Lab 08: Working with Database

In this lab exercise, you will learn how to connect to MySQL database. This marks the beginning of the more complex topics of this course. Ensure you have successfully completed all tasks from all previous topics, as they serve as foundation and core skills to move forward hereon.

1.1 Managing MySQL Database Server

Instruction on how to install MySQL database server is outlined in Lab 01. Ensure that it has been successfully installed before moving forward in this lab exercise.

MySQL database server can be managed using MySQL Workbench application. Establish connection to an instance that has been assigned to you. For local work, you may use localhost.

1.2 Creating a Schema

Before you can work with a database server to store, retrieve, and/or manipulate data, a schema needs to be created for your PHP application.

Create a schema as per Figure 1 below. Name the schema as classlabs.

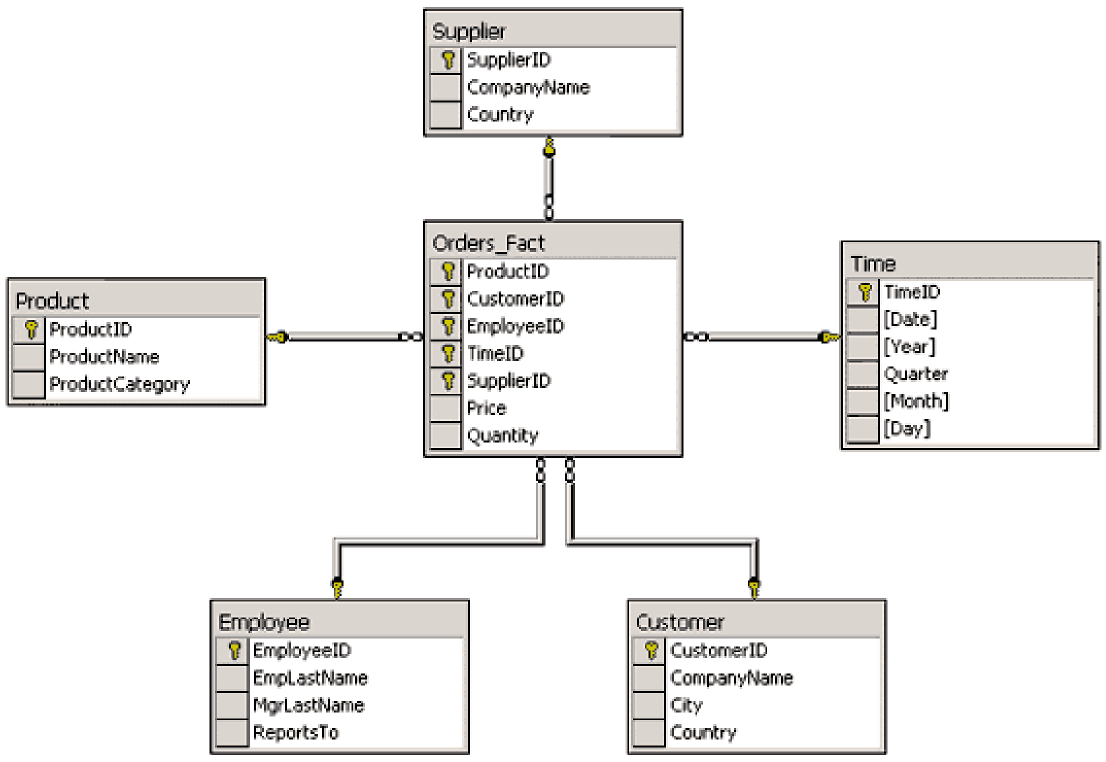


Figure 1: Database schema to be created for this and subsequence lab exercises.

1.3 Connecting to Database Server

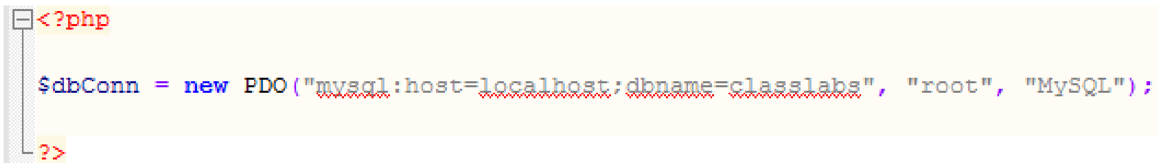
There are 2 common methods to connect to MySQL database in PHP application, i.e. MySQLi and PDO (PHP Data Objects).

Versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

Both MySQLi and PDO have their advantages. PDO will work on 12 different database systems, where as MySQLi will only work with MySQL databases. If you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code, including queries.

Both are object-oriented, but MySQLi also offers a procedural API. Both also support Prepared Statements, which are very important for web application security as they protect from SQL injection.

In this course, we will use PDO. Figure 2 below shows how to connect to MySQL database.

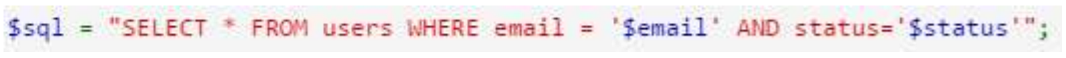


```
<?php

$dbConn = new PDO("mysql:host=localhost;dbname=classlabs", "root", "MySQL");

?>
```

Figure 2: Sample connection string to MySQL using PDO method.

1.4 Preparing SQL Statement

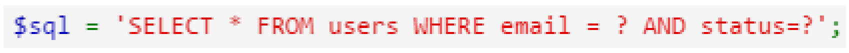
SQL statement for execution may normally look like in the Figure 3 below.



```
$sql = "SELECT * FROM users WHERE email = '$email' AND status='$status'";
```

Figure 3: SQL statement normally used in programming.

The method above, however, does not provide protection from SQL injection attack. An alternative is to use placeholder to hold the values. Sample is shown in Figure 4 below.



```
$sql = 'SELECT * FROM users WHERE email = ? AND status=?';
```

Figure 4: Alternative way to write SQL statement using placeholder. This provides protection from SQL injection attack.

The above applies to all SQL statements: INSERT, SELECT, UPDATE, DELETE, and other DB related SQL statements.

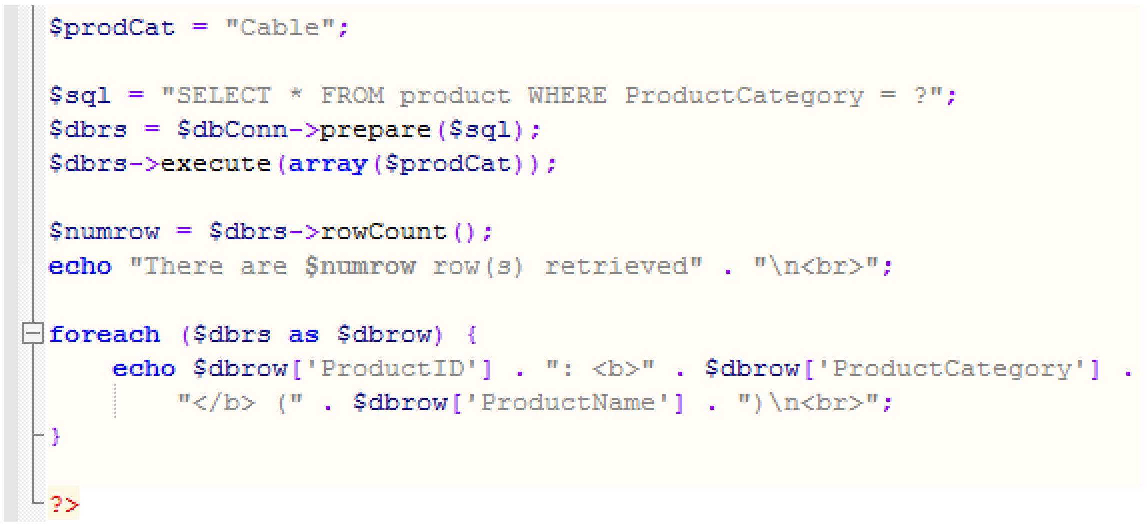
1.5 Running SQL Statement

An SQL statement is executed with the following steps:

1. Write an SQL statement to execute
2. Call PDO connection object to prepare the SQL for execution
3. Call an execution keyword

Number of row(s) affected can be retrieved using keyword rowCount ( );

Figure 5 below illustrates how to execute a SELECT statement and retrieve the recordset result and number of row(s) affected.



```
$prodCat = "Cable";

$sql = "SELECT * FROM product WHERE ProductCategory = ?";
$dbrs = $dbConn->prepare($sql);
$dbrs->execute(array($prodCat));

$numrow = $dbrs->rowCount();
echo "There are $numrow row(s) retrieved" . "\n<br>";

foreach ($dbrs as $dbrow) {
    echo $dbrow['ProductID'] . ": <b>" . $dbrow['ProductCategory'] .
        "</b> (" . $dbrow['ProductName'] . ")\n<br>";
}

?>
```

Figure 5: Sample code to retrieve row(s) from database table product.

You tasks:

1. Create a page that allows the user to select information to retrieve from a table.
    a. Information is to be displayed in an HTML table form.
    b. Maximum number of record to display is 10 rows per page.
2. Create a link on each record to allow the user to:
    a. Insert a new record.
    b. Update an existing record.
    c. Delete an existing record.
3. A message displaying number of row(s) affected in execution is placed at the bottom of page.

-- END --