4

PHP and MySQL Programming

This chapter covers how to create databases and tables in MySQL via PHP scripts. Instead of maintaining the information in the data tables through the MySQL prompt, we will learn doing it through PHP programming. Maintenance of database tables through the MySQL prompt is cumbersome and also requires the user to be well versed with SQL commands. To make it easier for visitors to our website, we will provide them an easy-to-use GUI form (made in PHP) and the information entered in the form will be automatically stored in the database tables. Similarly, to fetch some information from the tables, the visitor will be asked to fill an enquiry form and the desired information will be retrieved from the database and displayed. The information of the database name, table name and different MySQL commands used to extract information from the database will not be visible to the visitor, making the web application more secure. In this chapter we will make and understand small PHP modules that perform the following tasks:

- Creating a table in PHP
- Getting information from the user
- Inserting records in the table
- Retrieving all records from the table
- Searching records from the table (retrieving only the desired records, not all)
- Deleting records from the table

Before we begin with these tasks, let us first look at all the statements and functions that we will require for accessing MySQL from PHP.

4.1 FUNCTIONS ACCESSING MYSQL FROM PHP

The following are the functions that we will be using for selecting databases, accessing tables, executing SQL statements and fetching results. Let us look at them one by one:

4.1.1 MySQL_Connect()

This function is used for establishing the connection to the MySQL database. It returns `true` if the connection is established else `false` is returned. This function takes three parameters; the first one is hostname, then user-id and then password.

Syntax:

```
mysql_connect ("$servername","$dbuser","$dbpassword");
```

Example:

```
$connect = mysql_connect("localhost", "root", "mce");
if(!$connect) { die("Please, check your server connection."); }
```

`Die` is the command to exit. If the `mysql_connect()` returns false then we exit from the PHP scripts using `die` command after displaying the message: "Please, check your server connection".

This example can also be written as:

```
$connect = mysql_connect("localhost", "root", "mce") or die("Please, check your server connection.");
```

When two statements are connected with `or` operator, then the second statement is only executed if the first statement returns `false`.

4.1.2 Mysql_Select_db()

This function is used for selecting a MySQL database to be operated on.

Syntax:

```
bool mysql_select_db (string $database_name [, resource $link_identifier])
```

This selects the active database on the server with which connection is established by using a link identifier. If the link identifier is not specified, the last link opened by `mysql_connect()` is assumed. If no such link is found, it will create one by executing the MySQL with no arguments. An error is displayed if a connection is not established. It returns `true` if the database is found in the MySQL server else `false` is returned.

Examples:

- `mysql_select_db("shopping",$connect);`
- `mysql_select_db("shopping");`

4.1.3 Mysql_Query()

This function sends the query to the currently active database on the server and the records fetched are stored in an array.

Syntax:

```
resource mysql_query (string $query [, resource $link_identifier])
```

where `link_identifier` identifies the connection established with MySQL. If the link identifier is not specified, the last link opened by `mysql_connect()` is assumed. If no such link is found, it tries to create one by executing `mysql_connect()` with no arguments. An error is displayed if a connection is not established.

The returned result resource may be passed to `mysql_fetch_array()`, to display fetched records one by one.

Example:

```
$result = mysql_query($query) or die(mysql_error());
```

The `$result` returned by `mysql_query()` contains all the records fetched from the table on the basis of the specified query and by passing this array of records `$result` to `mysql_fetch_array()`, we can extract one record at a time from it.

4.1.4 $_POST()

The `$_POST` array is an array which collects the values sent from a form using HTTP POST method. When the user clicks the "`Submit`" button, the value of the `$_POST["variable_name"]` will be automatically set by PHP where `variable_name` is the id of the input HTML tag used on the form. The destination PHP script can use the `$_POST` array to retrieve the form data with the help of the following code:

```
local_variable=$_POST["variable_name"];
```

Hence `local_variable` will contain the information passed by another web form.

4.1.5 mysql_fetch_array()

This function is used for fetching one record at a time from the array of records (produced as a result of a query). It gets one row from the recordset and returns `true`, and `false` when there are no more rows left in the recordset. It returns a row from a recordset as an associative array or a numeric array.

Syntax:

```
row=mysql_fetch_array(data,array_type)
```

where the data pointer is the result from the `mysql_query()` function and

array type is optional that specifies what kind of array to return. Its value can be:

MYSQL_ASSOC -	Associative array
MYSQL_NUM -	Numeric array
MYSQL_BOTH -	Default. Both associative and numeric array

Note: After the data is retrieved, this function moves to the next row in the recordset. Each subsequent call to `mysql_fetch_array()` returns the next row in the recordset.

Example:

```
$row = mysql_fetch_array($results)
```

One record is extracted from the recordset `$results` and is stored in variable: `$row`

4.1.6 Extract()

This function is used for extracting all the variables stored in the specified array (or record).

Syntax:

```
extract(array name)
```

Example:

```
extract($row);
```

If `$row` contains a record (row) of a table, say `customers`, with two fields say `userid` and `address`, then by the above `extract()` function, two variables `$userid` and `$address` will be generated that will contain the information of `userid` and `address` in that particular row (record).

4.1.7 mysql_close()

This is a very important command as it closes the connection to the database server. It is a better idea to close the connection with the server when all the operations are performed on the database or else it may diminish the efficiency of the web host.

4.1.8 mysql_numrows()

To know how many rows there are in the resultset, we use this function. Recall that we get the rows extracted from the table and stored in the form of resultset when an SQL query consisting of "`select`" clause is executed. Thus this function counts how many rows there are in the given resultset.

Syntax:

```
mysql_numrows(resultset name);
```

Example:

```
$num=mysql_numrows($result);
```

This will set the value of `$num` to be the number of rows stored in `$result` (the array where the records retrieved from the table are stored on execution of the SQL statement with select clause).

4.2 CREATING A TABLE IN PHP

Creating a table in a MySQL database from PHP script requires the following steps:

1. Establishing connection with MySQL server
2. Creating a database (if it does not exist). If database already exists then skip this step
3. Selecting the database (in which table is to be created)
4. Writing the SQL statement for creating a table
5. Executing the SQL statement

To understand the above steps let us look at the following program that creates a table named `products` in the `shopping` database. The products table that we are going to create will have five fields (columns): `id`, `item_code`, `item_name`, `quantity` and `price`. The fields `id`, `item_code` and `quantity` are considered to be of `int` (integer) data type, `item_name` of `character` data type and `price` of `float` data type.

createtable_products.php

```
<?php
$user="username";
$password="password";
$connect = mysql_connect("localhost", $user, $password) or die ("Please
check your server connection.");

//create the database if it doesn't already exist
$create = mysql_query("CREATE DATABASE IF NOT EXISTS shopping") or
die(mysql_error());
mysql_select_db("shopping");

$sql = "CREATE TABLE products(
   id int(6) NOT NULL auto_increment,
   item_code int(4),
   item_name varchar(50),
```

```
        quantity int(4),
        price float)";
        $res = mysql_query($sql) or die (mysql_error());
    echo "products table successfully created!";
?>
```

The above program has all the five steps of creating a table. In the first two lines, two variables `$user` and `$password` are initialised to the username and password of the MySQL server respectively. From the third line onwards, we see that the five steps for creating a table are executed. Let us see look at each statement step by step:

1. Establishing connection with MySQL server

Before proceeding with the creation of a database or a table, first we need to connect to the MySQL server. The connection is established with the help of a valid user name and password.

```
$connect = mysql_connect("localhost", $user, $password) or die ("Please check
your server connection.");
```

In the above command `localhost` signifies that the MySQL server is installed on the local machine. The string `localhost` will be replaced by the IP address of the server or server name (Example: sqlserver.com or 216.237.120.79) if we try to connect to the remote server. The `$user` and `$password` variables contain the valid userid and password supplied by the administrator. The keyword `die` is for displaying error messages if any information supplied is wrong.

2. Creating a database

After we are connected to the MySQL server, we must then create the database (if it does not exist) followed by selecting it i.e. making it active in memory. The command for creating database is:

```
$create = mysql_query("CREATE DATABASE IF NOT EXISTS shopping") or
die(mysql_error());
```

The above statement creates a database named `"shopping"` if it doesn't already exist. If the database is already there, this command won't do anything.

3. Selecting the database

Selecting the database means loading the database in memory (i.e. making it active). The command for selecting a given database is:

```
mysql_select_db("shopping");
```

The above command selects the database `shopping`. Now all SQL statements that are executed will be applied to the tables belonging to this (`shopping`) active database.

4. Writing the SQL statement for creating the table

The SQL statement given below creates a table named `products` that has five fields out of which `id` field is set to `auto_increment` i.e. its value will automatically increase by 1 with every record inserted.

```
$sql = "CREATE TABLE products(
id int(6) NOT NULL auto_increment,
item_code int(4),
item_name varchar(50),
quantity int(4),
price float)";
```

The rest of the fields (columns) in the table are `item_code` that will store an integer values of up to four digits, `item_name` field will store the name of the item of up to 50 characters long and `quantity` field can store an integer value of up to four digits and `price` field can store a fractional value (value with decimal points).

5. Executing the SQL statement

The statement given below executes the above SQL statement of creating the products table

```
$res = mysql_query($sql) or die (mysql_error());
```

The `$res` variable is for storing the result of the execution of the SQL statement.

4.3 GETTING INFORMATION FROM THE USER

In every website, getting information from the visitor is an essential requirement. Either in the form of a Sign Up form, Order form or a Feedback form, we gather information from the visitor. Getting information from the visitor requires the following steps:

1. Providing the text boxes, radio buttons, checkboxes etc. to the visitor to enter information
2. Submitting the information for some action. The action may be either storing the information in the tables, fetching information from the tables or updating information in the tables

Consider the following program, where we provide two textboxes to the visitor to fill in `userid` and `address`. After filling up the information, the visitor is supposed to select `submit` button to proceed with necessary action.

enterinfo.php

```
<html>
  <head>
```

```
    </head>
    <body>
        <form action="insertrec.php" method="post">
            <table border="0" cellspacing="1" cellpadding="3">
                <tr><td>Userid: </td><td> <input size="20" type="text"
                    name="userid"></td></tr>
                <tr><td>Address: </td><td> <input size="80" type="text"
                    name="add"></td></tr>
                <tr><td><input type="submit" name="submit" value="Submit">
                    </td><td><input type="reset" value="Cancel"></td></tr>
            </table>
        </form>
    </body>
</html>
```

Explanation

`<form action="insertrec.php" method="post">`

The above statement means that when the `Submit` button is pressed, we will be navigated to the file: `insertrec.php` and the HTTP Request method that is used for passing the information entered in the current file is `Post` method.

In other words, when the user clicks the "Submit" button, the `$_POST["userid"]` and `$_POST["add"]` variables will be automatically initialised to the data entered by the user. The information in these variables can be retrieved in the `insertrec.php` file to physically store it in the table.

`<table border="0" cellspacing="1" cellpadding="3">`

This statement defines a table with the specified cell spacing and cell padding. Tables are usually used for the purpose of aligning labels and textboxes on the web form.

```
<tr><td>Userid: </td><td> <input size="20" type="text"
            name="userid"></td></tr>
<tr><td>Address: </td><td> <input size="80" type="text"
            name="add"></td></tr>
```

The user is provided with two textboxes to enter the information (we can use other controls like checkboxes, radio buttons, drop down lists etc. to get information from the user). The information entered by the user will be automatically stored in `$_POST["userid"]` and `$_POST["add"]` variables.

```
<tr><td><input type="submit" name="submit" value="Submit">
```

This will navigate us to another PHP web form: `insertrec.php` file.

The conclusion is that the information filled up by the user is stored in `$_POST["userid"]` and `$_POST["add"]` variables and is sent to `insertrec.php` file for further action.

4.4 INSERTING RECORDS IN THE TABLE

The information sent by the user via enterinfo.php program: userid and add (for address) is passed to insertrec.php program (as declared by the statement: <form action="insertrec.php" method="post">). In the insertrec.php program we will be storing the received information in a table. Insertion of records in a table through PHP program requires the following steps:

1. Establishing connection with the MySQL server
2. Selecting the database containing the table in which record is supposed to be inserted
3. Collecting information to be stored in the table
4. Writing the SQL statement for inserting the record
5. Executing the SQL statement

Let us observe the following PHP program where a step-by-step approach is taken for inserting the information (`userid` and `add`) passed from the previous PHP program (`enterinfo.php`) into a table named `customers`. The program is as shown below:

insertrec.php

```
<?php
    connect = mysql_connect("localhost", "root", "mce") or die ("Please,
        check the server connection.");
    mysql_select_db("shopping");
    userid = $_POST['userid'];
    add = $_POST['add'];

    $sql = "INSERT INTO customers (userid, address) VALUES ('$userid',
    '$add')";
    $result = mysql_query($sql) or die(mysql_error());
    echo "Record is saved";
?>
```

Explanation

The five steps of inserting records can be seen:

1. Establishing connection with MySQL server

```
connect = mysql_connect("localhost", "root", "mce") or die ("Please,
    check the server connection.");
```

The above statement establishes the connection with the local MySQL server with the `userid` as `"root"` and the `password` of the root is assumed to be `"mce"` (any text).

2. Selecting the database

```
mysql_select_db("shopping");
```

This statement selects the database "`shopping`" (i.e. loads it in memory) so that we can insert record in its "`customers`" table (We assume that a `customers` table exists in the shopping table and the table has two fields (columns): `userid` and `address`).

3. Collecting information to be stored in the table

```
userid = $_POST['userid'];
add = $_POST['add'];
```

These two statements are for collecting information from the `$_POST` array (containing the information – `userid` and `add` passed by the `enterinfo.php` program).

4. Writing SQL statement for inserting the record

```
$sql = "INSERT INTO customers (userid, address) VALUES ('$userid', '$add')";
```

This is the SQL statement for inserting the record into `customers` table of `shopping` database. It is a simple SQL statement where `$userid` and `$add` represent the information entered by the visitor and are enclosed in single quotes as both contain text matter.

5. Executing the SQL statement

```
$result = mysql_query($sql) or die(mysql_error());
```

This statement executes the SQL statement stored in `$sql` variable and the result of the execution (whether record insertion was successful or failure) is stored in `$result` variable.

Note: Inserting records in a table requires two PHP programs; one gathers information from the visitor and passes it to the other program, while the second program processes the information received from the earlier program. Processing here means any task such as inserting the information into the table, searching information from the table, deleting information from the table etc.

It also means that the process of searching records from the table (Section 4.6) and deleting records from the table (in Section 4.7) will also be requiring two programs, one for gathering information and the other for processing information.

4.5 RETRIEVING ALL RECORDS FROM THE TABLE

The information stored in the back-end database tables is often required to be retrieved for displaying it to visitors to our website. Good examples of records retrieval include the list of the products for sale on a website, display of facilities provided on different plans (PostPaid or Prepaid) on a telecommunication website or display of examination results on a university website etc.

Retrieving all records from the table requires following five steps:

1. Establishing a connection with the MySQL server
2. Selecting the database containing the table whose records have to be retrieved
3. Writing the SQL statement for selecting the desired fields from the table
4. Executing the SQL statement and storing the fetched records in a resultset
5. Fetching one row at a time from the result set and displaying it one by one

Let us look at the following program that fetches all the records from the `customers` table of `shopping` database and displays them on the screen in tabular format. All the steps followed in retrieval of records can be clearly seen in the program below:

customers_list.php

```
<?php
   $connect = mysql_connect("localhost", "root", "mce")
      or die("Please, check your server connection.");
   mysql_select_db("shopping");
   $query = "SELECT userid, address from customers ";
   $results = mysql_query($query) or die(mysql_error());
   echo "<table border=\"1\">\n";
  echo "<th>Userid</th><th>Address</th>\n";
  while ($row = mysql_fetch_array($results)) {
   extract($row);
   echo  "<tr>";
   echo  "<td>";
   echo  $userid;
   echo  "</td>";
   echo  "<td>";
   echo  $address;
   echo  "</td>";
   echo  "</tr>\n";
  }
```

```
    echo "</table>\n";
?>
```

Explanation

The five steps of retrieving all records from the table are clearly visible:

1. Establishing a connection with the MySQL server

```
$connect = mysql_connect("localhost", "root", "mce")
or die("Please, check your server connection.");
```

This statement establishes the connection with the local MySQL server with the userid as "root" and the password of the root is assumed to be "mce" (any text).

2. Selecting the database

```
mysql_select_db("shopping");
```

This statement selects the database "shopping" so that we can retrieve records from its customers table. (We assume that a customers table exists in the shopping database and it consists of two fields (columns): userid and address).

3. Writing the SQL statement for selecting the desired fields

```
$query = "SELECT userid, address from customers ";
```

This is the SQL statement that will retrieve userid and address fields (columns) from the customers table.

4. Executing the SQL statement

```
$results = mysql_query($query) or die(mysql_error());
```

This statement executes the SQL statement, and all the fetched records (rows) from the customers table are stored temporarily in the resultset: $results. The resultset is a sort of array where rows of the tables are fetched and temporarily stored.

5. Fetching one row at a time from the result set

```
while ($row = mysql_fetch_array($results)) {
```

The mysql_fetch_array() is a function used for extracting one record (row) at a time from the array of records - $results (resultset). In the above statement, we are using a while loop for extracting one row at a time from the resultset and storing it in a variable $row. When all rows from the resultset are extracted (i.e. when there are no rows left in the $results array),

`mysql_fetch_array()` function will return `false`, hence will come out of the `while` loop. The row returned by the `mysql_fetch_array()` from the resultset will be in a form of an associative array or a numeric array. In other words, the `$row` that will contain one row (record) at a time is an associative array or a numeric array.

In the `while` loop, we use extract function to extract the fields (columns) from array `$row` that is assumed to contain one row at a time of the table:

```
extract($row);
```

The extract function extracts all the fields (columns) stored in the specified array. After getting all the fields extracted from `$row` array, they are displayed one by one with the help of the following statements:

```
echo "<tr>";
echo "<td>";
echo $userid;
echo "</td>";
echo "<td>";
echo $address;
echo "</td>";
echo "</tr>\n";
```

The `$userid` and `$address` are the variables or fields that are extracted from `$row` array (by using `extract()` function) which contain the information of the userid and address.

4.6 SEARCHING RECORDS FROM THE TABLE

Searching is the most popular and frequent activity among visitors to a website. Visitors enter certain keywords (text) related to the product, service etc. which they are interested in and submit it. That keyword is searched for in the database tables and the matching rows (records) are displayed on the screen. A search engine is the most common example of searching records as here we specify the keyword related to the object whose information we want to look at. On submission of keyword, we get the desired information.

Let us make a searching program where a visitor enters the `userid` of the user whose information (address) he wants to access. The specified userid will be then searched in the `customers` table of `shopping` database and the information (address) of the matching record is displayed on the screen.

As said earlier, the tasks of insertion, searching, deletion etc. in the database require two PHP programs, one gathers information from the visitor and passes on to the next program which subsequently takes the necessary action.

Let us name the first PHP program which will prompt the visitor to enter `userid` (whose information has to be searched) as `search.php` and its code may appear as shown below:

search.php

```
<html>
   <head>
   </head>
   <body>
      <form action="searchcustomer.php" method="post">
         <table border="0" cellspacing="1" cellpadding="3">
            <tr><td>Userid: </td><td> <input size="20" type="text"
               name="userid"></td></tr>
            <tr><td><input type="submit" name="submit" value="Submit">
               </td><td><input type="reset" value="Cancel"></td></tr>
         </table>
      </form>
   </body>
</html>
```

We can see that in the above program, the `userid` entered by the visitor will be passed to the PHP program named: `searchcustomer.php` that will search the information of the user from the tables whose userid is passed. Searching records from the table requires the following steps:

1. Establishing a connection with the MySQL server
2. Selecting the database containing the table in which the desired record is supposed to be searched
3. Getting the criteria i.e. the condition to retrieve the record
4. Writing the SQL statement for searching the record
5. Executing the SQL statement and retrieving the desired record in the resultset
6. Fetching one row at a time from the result set and displaying it one by one

Let us study the following `searchcustomer.php` program that fetches the `userid` sent by `search.php` program and searches the `customers` table of `shopping` database and retrieves the desired information (address) of the `userid` supplied and displays it on the screen.

searchcustomer.php

```
<?php
   $connect = mysql_connect("localhost", "root", "mce")
```

```
    or die("Please, check your server connection.");
    mysql_select_db("shopping");
    $uid=$_POST['userid'];
    $query = "SELECT userid, address from customers where userid like
    '$uid'";
    $results = mysql_query($query) or die(mysql_error());
    echo "<table border=\"1\">\n";
    echo "<th>Userid</th><th>Address</th>\n";
    while ($row = mysql_fetch_array($results)) {
        extract($row);
        echo "<tr>";
        echo "<td>";
        echo $userid;
        echo "</td>";
        echo "<td>";
        echo $address;
        echo "</td>";
        echo "</tr>\n";
    }
    echo "</table>\n";
?>
```

Explanation

The six steps of searching the desired record from the table can be easily seen in the above program:

1. Establishing a connection with the MySQL server

```
$connect = mysql_connect("localhost", "root", "mce")
    or die("Please, check your server connection.");
```

This statement establishes the connection with the local MySQL server with the `userid` as `"root"` and the password as `"mce"`.

2. Selecting the database containing the table to be searched

```
mysql_select_db("shopping");
```

This statement selects the database "shopping" so that we can retrieve the desired record from its customers table.

3. Getting the criteria i.e. the condition to retrieve the record

```
$uid=$_POST['userid'];
```

As we know the current program is invoked by the earlier program search.php where the visitor was prompted to enter the userid of the person whose information is required. In search.php program, after entering the userid of the person, when Submit button is clicked, the current PHP program: searchuser.php will be invoked and here, we retrieve that userid (entered in the search.php program) through $_POST array and store it in a local variable: $uid.

4. Writing the SQL statement for searching the record

```
$query = "SELECT userid, address from customers where userid like '$uid'";
```

This is the SQL statement that retrieves the userid and address fields from the customers table whose userid is supplied from search.php file.

5. Executing the SQL statement and retrieving the desired record in the resultset

```
$results = mysql_query($query) or die(mysql_error());
```

Executing the SQL statement and storing the fetched records from the table in the resultset: $results

6. Fetching one row at a time from the result set and displaying it one by one

```
while ($row = mysql_fetch_array($results)) {
    extract($row);
    echo "<tr>";
    echo "<td>";
    echo $userid;
    echo "</td>";
    echo "<td>";
    echo $address;
    echo "</td>";
    echo "</tr>\n";
}
```

The mysql_fetch_array() is a function used for fetching one record at a time from the array of records ($results). It gets one row from the resultset and stores it in the variable $row.

The `extract()` function is used for extracting all the variables i.e. fields or columns stored in the `$row` array. The `$row` is an array that contains one record of the customer and the elements of this array are the fields (columns) of the record. The extracted variables `$userid` and `$address` (from `$row`) are displayed one by one with the help of `while` loop.

4.7 DELETING RECORDS FROM THE TABLE

In the task of deleting a record from the database table, again we will require two PHP programs; one of them will prompt the visitor to enter the `userid` of the user to be deleted. After entering the `userid`, when the visitor clicks the Submit button, that `userid` will be passed to another PHP program which then deletes the record from the `customers` table that contains the supplied `userid`. There is no need of making the first PHP program (that asks the visitor to enter `userid`) from scratch as we can use the same one that we used in searching program (Section 4.6), i.e. `search.php`. The only change we need to do in that program is that we replace the following statement:

```
<form action="searchcustomer.php" method="post">
```

with the below statement:

```
<form action="deletecustomer.php" method="post">
```

because this time, we want the `userid` to be passed to `deletecustomer.php` program instead of `searchcustomer.php` program.

The task of deleting records from the table requires the following steps:

1. Establishing a connection with the MySQL server
2. Selecting the database containing the table from where we want to delete the record(s)
3. Getting the criteria i.e. the condition to delete the record
4. Writing the SQL statement for deleting the record(s)
5. Executing the SQL statement to physically delete the record(s) from the table

Let us study the following `deleteuser.php` program that fetches the `userid` sent by `search.php` program and deletes the record with the matching `userid` from the `customers` table of `shopping` database and displays a message: "Record successfully deleted" on successful deletion of record.

deleteuser.php

```
<?php
   $connect = mysql_connect("localhost", "root", "mce")
      or die("Please, check your server connection.");
   mysql_select_db("shopping");
   $uid=$_POST['userid'];
```

```
$sql = "DELETE from customers where userid like '$uid'";
$res = mysql_query($sql) or die (mysql_error());
echo "Record successfully deleted";
?>
```

Explanation

The five steps of deleting the desired record(s) from the table can be seen:

1. Establishing a connection with the MySQL server

```
$connect = mysql_connect("localhost", "root", "mce")
or die("Please, check your server connection.");
```

This statement establishes the connection with the local MySQL server with the `userid` as "`root`" and the password as "`mce`".

2. Selecting the database containing the table

```
mysql_select_db("shopping");
```

This statement selects the database "`shopping`" so that we can delete the desired record from its `customers` table.

3. Getting the criteria i.e. the condition to delete the record

```
$uid=$_POST['userid'];
```

We know that the current program `deleteuser.php` is invoked by an earlier program named `search.php` where the user was asked to enter the `userid` of the person whose information has to be erased. The `userid` entered in that program is passed to the current program and here, we retrieve it through `$_POST` array and store it in a local variable: `$uid`.

4. Writing the SQL statement for deleting the record(s)

```
$sql = "DELETE from customers where userid like '$uid'";
```

This is the SQL statement that deletes the record from the `customers` table whose `userid` is supplied from `search.php` file.

5. Executing the SQL statement to delete the record(s)

```
$res = mysql_query($sql) or die (mysql_error());
```

Executing the SQL statement that physically deletes the record from the `customers` table of the `shopping` database.

SUMMARY

In this chapter, we have seen different functions that are used for accessing MySQL tables from PHP programs. Then using these functions we have seen all the modules that are essential in making a complete website. We also saw the statements used in these modules along with their meanings. The modules that we saw in this chapter were: Creation of table, Getting information from the user, Inserting records in the table, Retrieving all records from the table, Searching desired records from the table and Deleting records from the table.

In the next chapter, we will see the sample output of our Shopping Cart Project. We will see step-by-step the different outputs that will appear on selecting different links and buttons on our website. Also we will see how different items can be selected in the Shopping Cart, how Shopping Cart is updated and finally how an order is placed on our website.