysql> command prompt where you will be able to execute any SQL command. Following is the result of above command:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2854760 to server version: 5.0.9

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

In above example we have used **root** as a user but you can use any other user. Any user will be able to perform all the SQL operation which are allowed to that user.

You can disconnect from MySQL database any time using **exit** command at mysql> prompt.

```
mysql> exit
Bye
```

MySQL Connection using PHP Script:

PHP provides **mysql_connect()** function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success, or FALSE on failure.

Syntax:

```
connection mysql_connect(server,user,passwd,new_link,client_flag);
```

Parameter	Description
server	Optional - The host name running database server. If not specified then default value is **localhost:3036**.

user	Optional - The username accessing the database. If not specified then default is the name of the user that owns the server process.
passwd	Optional - The password of the user accessing the database. If not specified then default is an empty password.
new_link	Optional - If a second call is made to mysql_connect() with the same arguments, no new connection will be established; instead, the identifier of the already opened connection will be returned.
client_flags	Optional - A combination of the following constants: • MYSQL_CLIENT_SSL - Use SSL encryption • MYSQL_CLIENT_COMPRESS - Use compression protocol • MYSQL_CLIENT_IGNORE_SPACE - Allow space after function names • MYSQL_CLIENT_INTERACTIVE - Allow interactive timeout seconds of inactivity before closing the connection

You can disconnect from MySQL database anytime using another PHP function **mysql_close()**. This function takes a single parameter which is a connection returned by **mysql_connect()** function.

Syntax:

```
bool mysql_close ( resource $link_identifier );
```

If a resource is not specified then last opened database is closed. This function returns true if it closes connection successfully otherwise it returns false.

Example:

Try out following example to connect to a MySQL server:

```
<html>
<head>
<title>Connecting MySQL Server</title>
</head>
<body>
<?php
   $dbhost = 'localhost:3036';
   $dbuser = 'guest';
   $dbpass = 'guest123';
   $conn = mysql_connect($dbhost, $dbuser, $dbpass);
   if(! $conn )
   {
     die('Could not connect: ' . mysql_error());
   }
   echo 'Connected successfully';
   mysql_close($conn);
?>
</body>
</html>
```

Create MySQL Database

You would need special privilege to create or to delete a MySQL database. So assuming you have access to root user, you can create any database using mysql **mysqladmin** binary.

Example:

Here is a simple example to create database called **TUTORIALS**:

```
[root@host]# mysqladmin -u root -p create TUTORIALS
Enter password:******
```

This will create a MySQL database TUTORIALS.

Create Database using PHP Script:

PHP uses **mysql_query** function to create or delete a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax:

```
bool mysql_query( sql, connection );
```

Parameter	Description
sql	Required - SQL query to create or delete a MySQL database
connection	Optional - if not specified then last opened connection by mysql_connect will be used.

Example:

Try out following example to create a database:

```
<html>
<head>
<title>Creating MySQL Database</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = 'CREATE DATABASE TUTORIALS';
$retval = mysql_query( $sql, $conn );
```

```
if(! $retval )
{
  die('Could not create database: ' . mysql_error());
}
echo "Database TUTORIALS created successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

Drop MySQL Database

You would need special privilege to create or to delete a MySQL database. So assuming you have access to root user, you can create any database using mysql **mysqladmin** binary.

Be careful while deleting any database because it will lose your all the data available in your database.

Here is an example to delete a database created in previous chapter:

```
[root@host]# mysqladmin -u root -p drop TUTORIALS
Enter password:******
```

This will give you a warning and it will confirm if you really want to delete this database or not.

```
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the 'TUTORIALS' database [y/N] y
Database "TUTORIALS" dropped
```

Drop Database using PHP Script:

PHP uses **mysql_query** function to create or delete a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax:

```
bool mysql_query( sql, connection );
```

Parameter	Description
sql	Required - SQL query to create or delete a MySQL database
connection	Optional - if not specified then last opened connection by mysql_connect will be used.

Example:

Try out following example to delete a database:

```
<html>
<head>
<title>Deleting MySQL Database</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = 'DROP DATABASE TUTORIALS';
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not delete database: ' . mysql_error());
}
echo "Database TUTORIALS deleted successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

WARNING: While deleting a database using PHP script, it does not prompt you for any confirmation. So be careful while deleting a MySQL database.

Selecting MySQL Database

Once you get connection with MySQL server,it is required to select a particular database to work with. This is because there may be more than one database available with MySQL Server.

Selecting MySQL Database from Command Prompt:

This is very simple to select a particular database from mysql> prompt. You can use SQL command **use** to select a particular database.

Example:

Here is an example to select database called **TUTORIALS**:

```
[root@host]# mysql -u root -p
Enter password:******
mysql> use TUTORIALS;
Database changed
mysql>
```

Now you have selected TUTORIALS database and all the subsequent operations will be performed on TUTORIALS database.

NOTE: all the database name, table names, table fields name are case sensitive. So you would have to use prpoer names while giving any SQL command.

Selecting MySQL Database Using PHP Script:

PHP provides function **mysql_select_db** to select a database.It returns TRUE on success or FALSE on failure.

Syntax:

```
bool mysql_select_db( db_name, connection );
```

Parameter	Description
db_name	Required - MySQL Database name to be selected
connection	Optional - if not specified then last opened connection by mysql_connect will be used.

Example:

Here is the example showing you how to select a database.

```
<html>
<head>
<title>Selecting MySQL Database</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'guest';
$dbpass = 'guest123';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_select_db( 'TUTORIALS' );
mysql_close($conn);
?>
</body>
</html>
```

MySQL Data Types

Properly defining the fields in a table is important to the overall optimization of your database. You should use only the type and size of field you really need to use; don't define a field as 10 characters wide if you know you're only going to use 2 characters. These types of fields (or columns) are also referred to as data types, after the **type of data** you will be storing in those fields.

MySQL uses many different data types, broken into three categories: numeric, date and time, and string types.

Numeric Data Types:

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you. The following list shows the common numeric data types and their descriptions.

- **INT** - A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- **TINYINT** - A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT** - A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT** - A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** - A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 11 digits.
- **FLOAT(M,D)** - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D)** - A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
- **DECIMAL(M,D)** - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types:

The MySQL date and time datatypes are:

- **DATE** - A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** - A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** - A timestamp between midnight, January 1, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
- **TIME** - Stores the time in HH:MM:SS format.
- **YEAR(M)** - Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

String Types:

Although numeric and date types are fun, most data you'll store will be in string format. This list describes the common string datatypes in MySQL.

- **CHAR(M)** - A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** - A variable-length string between 1 and 255 characters in length; for example VARCHAR(25). You must define a length when creating a VARCHAR field.

- **BLOB or TEXT** - A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.
- **TINYBLOB or TINYTEXT** - A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT** - A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** - A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

Create MySQL Tables

The table creation command requires:

- Name of the table
- Names of fields
- Definitions for each field

Syntax:

Here is generic SQL syntax to create a MySQL table:

```
CREATE TABLE table_name (column_name column_type);
```

Now we will create following table in **TUTORIALS** database.

```
tutorials_tbl(
   tutorial_id INT NOT NULL AUTO_INCREMENT,
   tutorial_title VARCHAR(100) NOT NULL,
   tutorial_author VARCHAR(40) NOT NULL,
   submission_date DATE,
   PRIMARY KEY ( tutorial_id )
);
```

Here few items need explanation:

- Field Attribute **NOT NULL** is being used because we do not want this field to be NULL. SO if user will try to create a record with NULL value then MySQL will raise an error.
- Field Attribute **AUTO_INCREMENT** tells to MySQL to go ahead and add the next available number to the id field.
- Keyword **PRIMARY KEY** is used to define a column as primary key. You can use multiple columns separated by comma to define a primary key.

Creating Tables from Command Prompt:

This is easy to create a MySQL table from mysql> prompt. You will use SQL command **CREATE TABLE** to create a table.

Example:

Here is an example which creates **tutorials_tbl**:

```
root@host# mysql -u root -p
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> CREATE TABLE tutorials_tbl(
   -> tutorial_id INT NOT NULL AUTO_INCREMENT,
   -> tutorial_title VARCHAR(100) NOT NULL,
   -> tutorial_author VARCHAR(40) NOT NULL,
   -> submission_date DATE,
   -> PRIMARY KEY ( tutorial_id )
   -> );
Query OK, 0 rows affected (0.16 sec)
mysql>
```

NOTE: MySQL does not terminate a command until you give a semi colon (;) at the end of SQL command.

Creating Tables Using PHP Script:

To create new table in any existing database you would need to use PHP function **mysql_query()**. You will pass its second argument with proper SQL command to create a table.

Example:

Here is an example to create a table using PHP script:

```
<html>
<head>
<title>Creating MySQL Tables</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully<br />';
$sql = "CREATE TABLE tutorials_tbl( ".
       "tutorial_id INT NOT NULL AUTO_INCREMENT, ".
       "tutorial_title VARCHAR(100) NOT NULL, ".
       "tutorial_author VARCHAR(40) NOT NULL, ".
       "submission_date DATE, ".
       "PRIMARY KEY ( tutorial_id )); ";
mysql_select_db( 'TUTORIALS' );
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not create table: ' . mysql_error());
```

```
}
echo "Table created successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

Drop MySQL Tables

It is very easy to drop an existing MySQL table. But you need to be very careful while deleting any existing table because data lost will not be recovered after deleting a table.

Syntax:

Here is generic SQL syntax to drop a MySQL table:

```
DROP TABLE table_name ;
```

Dropping Tables from Command Prompt:

This needs just to execute **DROP TABLE** SQL command at mysql> prompt.

Example:

Here is an example which deletes **tutorials_tbl**:

```
root@host# mysql -u root -p
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> DROP TABLE tutorials_tbl
Query OK, 0 rows affected (0.8 sec)
mysql>
```

Dropping Tables Using PHP Script:

To drop an existing table in any database you would need to use PHP function **mysql_query()**. You will pass its second argument with proper SQL command to drop a table.

Example:

```
<html>
<head>
<title>Creating MySQL Tables</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
```

```
echo 'Connected successfully<br />';
$sql = "DROP TABLE tutorials_tbl";
mysql_select_db( 'TUTORIALS' );
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not delete table: ' . mysql_error());
}
echo "Table deleted successfully\n";
mysql_close($conn);
?>
</body>
</html>
```

MySQL Insert Query

To insert data into MySQL table you would need to use SQL **INSERT INTO** command. You can insert data into MySQL table by using mysql> prompt or by using any script like PHP.

Syntax:

Here is generic SQL syntax of INSERT INTO command to insert data into MySQL table:

```
INSERT INTO table_name ( field1, field2,...fieldN )
                       VALUES
                       ( value1, value2,...valueN );
```

To insert string data types it is required to keep all the values into double or single quote, for example:- **"value"**.

Inserting Data from Command Prompt:

This will use SQL INSERT INTO command to insert data into MySQL table tutorials_tbl

Example:

Following example will create 3 records into **tutorials_tbl** table:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> INSERT INTO tutorials_tbl
     ->(tutorial_title, tutorial_author, submission_date)
     ->VALUES
     ->("Learn PHP", "John Poul", NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO tutorials_tbl
     ->(tutorial_title, tutorial_author, submission_date)
     ->VALUES
     ->("Learn MySQL", "Abdul S", NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO tutorials_tbl
     ->(tutorial_title, tutorial_author, submission_date)
     ->VALUES
```

```
    ->("JAVA Tutorial", "Sanjay", '2007-05-06');
Query OK, 1 row affected (0.01 sec)
mysql>
```

NOTE: Please note that all the arrow signs (->) are not part of SQL command they are indicating a new line and they are created automatically by MySQL prompt while pressing enter key without giving a semi colon at the end of each line of the command.

In the above example we have not provided tutorial_id because at the time of table create we had given AUTO_INCREMENT option for this field. So MySQL takes care of inserting these IDs automatically. Here **NOW()** is a MySQL function which returns current date and time.

MySQL SELECT Query

The SQL **SELECT** command is used to fetch data from MySQL database. You can use this command at mysql> prompt as well as in any script like PHP.

Syntax:

Here is generic SQL syntax of SELECT command to fetch data from MySQL table:

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE Clause]
[OFFSET M ][LIMIT N]
```

- You can use one or more tables separated by comma to include various condition using a WHERE clause. But WHERE clause is an optional part of SELECT command.
- You can fetch one or more fields in a single SELECT command.
- You can specify star (*) in place of fields. In this case SELECT will return all the fields
- You can specify any condition using WHERE clause.
- You can specify an offset using **OFFSET** from where SELECT will start returning records. By default offset is zero
- You can limit the number of returned using **LIMIT** attribute.

Fetching Data from Command Prompt:

This will use SQL SELECT command to fetch data from MySQL table tutorials_tbl

Example:

Following example will return all the records from **tutorials_tbl** table:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           1 | Learn PHP      | John Poul       | 2007-05-21      |
|           2 | Learn MySQL    | Abdul S         | 2007-05-21      |
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
```

```
3 rows in set (0.01 sec)

mysql>
```

MySQL WHERE Clause

We have seen SQL **SELECT** command to fetch data from MySQL table. We can use a conditional clause called **WHERE** clause to filter out results. Using WHERE clause we can specify a selection criteria to select required records from a table.

Syntax:

Here is generic SQL syntax of SELECT command with WHERE clause to fetch data from MySQL table:

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE condition1 [AND [OR]] condition2.....
```

- You can use one or more tables separated by comma to include various condition using a WHERE clause. But WHERE clause is an optional part of SELECT command.
- You can specify any condition using WHERE clause.
- You can specify more than one conditions using **AND** or **OR** operators.
- A WHERE clause can be used alongwith DELETE or UPDATE SQL command also to specify a condition.

The **WHERE** clause works like a if condition in any programming language. This clause is used to compare given value with the field value available in MySQl table. If given value from outside is equal to the available field value in MySQL table then it returns that row.

Here is the list of operators which can be used with **WHERE** clause.

Assume field A holds 10 and field B holds 20 then:

Operator	Description	Example
=	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A = B) is not true.
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.

>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

The WHERE clause is very useful when you want to fetch selected rows from a table, Specially when you use **MySQL Join**. Joins are discussed in another chapter.

It is a common practice to search records using **Primary Key** to make search fast.

If given condition does not match any record in the table then query would not return any row.

Fetching Data from Command Prompt:

This will use SQL SELECT command with WHERE clause to fetch selected data from MySQL table tutorials_tbl

Example:

Following example will return all the records from **tutorials_tbl** table for which author name is **Sanjay**:

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl WHERE tutorial_author='Sanjay';
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
1 rows in set (0.01 sec)

mysql>
```

All the Unless performing a **LIKE** comparison on a string, the comparison is not case sensitive. You can make your search case sensitive using **BINARY** keyword as follows.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl \
          WHERE BINARY tutorial_author='sanjay';
Empty set (0.02 sec)

mysql>
```

MySQL UPDATE Query

There may be a requirement where existing data in a MySQL table need to be modified. You can do so by using SQL **UPDATE** command. This will modify any field value of any MySQL table.

Syntax:

Here is generic SQL syntax of UPDATE command to modify data into MySQL table:

```
UPDATE table_name SET field1=new-value1, field2=new-value2
[WHERE Clause]
```

- You can update one or more field all together.
- You can specify any condition using WHERE clause.
- You can update values in a single table at a time.

The WHERE clause is very useful when you want to update selected rows in a table.

Updating Data from Command Prompt:

This will use SQL UPDATE command with WHERE clause to update selected data into MySQL table tutorials_tbl

Example:

Following example will update **tutorial_title** field for a record having tutorial_id as 3.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> UPDATE tutorials_tbl
    -> SET tutorial_title='Learning JAVA'
    -> WHERE tutorial_id=3;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

MySQL DELETE Query

If you want to delete a record from any MySQL table then you can use SQL command **DELETE FROM**. You can use this command at mysql> prompt as well as in any script like PHP.

Syntax:

Here is generic SQL syntax of DELETE command to delete data from a MySQL table:

```
DELETE FROM table_name [WHERE Clause]
```

- If WHERE clause is not specified then all the records will be deleted from the given MySQL table.

- You can specify any condition using WHERE clause.
- You can delete records in a single table at a time.

The WHERE clause is very useful when you want to delete selected rows in a table.

Deleting Data from Command Prompt:

This will use SQL DELETE command with WHERE clause to delete selected data into MySQL table tutorials_tbl

Example:

Following example will delete a record into tutorial_tbl whose tutorial_id is 3.

```
root@host# mysql -u root -p password;
Enter password:*******
mysql> use TUTORIALS;
Database changed
mysql> DELETE FROM tutorials_tbl WHERE tutorial_id=3;
Query OK, 1 row affected (0.23 sec)

mysql>
```

Further Detail:

Refer to the link http://www.tutorialspoint.com/mysql

▪ Learn ebXML ▪ Learn Euphoria ▪ Learn GDB Debugger ▪ Learn Makefile ▪ Learn Parrot ▪ Learn Perl Script ▪ Learn PHP Script ▪ Learn Six Sigma ▪ Learn SEI CMMI ▪ Learn WiMAX ▪ Learn Telecom Billing	▪ Learn XML-RPC ▪ Learn UML ▪ Learn UNIX ▪ Learn WSDL ▪ Learn i-Mode ▪ Learn GPRS ▪ Learn GSM ▪ Learn WAP ▪ Learn WML ▪ Learn Wi-Fi
webmaster@TutorialsPoint.com	