PHP MySQL INSERT Query

Inserting Data into a MySQL Database Table

The `INSERT INTO` statement is used to insert new rows in a database table.

Let's make a SQL query using the `INSERT INTO` statement with appropriate values, after that we will execute this insert query through passing it to the PHP `mysqli_query()` function to insert data in table. Here's an example, which insert a new row to the *persons* table by specifying values for the *first_name*, *last_name* and *email* fields.

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are
running MySQL
server with default setting (user 'root' with no
password) */
$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " .
mysqli_connect_error());
}

// Attempt insert query execution
$sql = "INSERT INTO persons (first_name, last_name,
email) VALUES ('Peter', 'Parker',
'peterparker@mail.com')";
if(mysqli_query($link, $sql)){
    echo "Records inserted successfully.";
} else{
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

Inserting Multiple Rows into a Table

You can also insert multiple rows into a table with a single insert query at once. To do this, include multiple lists of column values within the `INSERT INTO` statement, where column values for each row must be enclosed within parentheses and separated by a comma.

Let's insert few more rows into the *persons* table, like this:

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are
running MySQL
server with default setting (user 'root' with no
password) */
$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " .
mysqli_connect_error());
}

// Attempt insert query execution
$sql = "INSERT INTO persons (first_name, last_name,
email) VALUES
            ('John', 'Rambo', 'johnrambo@mail.com'),
            ('Clark', 'Kent', 'clarkkent@mail.com'),
            ('John', 'Carter', 'johncarter@mail.com'),
            ('Harry', 'Potter', 'harrypotter@mail.com')";
if(mysqli_query($link, $sql)){
    echo "Records added successfully.";
} else{
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

Now, go to phpMyAdmin (`http://localhost/phpmyadmin/`) and check out the *persons* table data inside *demo* database. You will find the value for the *id* column is assigned automatically by incrementing the value of previous *id* by 1.

Note: Any number of line breaks may occur within a SQL statement, provided that any line break does not break off keywords, values, expression, etc.

Insert Data into a Database from an HTML Form

In the previous section, we have learned how to insert data into database from a PHP script. Now, we'll see how we can insert data into database obtained from an HTML form. Let's create an HTML form that can be used to insert new records to *persons* table.

Step 1: Creating the HTML Form

Here's a simple HTML form that has three text `<input>` fields and a submit button.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Add Record Form</title>
</head>
<body>
<form action="insert.php" method="post">
    <p>
        <label for="firstName">First Name:</label>
        <input type="text" name="first_name"
id="firstName">
    </p>
    <p>
        <label for="lastName">Last Name:</label>
        <input type="text" name="last_name"
id="lastName">
    </p>
    <p>
        <label for="emailAddress">Email Address:</label>
        <input type="text" name="email"
id="emailAddress">
    </p>
```

```
    <input type="submit" value="Submit">
</form>
</body>
</html>
```

Step 2: Retrieving and Inserting the Form Data

When a user clicks the submit button of the add record HTML form, in the example above, the form data is sent to 'insert.php' file. The 'insert.php' file connects to the MySQL database server, retrieves forms fields using the PHP `$_REQUEST` variables and finally execute the insert query to add the records. Here is the complete code of our 'insert.php' file:

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are
running MySQL
server with default setting (user 'root' with no
password) */
$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " .
mysqli_connect_error());
}

// Escape user inputs for security
$first_name = mysqli_real_escape_string($link,
$_REQUEST['first_name']);
$last_name = mysqli_real_escape_string($link,
$_REQUEST['last_name']);
$email = mysqli_real_escape_string($link,
$_REQUEST['email']);

// Attempt insert query execution
$sql = "INSERT INTO persons (first_name, last_name,
email) VALUES ('$first_name', '$last_name', '$email')";
if(mysqli_query($link, $sql)){
    echo "Records added successfully.";
} else{
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}

// Close connection
mysqli_close($link);
```

```
?>
```

In the next chapter we will extend this insert query example and take it one step further by implementing the prepared statement for better security and performance.

Note: The `mysqli_real_escape_string()` function escapes special characters in a string and create a legal SQL string to provide security against SQL injection.

This is very basic example of inserting the form data in a MySQL database table. You can extend this example and make it more interactive by adding validations to the user inputs before inserting it to the database tables.

PHP MySQL SELECT Query

Selecting Data From Database Tables

So far you have learnt how to create database and table as well as inserting data. Now it's time to retrieve data what have inserted in the preceding tutorial. The SQL `SELECT` statement is used to select the records from database tables. Its basic syntax is as follows:

```
SELECT column1_name, column2_name, columnN_name FROM table_name;
```

Let's make a SQL query using the `SELECT` statement, after that we will execute this SQL query through passing it to the PHP `mysqli_query()` function to retrieve the table data.

Consider our *persons* database table has the following records:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
```

```
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

The PHP code in the following example selects all the data stored in the *persons* table (using the asterisk character (`*`) in place of column name selects all the data in the table).

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are
running MySQL
server with default setting (user 'root' with no
password) */
$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " .
mysqli_connect_error());
}

// Attempt select query execution
$sql = "SELECT * FROM persons";
if($result = mysqli_query($link, $sql)){
    if(mysqli_num_rows($result) > 0){
        echo "<table>";
            echo "<tr>";
                echo "<th>id</th>";
                echo "<th>first_name</th>";
                echo "<th>last_name</th>";
                echo "<th>email</th>";
            echo "</tr>";
        while($row = mysqli_fetch_array($result)){
            echo "<tr>";
                echo "<td>" . $row['id'] . "</td>";
                echo "<td>" . $row['first_name'] .
"</td>";
                echo "<td>" . $row['last_name'] .
"</td>";
                echo "<td>" . $row['email'] . "</td>";
            echo "</tr>";
        }
        echo "</table>";
        // Free result set
        mysqli_free_result($result);
```

```
    } else{
        echo "No records matching your query were
found.";
    }
} else{
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

Explanation of Code (Procedural style)

In the example above, the data returned by the `mysqli_query()` function is stored in the `$result` variable. Each time `mysqli_fetch_array()` is invoked, it returns the next row from the result set as an array. The while loop is used to loops through all the rows in the result set. Finally the value of individual field can be accessed from the row either by passing the field index or field name to the `$row` variable like `$row['id']` or `$row[0]`, `$row['first_name']` or `$row[1]`, `$row['last_name']` or `$row[2]`, and `$row['email']` or `$row[3]`.

If you want to use the `for` loop you can obtain the loop counter value or the number of rows returned by the query by passing the `$result` variable to the `mysqli_num_rows()` function. This loop counter value determines how many times the loop should run.

PHP MySQL UPDATE Query

Updating Database Table Data

The `UPDATE` statement is used to change or modify the existing records in a database table. This statement is typically used in conjugation with the `WHERE` clause to apply the changes to only those records that matches specific criteria.

The basic syntax of the `UPDATE` statement can be given with:

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

Let's make a SQL query using the `UPDATE` statement and `WHERE` clause, after that we will execute this query through passing it to the PHP `mysqli_query()` function to update the tables records. Consider the following *persons* table inside the *demo* database:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

The PHP code in the following example will update the email address of a person in the persons table whose *id* is equal to 1.

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are
running MySQL
server with default setting (user 'root' with no
password) */
$link = mysqli_connect("localhost", "root", "", "demo");
```

```php
// Check connection
if($link === false){
    die("ERROR: Could not connect. " .
mysqli_connect_error());
}

// Attempt update query execution
$sql = "UPDATE persons SET
email='peterparker_new@mail.com' WHERE id=1";
if(mysqli_query($link, $sql)){
    echo "Records were updated successfully.";
} else {
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

After update the persons table will look something like this:

```
+----+------------+-----------+--------------------------+
| id | first_name | last_name | email                    |
+----+------------+-----------+--------------------------+
|  1 | Peter      | Parker    | peterparker_new@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com       |
|  3 | Clark      | Kent      | clarkkent@mail.com       |
|  4 | John       | Carter    | johncarter@mail.com      |
|  5 | Harry      | Potter    | harrypotter@mail.com     |
+----+------------+-----------+--------------------------+
```

Warning: The WHERE clause in the UPDATE statement specifies which record or records should be updated. If you omit the WHERE clause, all records will be updated.

PHP MySQL DELETE Query

Deleting Database Table Data

Just as you insert records into tables, you can delete records from a table using the SQL `DELETE` statement. It is typically used in conjugation with the `WHERE` clause to delete only those records that matches specific criteria or condition.

The basic syntax of the `DELETE` statement can be given with:

```
DELETE FROM table_name WHERE column_name=some_value
```

Let's make a SQL query using the `DELETE` statement and `WHERE` clause, after that we will execute this query through passing it to the PHP `mysqli_query()` function to delete the tables records. Consider the following *persons* table inside the *demo* database:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  2 | John       | Rambo     | johnrambo@mail.com   |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  4 | John       | Carter    | johncarter@mail.com  |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

The PHP code in the following example will delete the records of those persons from the *persons* table whose *first_name* is equal to John.

Example

```
<?php
/* Attempt MySQL server connection. Assuming you are
running MySQL
server with default setting (user 'root' with no
password) */
$link = mysqli_connect("localhost", "root", "", "demo");
```

```php
// Check connection
if($link === false){
    die("ERROR: Could not connect. " .
mysqli_connect_error());
}

// Attempt delete query execution
$sql = "DELETE FROM persons WHERE first_name='John'";
if(mysqli_query($link, $sql)){
    echo "Records were deleted successfully.";
} else{
    echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

After the deletion the *persons* table will look something like this:

```
+----+------------+-----------+----------------------+
| id | first_name | last_name | email                |
+----+------------+-----------+----------------------+
|  1 | Peter      | Parker    | peterparker@mail.com |
|  3 | Clark      | Kent      | clarkkent@mail.com   |
|  5 | Harry      | Potter    | harrypotter@mail.com |
+----+------------+-----------+----------------------+
```

As you can see the records has been deleted successfully from the persons table.

Warning: The `WHERE` clause in the `DELETE` statement specifies which record or records should be deleted. If you omit the `WHERE` clause, all records will be deleted.