The Three Most Important MySQL Queries

(2009-07-06) - Contributed by Codex-M

Doing PHP queries to a MySQL database is one of the most important processes in any web application. This is where data is being fetched from or inserted into the database. If you are a beginning PHP web developer, then learning the most important MySQL queries is essential to your success in dealing with dynamic websites.

You can see dynamic websites almost everywhere on the net. Popular software packages, whether for blogging (like WordPress or Drupal) or e-commerce (such as osCommerce) are configured to be dynamic in nature. Dynamic websites rely on MySQL databases to fetch and insert data coming from and going to the browser.

If you are beginner, understanding the simple diagram below helps you to understand the role of MySQL in today's dynamic websites as compared to the traditional static websites.

In a simple static websites (popular in the old days of the Internet), you would see the following flow of events:

Client browser request for information from the server -> Server returns HTML code to the browser -> Browser displays the HTML, which becomes a web page.

Now, in modern dynamic websites:

The client browser requests a web page, and when the server receives the request, the web template (those files with the extension *.php) executes the PHP scripts that involve fetching information (involving queries) from MySQL (such as your name, password, content, or anything else). The information will then be added to the web template, and then return to the browser as HTML. When you view the source code, you cannot see the PHP scripts any longer, because they have already been executed in the server.

With dynamic websites, you can create a thousand URLs with only one server PHP template stored. But with static websites, you need to store 1000 HTML files in your server to create 1000 URLs (which is very difficult to manage or update). Do you see the difference?

If you know the most important MySQL queries, it will then be very easy to communicate with the database, thus enabling you to create complex PHP web applications. This will save you time and make your website as efficient as possible. In this tutorial, we will illustrate PHP scripts that will query a MySQL database using the basic INSERT, SELECT, and UPDATE commands.

{mospagebreak title=The INSERT MySQL Query Command}

Okay, before we can grab data from the database, we must know how to insert data. Assuming you have already created a database with a table name, then inserting data can be done with the INSERT SQL command. I've given an example below. Assuming that you are inserting three pieces of data simultaneously into three different fields in the MySQL database, it will look something like this:

```
INSERT INTO `tablename` (`fieldname1`,`fieldname2`,``,`fieldname3`)
VALUES('$datatobeinsertedtofieldname1','$datatobeinsertedtofieldname2',
'$datatobeinsertedtofieldname3')
```

When using this query with PHP, be careful about the exact use of quote symbols and exact spelling of fieldnames and variables. This is the common cause of errors in PHP programming that involve the use of the INSERT query.

Note that the table name and field names in the query command should be used with back ticks (`) , and that you need to maintain consistent cases between the actual MySQL tablenames and the ones used with your script. This will avoid

case sensitivity issues, which could be common in Linux/PHP and MySQL environment.

For example: if your table name is SongArchives, use also:

INSERT INTO `SongArchives` .....

Finally, to formulate a complete MySQL query in PHP using the INSERT command, the following is the recommended script:

```
<?php

//Step 1 Connect to database

$username = "Your MySQL username here";

$password = "Your MySQL password";

$hostname = "Hostname";

$table = "MySQL Table name where the data will be inserted";

$database = "The name of the MySQL database which holds the table";

$dbhandle = mysql_connect($hostname, $username, $password)

or die("Unable to connect to MySQL");

$selected = mysql_select_db($database,$dbhandle)

or die("Could not select $database");

//Step 2. Insert other PHP scripts here (such as grabbing data from HTML forms, etc)

//Step 3. Sanitize variables before inserting to database. This will prevent MySQL injection.

$datatobeinsertedtofieldname1 = mysql_real_escape_string(stripslashes($datatobeinsertedtofieldname1));

$datatobeinsertedtofieldname2 = mysql_real_escape_string(stripslashes($datatobeinsertedtofieldname2));

$datatobeinsertedtofieldname3 = mysql_real_escape_string(stripslashes($datatobeinsertedtofieldname3));

mysql_query("INSERT INTO `tablename` (`fieldname1`,`fieldname2`,``,`fieldname3`)
VALUES('$datatobeinsertedtofieldname1','$datatobeinsertedtofieldname2',
'$datatobeinsertedtofieldname3')")
```

```
or die(mysql_error());

?>
```

{mospagebreak title=The SELECT MySQL Query Command}

So we now have data in our database. If we would like to grab data from it, we will need the SELECT command. There are several options for grabbing data.

Option 1: Grabbing the entire data table out of the database:

```
SELECT * FROM `tablename`
```

Here is the PHP script to illustrate this example. It grabs the entire MySQL data table and displays it in your browser.

```
<?php

//Step 1 Connect to database

$username = "Your MySQL username here";

$password = "Your MySQL password";

$hostname = "Hostname";

$table = "MySQL Table name where the data will be inserted";

$database = "The name of the MySQL database which holds the table";

$dbhandle = mysql_connect($hostname, $username, $password)

or die("Unable to connect to MySQL");

$selected = mysql_select_db($database,$dbhandle)

or die("Could not select $database");

//Step 2. Insert other PHP scripts here (such as grabbing data from HTML forms, etc)

//Step 3. Sanitize variables before inserting to database. This will prevent MySQL injection.

$categories_id = mysql_real_escape_string(stripslashes($categories_id));
```

```
$language_id = mysql_real_escape_string(stripslashes($language_id));

$categories_name = mysql_real_escape_string(stripslashes($categories_name));

$result = mysql_query("SELECT * FROM `categories_description`")

or die(mysql_error());

echo '<table border="1">';

echo '<tr>';

echo '<td>Categories ID</td>';

echo '<td>Language ID</td>';

echo '<td>Categories Name</td>';

echo '</tr>';

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

echo '<tr>';

echo '<td>'.$row['categories_id'].'</td>';

echo '<td>'.$row['language_id'].'</td>';

echo '<td>'.$row['categories_name'].'</td>';

echo '</tr>';

}

echo '</table>';

?>
```

This will display the table in the HTML browser, which looks like this:

Option 2: Extracting specific data from a specific fieldname in the MySQL table.

There are cases when you need to extract specific data from a specific fieldname that satisfies certain conditions.

An example of this application is when you need to extract the customer email address if they can correctly provide the username and password. So there could be three fields in the table: username, password, and emailaddress.

So, to extract the customer email address in a MySQL table, for example customertable, the PHP query could be:

```
<?php

//Connect to database...

//Sanitize the variables..

//MySQL query

$result = mysql_query("SELECT `emailaddress` FROM `customertable` WHERE `username`='$username' AND
`password`='$password'")

or die(mysql_error());

$row = mysql_fetch_array($result)

or die("Invalid query: " . mysql_error());

$emailaddress = $row['emailaddress'];

echo "Your email address is: $emailaddress";

?>
```

In this PHP MySQL query, the WHERE statement specifies the condition under which it pulls out an email address matching a certain username AND a password.

You can query MySQL using several AND statements, as long as it is correctly done.

{mospagebreak title=The UPDATE MySQL Query Command}

One of the important MySQL queries that need to be done in PHP is UPDATE. As the command suggests, it enables you to update records in the MySQL table, without necessarily inserting a row again.

For example, you have the following data table below:

Assuming Peter Patter would like to change his password from "logan" to "weaponx," the MySQL query using UPDATE would be:

```
<?php

//Step 1 Connect to database

$username = "yourusername";

$password = "yoursqlpassword";

$hostname = "localhost";

$table = "usertable";

$database = "userdatabase";

$dbhandle = mysql_connect($hostname, $username, $password)

or die("Unable to connect to MySQL");

$selected = mysql_select_db($database,$dbhandle)

or die("Could not select $database");

//Initial variables

$username = 'wolverine';

$fullname = 'Peter Patter';

//Set new password for Peter Patter

$newpassword = 'weaponx';

//sanitize

$newpassword = mysql_real_escape_string(stripslashes($newpassword));

$username = mysql_real_escape_string(stripslashes($username));

$fullname = mysql_real_escape_string(stripslashes($fullname));

//Update records in MySQL database

mysql_query("UPDATE usertable SET password = '$newpassword' WHERE username = '$username' AND fullname =
'$fullname'")
```

```
or die(mysql_error());

echo 'The database has been updated. Thank you';

?>
```

The update script above will change the password of Peter Patter from "logan" to "weaponx." The script above is just a raw example; you can make the above PHP script work with HTML forms for more interactive updating of data.