Handling Databases in PHP

1.1 Unit Structure

- Learning Objectives.
- Introduction.
- Connecting PHP with MySQL.
- Database Manipulation Operations – Insert, Update, Delete.
- Retrieving Records from Database.

1.2 Learning Objectives

After completion of this unit students will be able to

- Connect PHP with MySQL.
- Perform database manipulation operations like Insert, Update and Delete in PHP.
- Retrieve data from tables and represent them in readable format.

1.3 Introduction

Data is an important entity for any web site and hence it should be properly organized and retrieved when needed. In web applications, we need to store various kinds of data such as if we are building an online shopping site, then we may need to store data such as customer details, supplier details, products, orders, invoice details etc. For this, we have to design a form to collect data from various stakeholders of the web site and then store these data in the proper table. To do so, we have to connect PHP with database application. In this unit we are going to learn how can we connect PHP with MySQL and how can we perform various database operations on it.

1.4 Connecting PHP with MySQL

The most common and popular database used with most of the PHP web applications is MySQL database because the connectivity of PHP with MySQL database is very easy.

To connect PHP 5 and later versions with database we can use:

- MySQLi extension ('i' stands for improved)
- PDO (PHP Data Objects)

Earlier versions use MySQL extension which was deprecated in 2012.

MySQLi Extension: To connect PHP with MySQL database, the MySQLi extension is used. The following function is used to connect PHP with MySQL.

mysqli_connect($servername, $username, $password, $databasename)

- $servername – Name or IP address of server/host.
- $username – MySQL username
- $password – MySQL Password
- $databasename – Database name to be used.

For example

$con = mysqli_connect("localhost","root","","student");

The above function establishes a connection with MySQL database called student. In the server's name, we have mentioned localhost because our database is located in the local server. Here root is default username of MySQL database and default password is blank. If you have set another username and password, then you need to mention that username and password. "student" is the name of database that we want to connect with. We need to create a student database first in MySQL. The function returns the object representing the connection to MySQL server. It returns False on failure.

If there is no error means we have established a successful connection with MySQL database called student.

Example:

```
<?php
$con = mysqli_connect("localhost","root","","student");
if ($con){
echo ("Database connection successful");
}
 Else {
die("Can't Connect. Connection
Error:".mysqli_connect_errno()."".mysqli_connect_error());
}
?>
```

The above code will establish the connection of PHP with MySQL student database. If there is no error, then it prints the message "Database connection successful". Else, in case of failure, it shows error with error number and description with the use of functions like mysqli_connect_errno() and mysqli_connect_error().

1.5 Database Manipulation Operations – Insert, Update, Delete

To perform any database manipulation operations like inserting a record, updating the record or deletion of record requires to execute respective queries. mysqli_query() function is used to execute any SQL query in PHP. We can use it as:

```
$result=mysqli_query($con,$query);
```

The above function is used to perform SQL queries on the mentioned database table.

- $con is the connection object that we set using mysqli_connect().
- $query is a SQL query that we want to execute on the database table.

Upon successful select queries, the function returns mysqli_result object. Else, it returns False on failure.

1.5.1 PHP-MySQL connection example – Inserting record in table:

The following example shows how we can insert a record in the student_details table located under student database. The structure of the student_details table is as follows:

Field name	Field Type
sno	Integer
name	Varchar
city	Varchar
pin	Integer

You need to create the student database in MySQL and then create student_details table under it. To create the database, and table we can use phpMyAdmin, a web interface to handle MySQL. Just type localhost/phpmyadmin in the browser address bar to open it. You can create databases and tables over there.

Inserting data into the student table

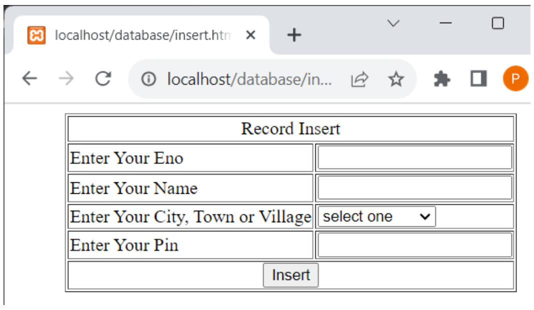
```
<?php
$con = mysqli_connect("localhost","root","","student");

if (!$con) {
    die("Can't Connect. Connection Error:".mysqli_connect_errno()."
".mysqli_connect_error());
}
$sno=1001;
$name="James";
$city="Port Louis";
$pin=80410;

$query = "insert into student_details values($sno,'$name','$city',$pin)";
$result = mysqli_query($con, $query);
if($result)
{
echo "Record inserted successfully";
}
else
{
echo "Record not inserted <br>";
echo "error no is:".mysqli_errno($con)."Error is:".mysqli_error($con);
}
mysqli_close($con);
?>
```

The above code inserts one record in the student_details table. Here we have directly specified the values of the fields. We can also design input form to enter all those details. Let's see how we can do that. We have created two files – one is input design form named as insert.html and other is PHP file where connection code is written which is named as insertrecord.php

Insert.html

```
<html>
<body>
<form method="post" action="insertrecord.php">
    <table border="1" align="center">
        <tr>
            <td colspan=2 align="center">Record Insert</td>
        </tr>
        <tr>
            <td>Enter Your Eno </td>
            <td><input type="text" name="eno"></td>
        </tr>
        <tr>
            <td>Enter Your Name </td>
            <td><input type="text" name="name"></td>
        </tr>
        <tr>
            <td>Enter Your City, Town or Village </td>
            <td><select name="city">
                <option value="Select One" selected>select one</option>
                <option>Rose Hill</option>
                <option>Port Louis</option>
                <option>Curepipe</option>
                <option>Central Flacq</option>
                <option>Mahebourg</option>
                </select>
            </td>
        </tr>
        <tr>
            <td>Enter Your Pin </td>
            <td><input type="number" name="pin"></td>
        </tr>
        <tr>
            <td colspan=2 align="center"><input type="submit" value="Insert"></td>
        </tr>
</table>
</body>
</html>
```

The output is as follows:



Record Insert	
Enter Your Eno	
Enter Your Name	
Enter Your City, Town or Village	select one
Enter Your Pin	
Insert	

Insertrecord.php:

```
<?php
$con = mysqli_connect("localhost","root","","student");

if (!$con)
{
    die("Can't  Connect. Connection Error:".mysqli_connect_errno()."
".mysqli_connect_error());
}
$sno=$_POST['eno'];
$name=$_POST['name'];
$city=$_POST['city'];
$pin=$_POST['pin'];
```

```
$query = "insert into student_details values($sno,'$name','$city',$pin)";
$result = mysqli_query($con, $query);

if($result){
    echo "Record inserted successfully";
}
else {
    echo "Record not inserted <br>";
    echo "error no is:".mysqli_errno($con)."Error is:".mysqli_error($con);
}
mysqli_close($con);
?>
```

The insert.html file is executed first. After adding the different fields and the insert button is clicked, the user will be redirected to the insertrecord.php file where all the collected information from the form are posted with the $_POST[] super global array. The mysqli_query() executes the insert query and add the record in student_details table. You can cross verify by opening student_details table in PHPmyAdmin.

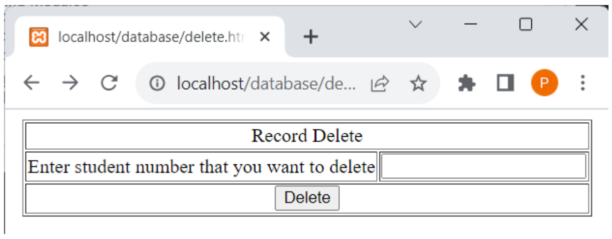
1.5.2 Delete operation.

Sometimes we need to delete the records from the MySQL table. There are multiple reasons of delete such as duplicate record, unwanted entry, wrong entry etc. To delete the record we need to execute delete query in mysqli_query(). The following example shows the delete operation.

Delete.html

```
<html>
<body>
<form method="post" action="deleterecord.php">
    <table border="1" align="center">
        <tr>
            <td colspan=2 align="center">Record Delete</td></tr>
        <tr>
            <td>Enter student number you want to delete </td>
            <td><input type="text" name="eno"></td>
        </tr>
        <tr>
            <td colspan=2 align="center"><input type="submit" value="Delete"></td>
        </tr>
    </table>
</body>
</html>
```

Output:



Record Delete	
Enter student number that you want to delete	
Delete	

This file is created to ask for the student enrolment number that we want to delete from student_details table.

deleterecord.php

```
<?php
$con = mysqli_connect("localhost","root","","student");

if (!$con){
    die("Can't  Connect. Connection Error:".mysqli_connect_errno()."
".mysqli_connect_error());
}

$eno=$_POST['eno'];

$query = "delete from student_details where sno=$eno";
$result = mysqli_query($con, $query);

if(mysqli_affected_rows($con) >= 1){
    echo "Record deleted successfully";
}
else {
    echo "Record not deleted <br>";
    echo "error no is:".mysqli_errno($con)."Error is:".mysqli_error($con);
}
mysqli_close($con);
```

Here, to confirm that the record is deleted or not we have used mysqli_affected_rows() function. The mysqli_affected_rows() function returns the number of affected rows in the table by the SELECT, INSERT, UPDATE or DELETE queries. If it returns one means one row is affected by the executed query. Here, if the return value of the function is one or greater than one, it means at least one record is deleted.

1.5.3 Update Operation

Like insert and delete records, update record is also an important operation required to be performed on the database. We need to perform update operation in case wrong entries are inserted or the entries whose value has now changed (For example, the student has changed his/her mobile number, address, or a student got married and change her surname). The following example shows how to perform update operation on tables in PHP.

Update.html

```
<html>
<body>
<form method="post" action="update.php">
Enter student number to update: <input type="text" name="num1">
<input type="submit" value="Search">
</body>
</html>
```

Update.php

```
<form name="fname" method="post" action="updaterecord.php">
<?php
$con = mysqli_connect("localhost","root","","student");
if (!$con){
    die('Connect Error: ' . mysqli_connect_errno().mysqli_connect_error());
}

$no=$_POST['num1'];
$query = "select * from student_details where sno='$no'";

$result = mysqli_query($con, $query);

    if(mysqli_affected_rows($con) == 0){
        echo "No Record found";
    }
    else{
        while($row = mysqli_fetch_row($result)){
?>

<table border="1">
    <tr>
        <td>Sno</td><td><input type="text" name="num" value="<?php echo $row[0];?>"
readonly></td>
    </tr>
    <tr>
        <td>Sname</td><td><input type="text" name="sname" value="<?php  echo
$row[1]; ?>"></td>
    </tr>
    <tr>
        <td>Scity</td><td><input type="text" name="scity" value="<?php echo $row[2];
?>"></td>
    </tr>
    <tr>
        <td>Pin</td><td><input  type="text" name="pin"value="<?php  echo $row[3];
?>"></td>
    </tr>
    <tr>
        <td colspan="2" align="center"><input type="submit" value="Update"></td>
    </tr>
<?php
}

}
mysqli_close($con);
?>
```

This file shows the existing records of the student before update. Here we have fetched the records from the database and put these values in the textboxes in edit mode so that the user can change them. To retrieve records from the table, we have used mysqli_fetch_row() method of data fetching. In this file user sees the existing records, update them and then press update button which redirects the user to the updaterecord.php file.

updaterecord.php

```
<?php
$con = mysqli_connect("localhost","root","","student");
if (!$con){
    die('Connect Error: '. mysqli_connect_errno().mysqli_connect_error());
}
$no=$_POST['num'];
$name=$_POST['sname'];
$city=$_POST['scity'];
$pin=$_POST['pin'];

$query = "update student_details set sno=$no, name='$name', city='$city', pin=$pin
where sno='$no'";

$result = mysqli_query($con, $query);

if(!$result){
    echo "error no is:".mysqli_errno($con)."Error is:".mysqli_error($con);
}
else {
echo "record updated successfully";
}
mysqli_close($con);
```

```
?>
```

1.6 Retrieving Records from Database

After inserting and updating the records, retrieving records from database is equally important. In web applications we need to retrieve the data stored in database tables for various purposes such as if we consider online shopping site, then we need to fetch the data from tables to generate customer bill, to view products, to view suppliers and customers, to find out the monthly sales of a product, to search particular product, to display shopping cart etc.

PHP supports different data fetching methods which are used to fetch records from tables. These methods are:

- mysqli_fetch_row()
- mysqli_fetch_assoc()
- mysqli_fetch_array()
- mysqli_fetch_object()

In this section we are going to learn these methods in detail

1.6.1 mysqli_fetch_row(result)

This method fetches one row from a result-set and returns it as an indexed array. Returns NULL in case of no rows in result set.

Result is an identifier returned by mysqli_query(). Let's take an example where we need to search student by its enrolment number.

Example: Searchbox.html

```
<html>
<body>
<form method="post" action="srch.php">
Enter enrolment num to search: <input type="text" name="no">
<input type="submit" value="Search">
</body>
</html>
```

Above code will ask the user to enter the enrolment number of student whom we are searching for the details. By clicking the search button, it will redirect to srch.php file.

srch.php file

```
<?php
$con = mysqli_connect("localhost","root","","student");
$no=$_POST['no'];
$query = "select * from student_details where sno='$no'";
$result = mysqli_query($con, $query);
if(mysqli_affected_rows($con) == 0){
    echo "Record not found";
}
else {

    while($row = mysqli_fetch_row($result)){ // here $row becomes any index array
and size of array = total fields in table.
        echo "Sno:".$row[0]; // retrieve first field value which is sno
        echo "Sname:".$row[1]; // retrieve second field value which is student name.
        echo "City:".$row[2]; // retrieve third field value which is student city.
        echo "Pin:".$row[3]; // retrieve forth field value which is pincode.
    }
}
?>
```

The above script will perform select query in student_details table to retrieve the details of the student whose enrolment number is entered by user in srchbox.php file. mysqli_fetch_row() method will fetch the details in the form of indexed array and hence $row now becomes an index array whose size is equivalent to number of fields in the table. Here in student_details table, we have total 4 fields (sno, name, city, pin) so the $row array size is 4. Though it is an index array, its values are retrieved by integer index like $row[0], $row[1], $row[2], $row[3]. We put mysqli_fetch_row() in while loop so if there are multiple records, then the loop will continue till it gets the details of the next records.

1.6.2 mysqli_fetch_assoc(result)

This method fetches a result row as an associative array. Returns NULL in case of no rows in result set. Result is an identifier returned by mysqli_query().
The difference between mysqli_fetch_row and mysqli_fetch_assoc is the return type. The first one returns index array while the second one returns associative array.

Let's take the same example and retrieve the student record using mysqli_fetch_assoc(). Srchbox.html will remain the same. We need to change the code of srch.php file.

srch.php file

```php
<?php
$con = mysqli_connect("localhost","root","","student");
$no=$_POST['no'];
$query = "select * from student_details where sno='$no'";
$result = mysqli_query($con, $query);
if(mysqli_affected_rows($con) == 0) {
    echo "Record not found";
}
else {
        while($row = mysqli_fetch_assoc($result)) {
            echo "Sno:".$row["sno"];
            echo "Sname:".$row["name"];
            echo "City:".$row["city"];
            echo "Pin:".$row["pin"];
        }
}
?>
```

If you observe the difference between these two methods, then it states that in the previous method, we have used integer index to retrieve records. In this method, we have used table field names as array index because here, the method returns associative array where the values are retrieved by its key. Here fields names are keys for the array $row.

1.6.3 mysqli_fetch_array(result, returntype)

This method fetches a result row as an associative array, a numeric array, or both. Returns NULL in case of no rows in result set. Result is an identifier returned by mysqli_query(). returntype is used to select the return type of array. Possible values are

- MYSQLI_NUM – if you want to return only index array.
- MYSQLI_ASSOC – if you want to return associative array.
- MYSQLI_BOTH (this is default) – Both type of array will be returned.

mysqli_fetch_array() returns both index as well as associative array by default and hence you can use integer index or key to retrieve values. Let’s continue the same example using mysqli_fetch_array().

srch.php

```
<?php
$con = mysqli_connect("localhost","root","","student");
$no=$_POST['no'];
$query = "select * from student_master where sno='$no'";
$result = mysqli_query($con, $query);
if(mysqli_affected_rows($con) == 0){
    echo "Record not found";
}
else {
      while($row = mysqli_fetch_array($result))
      {
        echo "Sno:".$row[0]; // retrieve student number by index array
        echo "Sname:".$row[1]; // retrieve student name by index array
        echo "City:".$row["city"]; // retrieve student city by the key city.
        echo "Pin:".$row["pin"]; // retrieve student pin code by the key pin.
       }
}
?>
```

1.6.4 mysqli_fetch_object(result)

This method returns the current row of a result-set, as an object. Returns NULL in case of no rows in result set. Result is an identifier returned by mysqli_query(). Apart from this method, all the three previous methods that we have learned so far return either index array, associative array or both. However, mysqli_fetch_object() returns the current row as an object. Fields names are used along with object to retrieve data from tables.

Let’s now retrieve data from student_details table using mysqli_fetch_object()

Example:

Srch.php

```php
<?php
$con = mysqli_connect("localhost","root","","student");
$no=$_POST['no'];
$query = "select * from student_details where sno='$no'";
$result = mysqli_query($con, $query);
if(mysqli_affected_rows($con) == 0){
      echo "Record not found";
}
else{
    while($row = mysqli_fetch_object($result)){
        echo "Sno:".$row->sno;
        echo "Sname:".$row->name;
        echo "City:".$row->city;
        echo "Pin:".$row->pin;
    }
}
?>
```

In this unit, we have discussed all the data fetching methods with examples. You can use any of these methods while retrieving records from a database.