MySQL Basics with PHP

Gayanath Jayarathne

MySQL Basics with PHP

Learn Essential MySQL in Just 99 Pages

Gayanath Jayarathne

This book is for sale at http://leanpub.com/mysqlbasics

This version was published on 2013-09-02

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Gayanath Jayarathne

PHP Fundamentals

To my family and friends

Contents

Preface

This book walks you through the essential MySQL skills a programmer should have in just 90 pages. The topics were chosen based on my years of experience in MySQL related applications.

I have covered the most frequently used features of MySQL in this book and have a chapter on explaining the advanced features so that once you know the basics, you can go ahead mastering the advanced features.

About Me

I started my professional career about six years ago and have gained sound knowledge on PHP, MySQL and related technologies. I experienced learning challenges of these technologies earlier in my career and have seen how other beginners struggle.

As a way of sharing my knowledge and helping beginners, I launched PHPKnowHow.com[1] in 2010.

About the Book

If you are a MySQL beginner struggling to grasp the basics, this book is for you. You will learn creating and managing MySQL users, creating databases and tables, CRUD (Insert, Select, Update, Delete) operations, and Joins.

Additionally you will learn how to use MySQL in PHP with PHP's MySQL Improved Extension. If PHP is not your programming language, you would still find the book useful since PHP usage is only an addition to the book.

Formatting of Code Examples

In some code, you would see a backslash at line-endings for terms that continue to next line. When you try these examples, remember to remove the backslashes.

How to Provide Feedback

I really appreciate your feedback on this book. Whether it's your opinion or any mistake you found, please send your feedback to **bookfeedback@phpknowhow.com**

[1] http://www.phpknowhow.com/

Chapter 1 - Introduction to MySQL

MySQL[2] is a popular database management system. It has a free and open source version. With its rich features, it has been the choice for many database driven PHP applications.

MySQL History

MySQL was founded by Michael Widenius[3] and David Axmark[4] in 1994. It was then developed under the company MySQL AB[5] where the company provided both open source and commercial licenses.

In January 2008, Sun Microsystems[6] acquired MySQL AB and in January 2010, Oracle[7] acquired Sun Microsystems making MySQL owned by Oracle. Read the MySQL Milestones in Wikipedia[8] for more information.

Since Oracle is primarily a proprietary software company, concerns have been raised about the future of MySQL. MySQL forks like MariaDB[9] and alternatives like PostgreSQL[10] have gained more attention with these concerns. However still MySQL is used by many open source software and large scale websites.

MySQL Infrastructure

A MySQL installation provides support for creating databases in it and allows applications to communicate with these databases when correct credentials are given via a database connection.

A MySQL database consists of set of data tables. MySQL uses Structured Query Language[11] for its commands (MySQL has also got few non-SQL commands).

We can use SQL to create a database and then create tables in it according to the format we want. Usually in a web application, database and respective data tables are created at installer time. Then data is inserted, fetched, updated and deleted from data tables in the runtime of the application.

---

[2] http://www.mysql.com/
[3] https://en.wikipedia.org/wiki/Michael_Widenius
[4] https://en.wikipedia.org/wiki/David_Axmark
[5] https://en.wikipedia.org/wiki/MySQL_AB
[6] https://en.wikipedia.org/wiki/Sun_Microsystems
[7] http://www.oracle.com/
[8] https://en.wikipedia.org/wiki/MySQL#Milestones
[9] https://mariadb.org/
[10] http://www.postgresql.org/
[11] http://en.wikipedia.org/wiki/SQL

id	first_name	last_name	age	joined_date	records
1	Robin	Jackman	34	2000-03-18	
2	Taylor	Edward	42	2004-08-10	

Above is a likely format if you are to store employee data in a MySQL table. Following is the SQL you need to create this table in a given database (Don't worry if you don't know SQL yet).

```
create table `employee` (
  `id` int(10),
  `first_name` varchar(40),
  `last_name` varchar(40),
  `age` tinyint(3),
  `joined_date` date,
  `records` text,
  primary key  (`id`)
);
```

Data Types

You can see five MySQL data types in the table definition above: **int** for integers, **varchar** for short text, **tinyint** for short integers, **date** for dates, and **text** for long text. In addition to these, MySQL provides some more data types[12]. Choosing the correct data type for each column is important for an optimal database.

Relational Behavior

Like many other enterprise-level database management systems, MySQL is a relational database management system[13] and lets you store related data in multiple tables and have relations among them.

For example, assume that the company in the example above also needs to store the department details. Instead of storing these in the **employee** table, you can create two more tables and store department details in them to avoid data redundancy. The first one can be defined as **department** and contains department details.

id	name	contact_details
1	Sales	
2	Marketing	
3	Human Resource	

[12] http://dev.mysql.com/doc/refman/5.5/en/data-types.html

[13] http://en.wikipedia.org/wiki/Relational_database_management_system

The second one (**employee_department**) can be defined to store the relationships between the employee and department tables. Provided that an employee's department can be changed time to time or an employee can work in multiple departments at the same time, this table would have records as shown below.

employee_id	department_id
1	1
1	2
2	3

Now assume that you want to show employee name and the departments they have worked in your application as below.

Robin Jackman : **Sales**
Robin Jackman : **Marketing**
Taylor Edward : **Human Resource**

You can fetch the records needed for the above representation using the following SQL query.

```
SELECT a.first_name, a.last_name, c.name
FROM employee a, employee_department b, department c
WHERE a.id = b.employee_id
AND b.department_id = c.id
```

Naming Conventions

There isn't a standard naming convention for MySQL. Following are some common conventions.

- All names are lower case (Ex: employee).
- Words are separated by underscore (Ex: first_name).
- Use singular form (Ex: employee, not employees).

Database Operations

MySQL allows all four CRUD[14] (Create, Read, Update, Delete) operations. Often you would see these operations as Insert, Select, Update, and Delete relevant to their SQL statements.

Storage Engines

MySQL has several Storage Engines[15] that enable different features on data tables. For example, if you want to automatically delete all the corresponding records in **employee_department** table when an employee is deleted from the **employee** table, then you have to use **InnoDB** storage engine (for defining constraints).

[14] http://en.wikipedia.org/wiki/Create,_read,_update_and_delete
[15] http://dev.mysql.com/doc/refman/5.5/en/storage-engines.html

SQL Modes

MySQL has the ability to run in different SQL modes[16], enabling different restrictions. For example, at default configuration, you can enter 0000-00-00 for a date field. But if you are running in **Traditional** mode, MySQL will throw an error when you try to insert 0000-00-00 into a date field, since it's not a valid date.

You can set the SQL mode per session (generally for the span of starting and stopping the MySQL server) or set it permanently by configuring the MySQL configuration file (**my.ini** in Windows and **my.cnf** in Mac OS/Linux).

User Types

MySQL allows the creation of users and to assigning different privileges for defined users. User credentials are needed when an application needs to make a connection with MySQL and execute operations on a database.

Allowing for different privileges makes operations on a database safe. For example, for the same database, you can have one user with all the privileges and another with only viewing privileges.

Installing MySQL

MySQL can be downloaded[17] and installed as a separate application. MySQL supports several operating systems, including Windows, Mac OS, and Linux. Once installed, you have to start MySQL server before accessing databases.

If you used an AMP stack like XAMPP[18], you could install Apache, MySQL, and PHP in a single installation. These stacks also provide an interface where you can start/stop Apache and MySQL.

[16] http://dev.mysql.com/doc/refman/5.5/en/server-sql-mode.html
[17] http://dev.mysql.com/downloads/mysql/
[18] http://www.phpknowhow.com/basics/working-with-xampp/

Chapter 2 - Using MySQL Command-line

MySQL comes with a command-line interface[19] that let you run MySQL commands and SQL queries. While there are graphical tools like phpMyAdmin[20] and MySQL Workbench[21], command-line interface will come in handy when you manage more databases and when you get more familiar with MySQL.

MySQL Path

You should be able to run **mysql** and other command-line utilities like **mysqladmin** and **mysqldump** (discussed later) from any folder. If these utilities are not accessible from any directory, add the path to the utilities to a global path variable in your operating system.

If you have installed XAMPP in Windows, the path would be the **bin** folder under **mysql** folder. For example, if you have installed XAMPP under C drive, path would be **C:\xampp\mysql\bin**. You can verify the path by making sure the utilities are available in the chosen path.

Add this path to the **path** environment variable in Windows, and then you will be able to access the utilities from any folder.

Logging into MySQL

You can log in to MySQL as **root** user (generally the user with all the privileges) by typing the command below.

```
mysql -u root -p
```

The command above will prompt you to enter the password for user **root**. If your MySQL installation is new and you haven't changed the **root** password, most of the time the **root** password is blank (just press the **Enter** key).

After that, you would see a prompt like below that lets you type commands. Prior to the prompt, you would see few instructions and the MySQL version number.

[19] http://dev.mysql.com/doc/refman/5.5/en/mysql.html
[20] http://www.phpmyadmin.net/
[21] http://www.mysql.com/products/workbench/

```
mysql>
```

Executing Commands

All SQL commands you type at the MySQL prompt should have a semicolon (;) at their ends. The commands will not run till you enter a semicolon (It's possible to use **\G** instead of semicolon as explained below).

In addition to the SQL commands, MySQL has its own set of commands. To see these commands, type **help** at the MySQL prompt as below. These commands aren't required to have a semicolon at the end. After typing a command, hit **Enter** key to execute the command.

```
mysql> help
```

Command-line Pretty Output

If you find the output of a certain SQL command difficult to read, try **\G** in place of the semicolon as shown in the following example. This will display the output in a vertical format and remove surrounding dashed lines.

```
mysql> SHOW TABLE STATUS FROM company_db \G
```

SHOW TABLE STATUS command is covered in **Database Commands** chapter.

Multiple Line Commands

To achieve clarity, you can span a command over multiple lines. Just hit the **Enter** key after each line, and MySQL will prompt an arrow indicating a new line. The following is a multiple line SQL command to create a data table.

```
mysql> create table `employee` (
    ->   `id` int(10),
    ->   `first_name` varchar(40),
    ->   `last_name` varchar(40),
    ->   `age` tinyint(3),
    ->   `joined_date` date,
    ->   `records` text,
    ->   primary key  (`id`)
    -> );
```

How to Clear Command-line

If you are on a Mac OS or a Linux command-line, you can use **Ctrl+L** for clearing the screen and **Ctrl+U** for clearing the current line.

Logging out of MySQL

Use the **exit** command to log out of MySQL.

```
mysql> exit
```

Chapter 3 - MySQL GUI Tools

While you can run all MySQL statements in command-line, having a tool with Graphical User Interface (GUI) can speed up your development.

A GUI tool can be especially helpful when you want to view data and table information. Following are a few popular and free MySQL GUI tools.

phpMyAdmin[22]

- A web application written in PHP.
- Rich set of features.
- Works in any OS with required PHP and MySQL setup.

SQL Buddy[23]

- A web application written in PHP.
- Clean interface.
- Painless installation (Just extract to the web folder).
- Ability to log in via MySQL user accounts without creating user accounts for the tool.
- Works in any OS with required PHP and MySQL setup.

MySQL Workbench[24]

- A Desktop application.
- Rich set of features (Database Design, Server Administration).
- Versions available for Microsoft Windows, Mac OS, Linux.

---

[22] http://www.phpmyadmin.net/

[23] http://sqlbuddy.com/

[24] http://www.mysql.com/products/workbench/

Chapter 4 - Managing MySQL Users

In MySQL, you can create user accounts[25] with different privileges. Privileges can vary from accessing several databases to accessing only one column in a table.

Root User

By default, MySQL has a super user called **root** that has all the privileges. You need to be logged in as **root** to execute many MySQL administrative tasks, including managing users.

Changing Root Password

If you didn't specifically set the **root** password when installing MySQL, most of the times it would be empty. If the **root** password is empty, make sure to reset it with a proper password for better security.

In command-line, you can use the following command to change **root** password. Type your preferred password in place of **newpassword**. After hitting the **Enter** key, it will ask you to enter the current password. If the current password is empty, just hit the **Enter** key.

```
mysqladmin -u root -p password 'newpassword'
```

Logging as a User

Use the following command to log in as **root** user. For logging in as a different user, type that username in place of **root**. After hitting the **Enter** key, it will ask you to enter the password. After entering the correct password, you would see the MySQL prompt (**mysql>**) where you can enter MySQL commands.

```
mysql -u root -p
```

[25] http://dev.mysql.com/doc/refman/5.5/en/adding-users.html

Viewing Existing Users

MySQL user details are stored in a table called **user** of a default database called **mysql**. In this **user** table, usernames are stored in a column called **user**, and corresponding host names are stored in a column called **host**.

Based on these facts, you can use the following SQL command to view the username and the host of existing users. You need to log in as **root** first.

```
SELECT user, host FROM mysql.user;
```

Creating a New User

For creating a user called **robin** with the password **robin123** at **localhost** (is the default host most of the time), log in as **root** and use the following command.

```
CREATE USER 'robin'@'localhost' IDENTIFIED BY 'robin123';
```

Granting Privileges to a User

MySQL has a series of privileges[26]. For a general PHP application, you only need a user with **SELECT**, **INSERT**, **UPDATE**, and **DELETE** privileges for the database you chose. You can grant these privileges to a user called **ronbin** for a database called **my_database** using the following command.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON my_database.* TO 'robin'@'localhost';
```

To grant all the privileges, use **ALL** as below.

```
GRANT ALL ON my_database.* TO 'robin'@'localhost';
```

Changing a User Password

To change the password of a user called **robin** to **robin456**, log in as the **root** user and use the following command.

[26] http://dev.mysql.com/doc/refman/5.5/en/privileges-provided.html

```
SET PASSWORD FOR 'robin'@'localhost' = PASSWORD('robin456');
```

Deleting a User

Log in as **root** and use the following command to delete user **robin**.

```
DROP USER robin@localhost;
```

Be careful when you delete a user, since applications that used the credentials of a deleted user may malfunction.

Summary

PHP needs the host name, username, and password of a privileged MySQL user to connect to a MySQL database. When it comes to production-level PHP applications, instead of using **root** user, for improved security it's a good practice to use a dedicated user with only the required privileges.

You will only need to deal with MySQL users when you manage your own web server or when you develop web applications locally. If you are on a shared web-hosting environment, most of the time, you will be provided privileged MySQL user accounts or a GUI tool to manage MySQL users.

Chapter 5 - PHP MySQL Connections

PHP provides three ways to connect with MySQL: MySQL Extension, MySQLi Extension, and PDO. Read the overview at the PHP manual[27] for more information and comparison of these three methods.

MySQL Extension

This was the first method PHP provided to interact with MySQL. This extension provides a set of built-in functions for connecting and making database queries. If a PHP built-in function starts with term **mysql**, it's a function of this extension. For example, the following is the function used for connecting to MySQL.

```
mysql_connect('hostname', 'username', 'password');
```

MySQL extension only supports features of MySQL versions prior to 4.1.3. You can't use advanced features of MySQL like Prepared Statements[28] that was introduced after MySQL version 5 with this extension.

But since many PHP applications only need to execute CRUD operations[29], you will still see functions of this extension in use. As of PHP version 5.3, there is no further development in this extension and it is only maintained.

MySQL Improved (MySQLi) Extension

This is the recommended method for executing MySQL operations in PHP. MySQLi supports basic and advanced MySQL features. It's available from PHP version 5. MySQLi provides two interfaces to interact with MySQL: Procedural Interface and Objected Oriented Interface.

MySQLi Procedural Interface

This interface provides built-in functions like MySQL Extension. If a PHP built-in function starts with term **mysqli**, it's a function of this interface. Below is how you need to connect with MySQL via this interface.

[27] http://php.net/manual/en/mysqli.overview.php

[28] http://dev.mysql.com/doc/refman/5.5/en/sql-syntax-prepared-statements.html

[29] http://en.wikipedia.org/wiki/Create,_read,_update_and_delete

```
mysqli_connect('hostname', 'username', 'password');
```

MySQLi Objected Oriented Interface

This interface allows instantiation of an instance of built-in class **mysqli**. After instantiating an instance properly, the instance will have methods and properties to interact with MySQL. Below is how you instantiate an instance of **mysqli**.

```
$mysqli = new mysqli('hostname', 'username', 'password');
```

PHP Data Objects (PDO)

PDO is a built-in object-oriented database abstraction layer that is available from PHP version 5. It abstracts specific database operations and provides a common interface for interacting with all supported databases. That is, if you use PDO for database operations in your PHP application, you can switch from MySQL to PostgreSQL[30] with minimal code changes.

Which Method to Use?

Obviously you don't need to use MySQL Extension. PHP discourages using it, and since MySQLi Procedural Interface provides similar and more functions, there is no reason for using it.

If your PHP application is object oriented and you need to have the flexibility of using more than one Database Management System (DBMS), then you can use PDO. For example, many PHP frameworks use PDO, since they need to give the user of the framework the option of choosing the DBMS.

But PDO doesn't support all MySQL advanced features. For instance, it doesn't support MySQL's ability of using multiple statements. Before choosing PDO, check whether it facilitates all the database operations you want to have in your PHP application.

If you are sure that your PHP application only uses MySQL, then the best option is to go with MySQLi, since it supports MySQL features more than the other two methods do and since it's being actively developed.

If your PHP application is more than a few PHP scripts, you would tend to use MySQLi Objected Oriented Interface with other object oriented code for code-reuse and maintainability.

[30] http://www.postgresql.org/

Chapter 6 - MySQLi Procedural Functions

MySQL Improved Extension (MySQLi) provides a Procedural Interface as well as an Object Oriented Interface. In this chapter, we will look into some of the common MySQLi Procedural functions.

mysqli_connect()

This function is used for connecting to MySQL. Before doing any database operation, you need to connect to MySQL. On success, this function returns a link identifier that you can use in other MySQLi functions. On failure, it will throw an error.

Following is how a user named **robin** with password **robin123** needs to connect to a database called **company_db** at **localhost**. User **robin** should have privileges to access **company_db**.

```
$link = mysqli_connect('localhost', 'robin', 'robin123', 'company_db');
```

If your MySQL port is different from the default one (3308), you need to give the port number as the fifth parameter.

```
$link = mysqli_connect('localhost', 'robin', 'robin123', 'company_db', '3800');
```

mysqli_connect_error()

mysqli_connect() throws an error at failure, and **mysqli_connect_error()** stores the error of the last call to **mysqli_connect()**. If there is no error, it returns NULL.

To try out a **mysqli_connect()** error, stop MySQL server and call **mysqli_connect()**. If you have enabled PHP errors, you would see an error that includes information like below. **mysqli_connect_error()** would return the same message.

Can't connect to MySQL server on 'localhost'

In practice, it's not good to show error messages like these to the users of your PHP application (They may contain sensitive data, and they can look too technical).

Therefore you can use the error suspension operator in front of **mysqli_connect()** to stop it throwing errors, and use **mysqli_connect_error()** to log the error for troubleshooting like below.

```
<?php

$link = @mysqli_connect('localhost', 'robin', 'robin123', 'company_db');

if (mysqli_connect_error()) {
        $logMessage = 'MySQL Error: ' . mysqli_connect_error();
        // Call your logger here.
        die('Could not connect to the database');
}

// Rest of the code goes here

?>
```

mysqli_select_db()

To change the database in use, you can use **mysqli_select_db()**. For example assume that user **robin** also has privileges for a database called **company_new_db**; then you can change the database as below.

```
$link = @mysqli_connect('localhost', 'robin', 'robin123', 'company_db');

// Operations on 'company_db'

mysqli_select_db($link, 'company_new_db');

// Operations on 'company_new_db'
```

You will only need this function if your PHP application deals with more than one database.

mysqli_close()

You can use this function to close a MySQL connection. It returns TRUE on success and FALSE on failure.

```php
$link = @mysqli_connect('localhost', 'robin', 'robin123', 'company_db');

// MySQL operations goes here.

mysqli_close($link);
```

PHP will close open connections and release resources at the end of your PHP script. But it's a good practice to explicitly use **mysqli_close()** at the end of MySQL operations to release resources immediately.

mysqli_query()

This is the function used for executing MySQL queries. It returns FALSE on failure. For SELECT, SHOW, DESCRIBE, and EXPLAIN queries (where there is an output), it returns a MySQL result set (resource) which can be used in functions like **mysqli_fetch_array()**.

For other queries, like INSERT, UPDATE, and DELETE, it returns TRUE on success.

```php
$link = @mysqli_connect('localhost', 'robin', 'robin123', 'company_db');

$query = "SELECT * FROM employee";

if (mysqli_query($link, $query)) {
    // Iterate and display result
} else {
    // Show error
}

mysqli_close($link);
```

mysqli_fetch_array()

This function is used for reading data from a MySQL result set (returned by a **mysqli_query()**). It reads and returns one row of data as an array and then moves the pointer to the next row. When there are no more rows to return, it returns NULL. Because of this behavior, it's often used with a While Loop as below.

```
while ($row = mysqli_fetch_array($result)) {
/* Till there is data, $row will be an array.
 * At the end, $row becomes NULL ending the loop.
 */
}
```

Let's assume the following **employee** table is available in our **company_db** database.

id	first_name	last_name	age	joined_date	records
1	Robin	Jackman	34	2000-03-18	
2	Taylor	Edward	42	2004-08-10	

Below is how we would fetch ID, First Name, and Last Name from this table.

```
<?php

$link = @mysqli_connect('localhost', 'robin', 'robin123', 'company_db');

$query = "SELECT `id`, `first_name`, `last_name` FROM `employee`";

$result = mysqli_query($link, $query);

while ($row = mysqli_fetch_array($result)) {
    echo $row[0] . ': ' . $row[1] . ' ' . $row[2];
    echo '<br />';
}

mysqli_free_result($result);

mysqli_close($link);

?>
```

When you run it, the code above will output the following content in the web browser.

1: Robin Jackman
2: Taylor Edward

In addition to an Indexed array, **mysqli_fetch_array()** also returns an Associated array where keys are the corresponding column names of the table. So, the following code segment will produce the same output.

```
while ($row = mysqli_fetch_array($result)) {
    echo $row['id'] . ': ' . $row['first_name'] . ' ' . $row['last_name'];
    echo '<br />';
}
```

You can limit which array to return as below.

```
// Returns only an Indexed array
mysqli_fetch_array($result, MYSQLI_NUM);

// Returns only an Associated array
mysqli_fetch_array($result, MYSQLI_ASSOC);
```

PHP provides two functions that produce the same results as you would get by passing constants to **mysqli_fetch_array()**.

```
// Same as mysqli_fetch_array($result, MYSQLI_NUM)
mysqli_fetch_row($result);

// Same as mysqli_fetch_array($result, MYSQLI_ASSOC)
mysqli_fetch_assoc($result);
```

mysqli_free_result()

Immediately after using a result set, you can free the memory used for it as below.

```
mysqli_free_result($result);
```

mysqli_num_rows()

mysqli_num_rows() returns the number of rows in a result set. Using it, you can take a different action when the result set is empty.

```
if (mysqli_num_rows($result) > 0) {
    // Proceed with the $result
} else {
    // Show an error message
}
```

mysqli_affected_rows()

This function provides information on the last MySQL query executed. For INSERT, UPDATE, REPLACE, and DELETE, it provides number of rows affected. For SELECT, it returns number of rows in the result set as **mysqli_num_rows()**.

For example, the following is an UPDATE query to update the last name of Taylor in **employee** table. Provided that the **id** field is unique, we know that only one row would be affected from this query.

```
$query = "UPDATE `employee` SET `last_name` = 'Adams' WHERE `id` = 2";

mysqli_query($link, $query);

if (mysqli_affected_rows($link) == 1) {
    // Rest of the code
} else {
    // Show an error message
}
```

mysqli_error()

If there was an error in the last MySQL query, this function will return the error. If there was no error, it would return an empty string.

```
if (!mysqli_query($link, $query)) {
        $logMessage = 'MySQL Error: ' . mysqli_error($link);
        // Call your logger here.
    die('There was an error in the query');
}
```

mysqli_real_escape_string()

Some characters like single quote have special meanings in SQL statements. For example, the single quote is used for wrapping strings. So, if your SQL statement contains these special characters, you need to escape them via **mysqli_real_escape_string()** before sending the query to **mysqli_query()**.

The following call to **mysqli_query()** returns FALSE, since the single quote in O'Neil hasn't been escaped.

```
$query = "SELECT `id` FROM `employee` WHERE `last_name` = 'O'Neil'";
mysqli_query($link, $query);
```

The following call to **mysqli_query()** would return a proper result set (provided that an employee exists with last name O'Neil), since the name is first escaped via **mysqli_real_escape_string()**.

```
$name = mysqli_real_escape_string($link, "O'Neil");
$query = "SELECT `id` FROM `employee` WHERE `last_name` = '$name'";
mysqli_query($link, $query);
```

If your SQL statements are built based on user inputs like those below, it's always a good idea to use this function, since user input may contain special characters.

```
$lastName = mysqli_real_escape_string($link, $_POST['lastName']);
$query = "SELECT `id` FROM `employee` WHERE `last_name` = '$lastName'";
mysqli_query($link, $query);
```

User inputs may contain attempts for security breaches by having special characters execute malicious actions (SQL injection[31]). Escaping user input will reduce the risk of SQL injection.

[31] http://en.wikipedia.org/wiki/SQL_injection

Chapter 7 - MySQL Database Commands

MySQL has four database commands: **show**, **create**, **drop**, and **use**. You can execute these commands in the command-line and in PHP scripts. However since many PHP applications use only one database, it's rare to find these commands in PHP scripts.

Show Databases

Log in to MySQL in command-line as the root user or any other privileged user, and enter the following command at the MySQL prompt. You will be able to see a list of databases allowed for the logged-in user.

```
SHOW DATABASES;
```

You can try this command in PHP as shown below. Replace the **mysqli_connect()** details with the details of a MySQL user in your computer.

```
<?php

$link = mysqli_connect('localhost', 'robin', 'robin123');

$result = mysqli_query($link, 'SHOW DATABASES');

while ($row = mysqli_fetch_array($result)) {

    echo $row[0] . '<br />';

}

?>
```

Create Database

To create a new database, log in as a user who has the privilege to create databases and use the following command. Type the database name you want in place of **company_db**.

For creating and dropping databases, you will need to use **root** user or a user with equivalent privileges.

```
CREATE DATABASE company_db;
```

It's possible to use mysqladmin[32] utility to create a database directly from the command-line, without logging in to MySQL as shown below.

```
mysqladmin -u root -p CREATE company_db
```

In PHP, you can create a database as below.

```
<?php

$link = mysqli_connect('localhost', 'root', 'root123');

$result = mysqli_query($link, 'CREATE DATABASE company_db');

if ($result) {
    echo 'Database was created successfully';
} else {
    echo mysqli_error($link);
}

?>
```

Drop Database

Use the command below to delete a database. Replace **company_db** with the name of the database you want to delete.

```
DROP DATABASE company_db;
```

You can use **mysqladmin** to drop a database directly from the command-line, without logging in to MySQL as shown below. You will be asked to confirm your action.

[32] http://dev.mysql.com/doc/refman/5.5/en/mysqladmin.html

```
mysqladmin -u root -p DROP company_db
```

You can achieve the same result in PHP by passing the **DROP DATABASE** command to **mysqli_query()** as below.

```
<?php

$link = mysqli_connect('localhost', 'root', 'root123');

$result = mysqli_query($link, 'DROP DATABASE company_db');

if ($result) {
    echo 'Database was dropped successfully';
} else {
    echo mysqli_error($link);
}

?>
```

Use Database

In MySQL, before executing any table level commands, you need to choose a database. You can select the database **company_db** to work on using the following command.

```
USE company_db;
```

Once you have selected a database, you can still query tables of other databases by specifying the database name in the query as shown below (**company_db_new** is the database name and **employee** is the table name).

```
SELECT * FROM company_db_new.employee;
```

This will be useful when you quickly need to query a table without changing current database. SELECT statement is covered later in its own chapter.

In PHP, you can simply pass the database to be used as the fourth parameter of **mysqli_connect()**. After connecting, if you want to change the database, use **mysqli_select_db()**.

How to Check the Current Database

Sometimes, when you work on couple of databases, you will need to know the current database in use. The following command will show the current database in use.

```
SELECT DATABASE();
```

How to Get Database Metadata

The following command will show metadata of the database **company_db** (If **company_db** doesn't contain any tables yet, use the commands in Appendix A to create and populate few tables).

```
SHOW TABLE STATUS FROM company_db \G
```

If you only need to know the number of columns and the number of rows each table has, you can use mysqlshow[33] in the command-line (without logging in to mysql) as shown below.

```
mysqlshow -u root -p --count company_db
```

```
+--------------------+----------+------------+
|       Tables       | Columns  | Total Rows |
+--------------------+----------+------------+
| education          |        2 |          3 |
| employee           |        6 |          8 |
| employee_education |        2 |          4 |
| leave_type         |        2 |          2 |
| meeting_user       |        4 |          5 |
| telephone          |        4 |         12 |
| user               |        5 |          7 |
+--------------------+----------+------------+
```

information_schema Database

MySQL has a database called information_schema[34] that stores meta information of other databases in the system. You can check various stats like number of rows each table has and database size (discussed later) with this database.

For example, following query returns data similar to **SHOW TABLE STATUS** command.

```
SELECT * FROM information_schema.TABLES WHERE TABLE_SCHEMA = 'company_db' \G
```

How to Get Database Size

We can query the database **information_schema** for getting the size of a database as shown below.

[33] http://dev.mysql.com/doc/refman/5.5/en/mysqladmin.html
[34] http://dev.mysql.com/doc/refman/5.5/en/information-schema.html

```
SELECT ROUND((SUM(DATA_LENGTH+INDEX_LENGTH))/1024/1024, 2) AS "Size in MB"
FROM information_schema.TABLES WHERE TABLE_SCHEMA = 'company_db';
```

The command above will display the size of database **company_db** in megabytes. The ROUND() function is used to round the value to two decimal places.

Similarly, we can get the size of a table. Refer to the chapter **MySQL Table Commands** to see the query for getting table size. Refer to the chapter **MySQL Select Statement** for more information on SELECT statement and SUM() function.

How to Rename a Database

MySQL doesn't offer a command to rename a database (It offered a command in version 5.1.7, but removed it in version 5.1.23[35]). However, there are workarounds. Refer to the discussion at Stack Overflow[36] for few approaches.

[35] http://dev.mysql.com/doc/refman/5.1/en/rename-database.html

[36] http://stackoverflow.com/questions/67093/how-do-i-quickly-rename-a-mysql-database-change-schema-name

Chapter 8 - MySQL Table Commands

A MySQL database consists of one or more data tables. It's rare that you will find MySQL table commands in PHP scripts, one exception being installer scripts where you will execute create table commands.

How to Create a Table

SQL Command for creating a MySQL table has the following syntax. Here ENGINE and CHARSET are optional.

```
CREATE TABLE table_name (column_definitions) ENGINE CHARSET;
```

The following is an example that you can run in command-line after creating and selecting a database. You can also run this as a MySQLi query in PHP by passing it to the **mysqli_query()** function without the semicolon at the end.

```
CREATE TABLE `employee` (
  `id` NOT NULL AUTO_INCREMENT,
  `first_name` varchar(100) NOT NULL,
  `last_name` varchar(100) NOT NULL,
  `job_title` varchar(100) DEFAULT NULL,
  `salary` double DEFAULT NULL,
  `notes` text,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

- There aren't standard naming conventions for MySQL table and field names. But you will find they are usually lower case, words are separated by the underscore (Ex: **first_name**), and singular form is used (**employee** instead of **employees**).
- Even though it's optional, it's better to enclose table and field names with the **backtick** (`) character. It prevents conflicts with MySQL reserved words[37]. However, it's a good practice **not to** use MySQL reserved words for your table and field names.
- Immediately after the field name, you have to specify the data type[38] of the field. Make sure you choose the most appropriate data type for each field.

[37] http://dev.mysql.com/doc/refman/5.5/en/reserved-words.html
[38] http://dev.mysql.com/doc/refman/5.0/en/data-types.html

- After specifying the data type, you can mention field properties such as NOT NULL (always has to have a value) and AUTO_INCREMENT (MySQL will assign an incrementing value automatically). To distinguish from field names and data types, field properties are usually written in upper case.
- Almost all the tables you find will have a primary key which will be used when dealing with related tables (like fetching records from more than one table). Values entered in primary key column have to be unique, and it's possible to mention more than one column for primary key by separating them with commas. In that case, it's called a **composite primary key**, and value combinations in composite fields have to be unique.
- Indexes[39] (not used in this example) can be specified to increase query speed in cases where number of records in a table is going to be high.
- Storage engine[40] determines the functionalities of a table. For example, InnoDB allows you to have constraints among tables. Default storage engine is InnoDB[41] (You can change this by editing the MySQL configuration file). You can use other storage engines as necessary.
- Charset[42] mainly determines what types of characters and symbols you can store in table columns. Default charset is latin1. You can use a different one if necessary.

How to Retrieve the Table Definition of a Table

You can retrieve the table definition of an existing table using **SHOW CREATE TABLE** as below. It will be useful when you already know a similar table and want to use its definition as a template for a new table.

```
SHOW CREATE TABLE employee;
```

You can list field names and their properties of a table using **DESCRIBE** as below. It will be useful when you want to see table properties in a tabular format.

```
DESCRIBE employee;
```

Alternatively, you can use **EXPLAIN** or **SHOW COLUMNS FROM**.

```
EXPLAIN employee;
SHOW COLUMNS FROM employee;
```

How to List All Tables

Following is the command to list the existing tables of a database.

[39] http://dev.mysql.com/doc/refman/5.5/en/mysql-indexes.html
[40] http://dev.mysql.com/doc/refman/5.5/en/storage-engines.html
[41] http://dev.mysql.com/doc/refman/5.5/en/innodb-default-se.html
[42] http://dev.mysql.com/doc/refman/5.5/en/charset.html

```
SHOW TABLES;
```

How to Delete a Table

Use the **DROP TABLE** command as below to delete a table.

```
DROP TABLE employee;
```

How to Rename a Table

For renaming a table, you can use **RENAME TABLE** command as shown below. First, you specify the current name, then the new name.

```
RENAME TABLE `employee` TO `employee_data`;
```

How to Copy a Table

To copy both schema and data of an existing table to a new table, use the following command.

```
CREATE TABLE retired_employee SELECT * FROM employee;
```

If you only need to copy table schema, use the following command.

```
CREATE TABLE retired_employee LIKE employee;
```

How to Change Table Schema

You can use the **ALTER TABLE** command to change a table schema (structure and definitions). The following is how you can add a new column to employee table.

```
ALTER TABLE `employee` ADD `middle_name` varchar(100) DEFAULT NULL AFTER `first_n\
ame`;
```

How to Rename a Column

In the following command, we rename column `job_title` to `job`. Note that we have to specify the data type and properties of the column after the new name.

```
ALTER TABLE `employee` CHANGE `job_title` `job` varchar(100) DEFAULT NULL;
```

How to Delete a Column

You can delete a column of a table as shown below.

```
ALTER TABLE `employee` DROP COLUMN `notes`;
```

How to Delete All the Records of a Table

You can use the **TRUNCATE TABLE** command to delete all the records of a table and reset its auto-increment counter.

```
TRUNCATE TABLE employee;
```

You can also use following command to delete all the records from a table. But it won't reset auto-increment counter.

```
DELETE FROM employee;
```

How to Get Table Size

As we got the size of a database, we can query the database **information_schema** to get the size of a table. The following command will display the size of table **employee** in the database **company_db**.

```
SELECT ROUND((DATA_LENGTH+INDEX_LENGTH)/1024/1024, 2) AS "Size in MB"
from information_schema.TABLES WHERE TABLE_SCHEMA = 'company_db' AND TABLE_NAME =\
 'employee';
```

Foreign Key Constraints

Foreign key constraints[43] are helpful when you want to maintain better data integrity. If the storage engine is InnoDB, MySQL can impose some constraints between two related tables via related columns.

Assume you have two tables **employee** and **user** where **employee** table's **id** field is related to user table's **employee_id** (That is, each time you fill a row of the **user** table, you fill the corresponding **employee_id** field of the **employee** table).

Then you can impose a foreign key constraint to automatically delete the relevant row from the **user** table when the corresponding record from the **employee** table is deleted. You can define a foreign key constraint with the table definition like below.

[43] http://dev.mysql.com/doc/refman/5.5/en/innodb-foreign-key-constraints.html

```
CREATE TABLE `user` (
  `id` int NOT NULL AUTO_INCREMENT,
  `employee_id` int NOT NULL,
  `username` varchar(40) NOT NULL,
  `password` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`employee_id`) REFERENCES `employee` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

To add a foreign key constraint to an existing table, use the ALTER TABLE command as below.

```
ALTER TABLE `user` ADD CONSTRAINT FOREIGN KEY (`employee_id`) REFERENCES `employe\
e` (`id`) ON DELETE CASCADE;
```