Data-Base

- My Sql
  - MySQL Tutorial
  - MySQL - Introduction
  - MySQL - Installation
  - MySQL - Administration
  - MySQL - PHP Syntax
  - MySQL - Connection
  - MySQL - Create Database
  - Drop MySQL Database
  - Selecting MySQL Database
  - MySQL - Data Types

My Sql

MySQL Tutorial

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 will give you quick start with MySQL and make you comfortable with MySQL programming.

Audience

This reference has been prepared for the beginners to help them understand the basics to advanced concepts related to MySQL languages.

Prerequisites

Before you start doing practice with various types of examples given in this reference, I'm making an assumption that you are already aware about what is database, especially RDBMS and what is a computer programming language.

MySQL - Introduction

What is a 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 also 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.

Nowadays, we use relational database management systems (RDBMS) to store and manage 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 the MySQL database system, let us revise a few definitions related to the 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 only find 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 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.

MySQL - Installation

All downloads for MySQL are located at MySQL Downloads. Pick the version number of **MySQL Community Server** which is required along with the platform you will be running it on.

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 website −

MySQL − The MySQL database server manages the databases and tables, controls user access and processes the SQL queries.
MySQL-client − MySQL client programs, which make 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 will usually work on other Linux variants with no difficulty.

Now, you will need to adhere to the steps given below, to proceed with the installation −

Login to the system using the **root** user.
Switch to the directory containing the RPMs.
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
```

The above command takes care of installing the MySQL server, creating a user of MySQL, creating necessary configuration and starting the 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 the /var/lib/mysql directory.

The following code box has an 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

The 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 the setup.exe file.

The 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 everything is working as it should be via some simple tests.

Use the mysqladmin Utility to Obtain Server Status

Use **mysqladmin** binary to check the 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 the 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 a message, then there may be some problem in your installation and you would need some help to fix it.

Execute simple SQL commands using the MySQL Client

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

You can 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 commands at the 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 the client, you need to set a root password as given in the following code block −

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

Now to make a connection to your MySQL server, you would have to use the 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 everytime 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 the MySQL server at boot time, then make sure you have the following entry in the /etc/rc.local file.

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

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

MySQL - Administration

Running and Shutting down MySQL Server

First check if your MySQL server is running or not. You can use the following command to check it −

```
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 the following command −

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

Now, if you want to shut down an already running MySQL server, then you can do it by using the following command −

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

Setting Up a MySQL User Account

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

The following program is an example of adding a 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. The following example will add user **zara** with password **zara123** for a particular database, which is named as **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 the MySQL database table called as **user**.

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

The /etc/my.cnf File Configuration

In most of the cases, you should not touch this file. By default, it will have the 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 the error log, otherwise you should not change any entry in this table.

Administrative MySQL Command

Here is the list of the important MySQL commands, which you will use time to time to work with MySQL database −

USE Databasename − This will be used to select a database in the MySQL workarea.
SHOW DATABASES − Lists out 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.

In the next chapter, we will discuss regarding how PHP Syntax is used in MySQL.

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 consider reading the PERL Tutorial.

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

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

```
mysqli 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 −

```
$mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
mysqli->query(,"SQL statement");
```

The following example shows a generic syntax of PHP to call any MySQL function.

```
<html>
   <head>
      <title>PHP with MySQL</title>
   </head>

   <body>
      <?php
         $retval = mysqli - > function( value, [ value,...]);
         if( !$retval ) {
            die ( "Error: a related error message" );
         }
         // Otherwise MySQL  or PHP Statements
      ?>
   </body>
</html>
```

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

MySQL - Connection

MySQL Connection Using MySQL Binary

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

Example

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

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

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

The following code block shows the result of above code −

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

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

```
mysql> exit
Bye
```

MySQL Connection Using PHP Script

PHP provides **mysqli** contruct or **mysqli_connect()** function to open a database connection. This function takes six parameters and returns a MySQL link identifier on success or FALSE on failure.

Syntax

```
$mysqli = new mysqli($host, $username, $passwd, $dbName, $port, $socket);
```

Sr.No.	Parameter & Description
1	**$host** Optional − The host name running the database server. If not specified, then the default value will be **localhost:3306**.
2	**$username** Optional − The username accessing the database. If not specified, then the default will be the name of the user that owns the server process.
3	**$passwd** Optional − The password of the user accessing the database. If not specified, then the default will be an empty password.
4	**$dbName** Optional − database name on which query is to be performed.
5	**$port** Optional − the port number to attempt to connect to the MySQL server.
6	**$socket** Optional − socket or named pipe that should be used.

You can disconnect from the MySQL database anytime using another PHP function **close()**.

Syntax

```
$mysqli->close();
```

Example

Try the following example to connect to a MySQL server −

Copy and paste the following example as mysql_example.php −

```
<html>
   <head>
      <title>Connecting MySQL Server</title>
   </head>
   <body>
      <?php
         $dbhost = 'localhost';
         $dbuser = 'root';
         $dbpass = 'root@123';
         $mysqli = new mysqli($dbhost, $dbuser, $dbpass);

         if($mysqli->connect_errno ) {
            printf("Connect failed: %s<br />", $mysqli->connect_error);
            exit();
         }
         printf('Connected successfully.<br />');
         $mysqli->close();
      ?>
   </body>
</html>
```

Output

Access the mysql_example.php deployed on apache web server and verify the output.

```
Connected successfully.
```

MySQL - Create Database

Create Database Using mysqladmin

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

Example

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

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

This will create a MySQL database called TUTORIALS.

Create a Database using PHP Script

PHP uses **mysqli query()** or **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

```
$mysqli->query($sql,$resultmode)
```

Sr.No.	Parameter & Description
1	**$sql** Required - SQL query to create a MySQL database.
2	**$resultmode** Optional - Either the constant MYSQLI_USE_RESULT or MYSQLI_STORE_RESULT depending on the desired behavior. By default, MYSQLI_STORE_RESULT is used.

Example

Try the following example to create a database −

Copy and paste the following example as mysql_example.php −

```
<html>
   <head><title>Creating MySQL Database</title></head>
   <body>
      <?php
         $dbhost = 'localhost';
         $dbuser = 'root';
         $dbpass = 'root@123';
         $mysqli = new mysqli($dbhost, $dbuser, $dbpass);

         if($mysqli->connect_errno ) {
            printf("Connect failed: %s<br />", $mysqli->connect_error);
            exit();
         }
         printf('Connected successfully.<br />');

         if ($mysqli->query("CREATE DATABASE TUTORIALS")) {
            printf("Database TUTORIALS created successfully.<br />");
         }
         if ($mysqli->errno) {
            printf("Could not create database: %s<br />", $mysqli->error);
         }
         $mysqli->close();
      ?>
   </body>
</html>
```

Output

Access the mysql_example.php deployed on apache web server and verify the output.

```
Connected successfully.
Database TUTORIALS created successfully.
```

Drop MySQL Database

Drop a Database using mysqladmin

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

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

Here is an example to delete a database(TUTORIALS) created in the 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 **mysqli query()** or **mysql_query()** function to drop a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax

```
$mysqli->query($sql,$resultmode)
```

Sr.No.	Parameter & Description
1	**$sql** Required - SQL query to drop a MySQL database.
2	**$resultmode** Optional - Either the constant MYSQLI_USE_RESULT or MYSQLI_STORE_RESULT depending on the desired behavior. By default, MYSQLI_STORE_RESULT is used.

Example

Try the following example to drop a database −

Copy and paste the following example as mysql_example.php −

```
<html>
   <head><title>Dropping MySQL Database</title></head>
   <body>
      <?php
         $dbhost = 'localhost';
         $dbuser = 'root';
         $dbpass = 'root@123';
         $mysqli = new mysqli($dbhost, $dbuser, $dbpass);

         if($mysqli->connect_errno ) {
            printf("Connect failed: %s<br />", $mysqli->connect_error);
            exit();
         }
         printf('Connected successfully.<br />');

         if ($mysqli->query("Drop DATABASE TUTORIALS")) {
            printf("Database TUTORIALS dropped successfully.<br />");
         }
         if ($mysqli->errno) {
            printf("Could not drop database: %s<br />", $mysqli->error);
         }
         $mysqli->close();
      ?>
   </body>
</html>
```

Output

Access the mysql_example.php deployed on apache web server and verify the output.

```
Connected successfully.
Database TUTORIALS dropped successfully.
```

Selecting MySQL Database

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

Selecting MySQL Database from the Command Prompt

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

Example

Here is an example to select a database called **TUTORIALS** −

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

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

NOTE − All the database names, table names, table fields name are case sensitive. So you would have to use the proper names while giving any SQL command.

Selecting a MySQL Database Using PHP Script

PHP uses **mysqli_select_db** function to select the database on which queries are to be performed. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax

```
mysqli_select_db ( mysqli $link , string $dbname ) : bool
```

Sr.No.	Parameter & Description
1	**$link** Required - A link identifier returned by mysqli_connect() or mysqli_init().
2	**$dbname** Required - Name of the database to be connected.

Example

Try the following example to select a database −

Copy and paste the following example as mysql_example.php −

```
<html>
   <head>
      <title>Selecting MySQL Database</title>
   </head>
   <body>
   <?php
      $dbhost = 'localhost';
      $dbuser = 'root';
      $dbpass = 'root@123';
      $conn = mysqli_connect($dbhost, $dbuser, $dbpass);

      if(! $conn ) {
         die('Could not connect: ' . mysqli_error($conn));
      }
      echo 'Connected successfully<br />';
      $retval = mysqli_select_db( $conn, 'TUTORIALS' );
      if(! $retval ) {
         die('Could not select database: ' . mysqli_error($conn));
      }
      echo "Database TUTORIALS selected successfully\n";
      mysqli_close($conn);
   ?>
   </body>
</html>
```

Output

Access the mysql_example.php deployed on apache web server and verify the output.

```
Database TUTORIALS selected successfully
```

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. For example, do not define a field 10 characters wide, if you know you are only going to use 2 characters. These type 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
- String Types.

Let us now discuss them in detail.

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 20 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 the 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 as follows −

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 1st, 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 a HH:MM:SS format.
YEAR(M) − Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

String Types

Although the numeric and date types are fun, most data you'll store will be in a 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 the sorts and comparisons on the 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.

In the next chapter, we will discuss how to create tables in MySQL.