Chapter 8
Using SQL in DB Application

We have been discussing SQL as an interactive language: We enter an SQL statement, send it over to the server, where the DBMS takes over to evaluate the query and send back the result.

For example, we can try the following query to find out the one who has the highest GPA, among her peers:

```
Select S.Name, S.Id From Student S
Where S.GPA >= (Select Max(S1.GPA) From Student S1);
```

and we will get the following back

```
+--------------+-----------+
| Name         | Id        |
+--------------+-----------+
| Bart Simpson | 987654321 |
+--------------+-----------+
```
Get real....

In most of the transaction processing, though, SQL statements are part of an application program written in a *host language*, such as *C, Cobol, Java, or PhP*, etc., and the program executes in a client workstation.

This often provides a much secure, friendlier, and even faster, user interface.

For example, to get the same information, we can run a web based script (Sec. 3 of the *PhP* notes)

http://turing.plymouth.edu/~zshen/PhPFiles/sendSpecificQuery.html

In this chapter, we will address the issues as *how to embed MySQL queries within a PhP program*. 
The key difference

In a procedural language such as C, Java, etc., a variable, often called a “host variable”, can hold *one value* at a time. On the other hand, a SQL query can return a *set, of a collection, of values*, which won’t fit in a host variable. 😞

**Question:** What to do?

**Answer:** Use a loop, often referred to as a “cursor”, in the host language, which processes one value (row) at a time until everything in the set is processed. Problem solved. 😊

Let's look at a general framework, and then a few examples to demonstrate the process.
A general framework

It is pretty easy to work with MySQL in PHP with a HTML based interface. A typical process could be the following:

0. Enter your credentials, such as username and password.

1. Establish a connection with the database(s) that you want to work with.

2. Select the table(s) you want to work with.

3. Construct a MySQL query to send to the database.

4. Get back the result.

5. Start an HTML table, using the <Table> tag.
6. Loop through the database result rows, with a cursor, and place it into each row of the table, using a <Tr></Tr> structure.

7. In each row, retrieve the successive fields and place it into that row, using a <Td></Td> structure.

8. Close off the HTML table, using the </Table> tag.

9. Close the database connection.

For details, check out § 5 of the Php notes, which goes through database programming in terms of Php, MySQL and HTML; and discuss several examples in details.

You will play with those stuff in the labs. (Have you checked out the course page recently?)
The workhorse in PhP

The following function prints an HTML table, i.e., Steps 3 through 8 in the above process.

```php
function display_db_table($ tablename , $ connection )
{
// Get all the stuff for the table, i.e., Step 3
$query_string = "Select * from $ tablename " ;
// Execute the query with the connection
$result_id = mysql_query ( $query_string , $ connection ) ;
// mysql_num_fields returns the number of attributes
$column_count = mysql_num_fields ( $result_id ) ;
// Start a table, i.e., Step 5
print ( "<Table border=1>\n" ) ;
// Here is the cursor and the fetch, Steps 6 and 7
// It runs as long as the fetch sends back something
while ( $ row = mysql_fetch_row ( $result_id ) )
{
print ( "<Tr Align=left V align=top>" ) ;
for ( $ column_num = 0 ; $ column_num < $column_count ; $ column_num++ )
print ( "<Td>$ row [ $ column_num ] </Td>\n" ) ;
print ( "</Tr>\n" ) ;
}
// Wrap it up, i.e., Step 8
print ( "</Table>\n" ) ;
}
```

It is described in detail in §5.4 of the PhP lab notes. Another similar function `display_db_query` is given at Page 18 of this set of notes.
Example: countries and cities

We have collected some information about countries, the continent they are located, and a few cities in each of those countries.

<table>
<thead>
<tr>
<th>ID</th>
<th>continent</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Africa</td>
<td>Kenya</td>
</tr>
<tr>
<td>2</td>
<td>South America</td>
<td>Brazil</td>
</tr>
<tr>
<td>3</td>
<td>North America</td>
<td>USA</td>
</tr>
<tr>
<td>4</td>
<td>North America</td>
<td>Canada</td>
</tr>
</tbody>
</table>

Its structure can be defined as follows in MySQL:

```sql
CREATE TABLE Country (
    ID Int(11) Not Null Auto_Increment Primary key,
    continent Varchar(50),
    countryname Varchar(50))
```

Notice the **Auto_increment** mechanism, which provides an automatically incremented key for the field. 😊
Information about the cities

Below is the data for the City table, together with its MySQL declaration.

<table>
<thead>
<tr>
<th>ID</th>
<th>countryID</th>
<th>cityname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Nairobi</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Meru</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Mombasa</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Rio</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>San Paulo</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Salvador</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>Boston</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>Chicago</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>Houston</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>Windsor</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>Montreal</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Winnipeg</td>
</tr>
</tbody>
</table>

Create table City (  
    ID Int(11) Not Null Auto_Increment Primary key,  
    countryID Varchar(50),  
    cityname Varchar(50))
How to print out cities?

<?php
   //0. The following keeps the display_db_table function
   include("singleTableDisplay.inc");

   //1. Set up the db connection
   $global_dbh=mysql_connect($hostname,
       $user, $password);

   //2. Choose a database to work with
   $db="zshen";

   //Connect to the database
   mysql_select_db($db, $global_dbh);
?>

<HTML><Head><Title>Cities</Title></Head><Body>
   <Table><Tr><Td>
      <?php
      //Call the function
      display_db_table("city", $global_dbh);
      ?>
   </Td></Tr>
   </Table>
</Body></HTML>

Let's run this testSingleTableDisplayClass.php script, which can be found in §5.4 of the Php notes.
Where is the secret?

**Question:** What about the credentials?

You want to keep the confidential stuff in a file, e.g., `phpbook-vars.inc`, as follows:

```php
<?php
    //Critical data to make the connection
    $hostname='localhost';
    $user='zshen';
    $password='Password';
?>
```

You can then give its location a rather strong access right (Remember the `chmod` command?), and lock it up so that only certain class of users can get access to such a file.

This file can be included in `singleTableDisplay.inc` as shown a bit later.
Any better way?

Besides the relocation and locking mechanism, another way is to encrypt such information, with, e.g., the following `passwordTest.php` script, where `hash` is a system function as provided by `Php`

```php
<?php
    $test = "password";
    $password = hash("sha256", $test);
    echo $test;
    echo "\n";
    echo $password;
?>
```

If you run this script, you will see the following stuff printed.

```
password
5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8
```
Is it secure?

Such a function hash is a one-way function in the sense that it takes just a little time to do the encryption, but its decryption takes years, if not centuries. Thus, it is impossible, at least improbable, for someone to figure out this string really goes back to the word “password”. 😊

**Question:** With this setting, how do you verify the credentials?

**Answer:** Keep the above encrypted information, encrypted, with user in a table. When you get a password passwd from a user user, do the same encryption of passwd, get out encrypted for user, and do a comparison of the encrypted version of passwd and encrypted... .

Check out §5.1.1 of the *PhP* notes, and the additional information on the course page.
<?php
//The following file contains the credential info
include("home/phpbook/phpbook-vars.inc");
function display_db_table($tablename,$connection)
{//3. A query to get all the stuff for the table
$query_string="Select * from $tablename";

//4. Get the result by executing the query
$result_id=mysql_query($query_string,$connection);
//How many columns?
$column_count=mysql_num_fields($result_id);

//5. Start a table
print("<Table border=1>\n");

//6. For each row contained in the result
while($row= mysql_fetch_row($result_id)){
  print("<Tr Align=left Valign=top>");

  // 7. Get columns for each row
  for($column_num=0; $column_num< $column_count;
       $column_num++)
    print("<Td>$row[$column_num]</Td>\n");

  //8. Finish off
  print("</Tr>\n");
}
print("</Table>\n");}?>

This file might contain other functions.
Static SQL

In the very first example of this chapter, a given SQL statement

$\text{query} \_\text{string} = \text{"Select S.Name, S.Id, S.GPA From Student S Where S.GPA} >= (\text{Select Max(S1.GPA) From Student S1})"$

is embedded in the program highestGPA.php, when this program is written, thus static SQL: We know what to do before running the program.

In a static SQL, everything, such as the schema information of the involved database, as well as the host variables used in the I/O process, e.g., $\text{query} \_\text{string},$ are known.

**Question:** What happens if we don’t know all the stuff before the execution time?
A *PhP* example

The following example, `sendGeneralQuery.html`, prints out the output of *any* query, which can be found in §5 of the *MySQL* lab notes.

```html
<html>
<!-- This file sends a query, to be picked up and -->
<!-- processed by showGeneralQueryResult.php -->
<head>
<style type="text/css">
/*
Body, P, TD {color: black; font-family: verdana;
    font-size: 10 pt}
H1 {color: black; font-family: arial; font-size:12 pt}
*/
</style>
</head>

<!--A table with only one row, consisting of two cells, -->
<!--the first being the left edge, 1/6; and the other -->
<!--contains the form, 5/6 -->
<table border=0 cellPadding=10 width=100%>
<!--Now define the row-->
<tr>
<!--The following cell gives the left cushion edge-->
<td bgcolor="#FFF0F0" align=center valign=top width=17%>
</td>
```

15
<!--The following gives the right entry form part, -->
<!-- "FFFFFF": completely white-->
<Td BGColor="FFFFFF" Align=Left VAlign=Top Width=83%>

<H1>Query submission form;</H1>
<p>Enter your query and we will send you back the result. </p>

<!--Your general query will be transmitted-->
<!--via the post method -->
<Form Method="post" Action="showGeneralQueryResult.php">
<!--First text box will be called $_POST['query'] -->
<!--in the handler. -->
<Input Type="text" Size=25 Name="query">
</Form>
</Td>
<!--end of the row definition>
</Tr>
</Table></Body></Html>
This is what it takes....

The following shows showGeneralQueryResult.php.

```php
<?php
  //What is this?
  include("displayQueryResult.inc");
?>

<html>
  <head>
    <title>The result of a Query</title>
  </head>
  <body>

    <!--A table with only one row, consisting of two cells,-->  
    <!--the first being the left edge, 1/6; and the other -->
    <!--contains the form, 5/6 --> 
    <table border=0 cellPadding=10 width=100%>
      <!--Now define the row-->
      <tr>
        <!--The following cell gives the left cushion edge-->
        <td bgcolor="#F0F8FF" align=center vAlign=top width=17%>
          17
        </td>
      </tr>

  </body>
</html>
```
```php
<?php
//The following gives the right entry form part,
//completely white
<Td BGCOLOR="FFFFFF" Align=Left VAlign=Top Width=83%>

//Below gets the query passed over via the post method
$query_string=$_POST['query'];

//echo the query
print("The following displays the result of a query:
$query_string.<BR><BR>");

//Call the predefined function, as contained in
//displayQueryResult.inc, to print out the cellar
//table, together with column titles "True" and an
//appropriate border, 2

display_db_query($query_string, $global_dbh,
    TRUE, "Border=2");
?>
</Td>
<!--end of the row definition>
</Tr>
</Table>
</Body>
</Html>

**Question:** Where is the beef? i.e., where is display_db_query defined?
<?php

// Private information
include("home/phpbook/phpbook-vars.inc");

// Set up all the other needed information
// Where did it die?
$global_dbh=mysql_connect($hostname, $user, $password)
    or die("Could not connect to database");

// Set the name of the database you want to work with
// You have to change it to your database.
$db="registration";

// Select the database to work with
mysql_select_db($db, $global_dbh)
    or die("Could not select database");

// This the function to display the result of the query
function display_db_query($query_string, $connection,
    $header_bool, $table_params){

    // Prepare and execute the dynamic query
    $result_id=mysql_query($query_string, $connection)
        or die("display_db_query:". mysql_error());

    // find out the number of columns in result
    $column_count=mysql_num_fields($result_id)
        or die("display_db_query:". mysql_error());
// Table form include optional HTML arguments
print("<Table $table_params >\n");

// Optionally print a bold header at top of table
if($header_bool){
    print("<Tr>");
    for ($column_num=0; $column_num< $column_count; $column_num++)
    {
        $field_name=mysql_field_name($result_id, $column_num);
        print("<Th>$field_name</Th>");
    }
    print("</Tr>");
} // end if

// Print out the body of the table
while($row= mysql_fetch_row($result_id)){
    print("<Tr Align=left Valign=top>");
    for ($column_num=0; $column_num< $column_count; $column_num++)
    {
        print("<Td>$row[$column_num]</Td>
        print("</Tr>
    }
} // end while
print("</Table>");
} // end display_db_query
?

Are you ready for Lab 13? 😞