PHP MySQL Database

With PHP, you can connect to and manipulate databases.
MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers
- Orders

PHP + MySQL Database System

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Database Queries

A query is a question or a request.
We can query a database for specific information and have a recordset returned.
Look at the following query (using standard SQL):

```
SELECT LastName FROM Employees
```

The query above selects all the data in the "LastName" column from the "Employees" table.
To learn more about SQL, please visit our SQL tutorial.

Download MySQL Database

If you don't have a PHP server with a MySQL Database, you can download it for free

here: http://www.mysql.com

Facts About MySQL Database

MySQL is the de-facto standard database system for web sites with HUGE volumes of both data and end-users (like Facebook, Twitter, and Wikipedia).
Another great thing about MySQL is that it can be scaled down to support embedded database applications.
Look at http://www.mysql.com/customers/ for an overview of companies using MySQL.

From <http://www.w3schools.com/php/php_mysql_intro.asp>

PHP Connect to MySQL

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".
Both MySQLi and PDO have their advantages:
PDO will work on 12 different database systems, where as MySQLi will only work with MySQL databases.
So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

From <http://www.w3schools.com/php/php_mysql_connect.asp>

MySQLi Installation

For Linux and Windows: The MySQLi extension is automatically installed in most cases, when php5 mysql package is installed.
For installation details, go to: http://php.net/manual/en/mysqli.installation.php

From <http://www.w3schools.com/php/php_mysql_connect.asp>

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

From <http://www.w3schools.com/php/php_mysql_connect.asp>

Example (MySQLi Procedural)

```php
<?php

$servername = "localhost";

$username = "username";

$password = "password";

// Create connection

$conn = mysqli_connect($servername, $username, $password);

// Check connection

if (!$conn) {

    die("Connection failed: " . mysqli_connect_error());

}

echo "Connected successfully";

?>
```

From <http://www.w3schools.com/php/php_mysql_connect.asp>

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

Example (MySQLi Procedural)

```php
mysqli_close($conn);
```

From <http://www.w3schools.com/php/php_mysql_connect.asp>

Create a MySQL Database Using MySQLi

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

From <http://www.w3schools.com/php/php_mysql_create.asp>

Example (MySQLi Procedural)

```php
<?php

$servername = "localhost";

$username = "username";
```

```
$password = "password";

// Create connection

$conn = mysqli_connect($servername, $username, $password);

// Check connection

if (!$conn) {

    die("Connection failed: " . mysqli_connect_error());

}

// Create database

$sql = "CREATE DATABASE myDB";

if (mysqli_query($conn, $sql)) {

    echo "Database created successfully";

} else {

    echo "Error creating database: " . mysqli_error($conn);

}

mysqli_close($conn);

?>
```

PHP Create MySQL Tables

A database table has its own unique name and consists of columns and rows.

From <http://www.w3schools.com/php/php_mysql_create_table.asp>

The CREATE TABLE statement is used to create a table in MySQL.
We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```
CREATE TABLE MyGuests (

id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,

firstname VARCHAR(30) NOT NULL,

lastname VARCHAR(30) NOT NULL,

email VARCHAR(50),
```

```
reg_date TIMESTAMP

)
```

From <http://www.w3schools.com/php/php_mysql_create_table.asp>

Notes on the table above:

The data type specifies what type of data the column can hold. For a complete reference of all the available data types, go to our Data Types reference.

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

MySQL Data Types

In MySQL there are three main types : text, number, and Date/Time types.

Text types:

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Number types:

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be

	specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

*The integer types have an extra option called UNSIGNED. Normally, the integer goes from an negative to positive value. Adding the UNSIGNED attribute will move that range up so it starts at zero instead of a negative number.

Date types:

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

Example (MySQLi Procedural)

```php
<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = mysqli_connect($servername, $username, $password, $dbname);
```

```php
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";
if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Insert Data Into MySQL Using MySQLi

After a database and a table have been created, we can start adding data in them.
Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)

To learn more about SQL, please visit our SQL tutorial.

In the previous chapter we created an empty table named "MyGuests" with five columns: "id", "firstname", "lastname", "email" and "reg_date". Now, let us fill the table with data.

Note: If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

From <http://www.w3schools.com/php/php_mysql_insert.asp>

The following examples add a new record to the "MyGuests" table:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Get ID of The Last Inserted Record

If we perform an INSERT or UPDATE on a table with an AUTO_INCREMENT field, we can get the ID of the last inserted/updated record immediately.
In the table "MyGuests", the "id" column is an AUTO_INCREMENT field:

```
CREATE TABLE MyGuests (

id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,

firstname VARCHAR(30) NOT NULL,

lastname VARCHAR(30) NOT NULL,

email VARCHAR(50),

reg_date TIMESTAMP

)
```

The following examples are equal to the examples from the previous page (PHP Insert Data Into MySQL), except that we have added one single line of code to retrieve the ID of the last inserted record. We also echo the last inserted ID:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
   $last_id = mysqli_insert_id($conn);
  echo "New record created successfully. Last inserted ID is: " . $last_id;
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Insert Multiple Records Into MySQL

Multiple SQL statements must be executed with the mysqli_multi_query() function.

The following examples add three new records to the "MyGuests" table:

From <http://www.w3schools.com/php/php_mysql_insert_multiple.asp>

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
```

```
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if (mysqli_multi_query($conn, $sql)) {
  echo "New records created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

From <http://www.w3schools.com/php/php_mysql_insert_multiple.asp>

Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:
SELECT column_name(s) FROM table_name
or we can use the * character to select ALL columns from a table:
SELECT * FROM table_name
To learn more about SQL, please visit our SQL tutorial.

Select Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

From <http://www.w3schools.com/php/php_mysql_select.asp>

Example (MySQLi Procedural)

```
<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection

if (!$conn) {
```

```
    die("Connection failed: " . mysqli_connect_error());

}

$sql = "SELECT id, firstname, lastname FROM MyGuests";

$result = mysqli_query($conn, $sql);


if (mysqli_num_rows($result) > 0) {

    // output data of each row

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

        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
$row["lastname"]. "<br>";

    }

} else {

    echo "0 results";

}

mysqli_close($conn);

?>
```

Run Example

http://www.w3schools.com/php/showphpfile.asp?filename=demo_db_select_proc

---

Delete Data From a MySQL Table

The DELETE statement is used to delete records from a table:

DELETE FROM table_name
WHERE some_column = some_value

Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

The following examples delete the record with id=3 in the "MyGuests" table:

From <http://www.w3schools.com/php/php_mysql_delete.asp>

Example (MySQLi Procedural)

```php
<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection

if (!$conn) {

    die("Connection failed: " . mysqli_connect_error());

}

// sql to delete a record

$sql = "DELETE FROM MyGuests WHERE id=3";


if (mysqli_query($conn, $sql)) {

    echo "Record deleted successfully";

} else {

    echo "Error deleting record: " . mysqli_error($conn);

}

mysqli_close($conn);

?>
```

After the record is deleted, the table will look like this:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

Update Data In a MySQL Table Using MySQLi and PDO

The UPDATE statement is used to update existing records in a table:

UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

From <http://www.w3schools.com/php/php_mysql_update.asp>

To learn more about SQL, please visit our SQL tutorial.
Let's look at the "MyGuests" table:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

The following examples update the record with id=2 in the "MyGuests" table:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

After the record is updated, the table will look like this:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Doe	mary@example.com	2014-10-23 10:22:30

Limit Data Selections From a MySQL Database

MySQL provides a LIMIT clause that is used to specify the number of records to return.
The LIMIT clause makes it easy to code multi page results or pagination with SQL, and is very useful on large tables. Returning a large number of records can impact on performance.
Assume we wish to select all records from 1 - 30 (inclusive) from a table called "Orders". The SQL query would then look like this:

$sql = "SELECT * FROM Orders LIMIT 30";

When the SQL query above is run, it will return the first 30 records.

What if we want to select records 16 - 25 (inclusive)?
Mysql also provides a way to handle this: by using **OFFSET.**

The SQL query below says "return only 10 records, start on record 16 (OFFSET 15)":
$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";

You could also use a shorter syntax to achieve the same result:

$sql = "SELECT * FROM Orders LIMIT 15, 10";

Notice that the numbers are reversed when you use a comma.

From <http://www.w3schools.com/php/php_mysql_select_limit.asp>