Instructor: Sudad H. Abed / Email: shabed@uoanbar.edu.iq

Lecture Reference: www.w3schools.com ,
PHP and MySQL for Dynamic Web Sites, 4th Edition

Chapter 4: Introduction to MySQL

What is MySQL?

- MySQL is the most popular database system used with PHP.
- 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 the de-facto standard database system for web sites with HUGE volumes of both data and end-users (like Facebook, Twitter, and Wikipedia).

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

  - Employees
  - Products
  - Customers
  - Orders

Specific Text Types

- CHAR
- VARCHAR
- TINYTEXT
- TEXT
- MEDIUMTEXT
- LONGTEXT

Specific Numeric Types

- TINYINT
- SMALLINT
- MEDIUMINT
- INT
- BIGINT
- FLOAT
- DOUBLE
- DECIMAL

Specific Date and Time Types

- DATE
- DATETIME
- TIMESTAMP
- TIME

Instructor: Sudad H. Abed / Email: shabed@uoanbar.edu.iq

Lecture Reference: www.w3schools.com ,
PHP and MySQL for Dynamic Web Sites, 4th Edition

CHAR

- Fixed length
- Generally, requires more disk space
- Generally faster
- Best used for values that will always be a fixed length

VARCHAR

- Variable length
- Generally, requires less disk space
- Generally slower
- Best used for values that will be of any length

Important Column Properties

- **Length**
- **NULL/NOT NULL** → Best to use NOT NULL as much as possible.
- **DEFAULT** →The DEFAULT *value* clause in a data type specification indicates a default value for a column.
- **UNSIGNED** → Best to use UNSIGNED whenever appropriate for numbers.
- **ZEROFILL → declared as INT(4) ZEROFILL, a value of 5 is retrieved as 0005.**
- **AUTO_INCREMENT**→ AUTO_INCREMENT is used with primary key columns.

Primary Key

- Must always have a value.
- The value must never change.
- The value must be unique for each record in the table.
- Almost always unsigned, not null integers.
- Use AUTO_INCREMENT to set the value.

To Create a Storage in Database

1- Determine the database's name.
2. Determine the table names.
3. Determine the column names for each table.
4. Determine types for columns.
5. Define Index, Restrictions etc.

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.

➢ Open a Connection to MySQL

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

PHP Create a MySQL Database

A database consists of one or more tables.

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

PHP Create MySQL Tables

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

- We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

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

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->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 ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

PHP Insert Data Into MySQL

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,...)
```

Ex:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```

PHP Get ID of 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:

Ex:

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

if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "New record created successfully. Last inserted ID is: " .
$last_id;
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
```

PHP 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:

```
$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 ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
```

PHP Prepared Statements

Prepared statements are very useful against SQL injections.

Prepared statements basically work like this:

1. Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO MyGuests VALUES(?, ?, ?).
2. The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it.
3. Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values.

Ex:

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

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
```

```
}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
?>
```

- The argument may be one of four types:

- i - integer
- d - double
- s - string
- b - BLOB

- We must have one of these for each parameter.
  By telling mysql what type of data to expect, we minimize the risk of SQL injections.