ICT286
Web and Mobile Computing

Topic 8
Accessing MySQL Databases from PHP

Objectives

- Understand the basic concepts of databases and Data Base Management Systems.
- Understand and be able to use SQL language
  - to create, alter and drop tables in a database.
  - to insert, update and delete the contents of a table
  - to make queries to database tables using SELECT commands.
- Be able to use MySQL from the command line.
- Understand and be able to use sequential arrays in PHP
- Understand and be able to use associative arrays in PHP
- Be able to use MySQL from PHP.

Readings

Sebesta: Ch 13.1 – 13.5.

PHP and DBMS

- PHP has strong support for Database Management Systems (DBMSs) interfacing.
- It has library modules (called PHP ***extensions***) to support many of the most popular DBMSs on the market.
- These DBMSs include MySQL, Oracle, Sybase, mSQL, Generic ODBC, and PostgreSQL.

PHP and MySQL

- PHP is most commonly associated with the MySQL DBMS, and used in tandem with it to create dynamic data-driven web sites.

- The close association is mainly due to:
    - Both packages being freely available,
    - Support for both usually comes through the same developer communities, and
    - PHP has large support through its MySQL extensions.

Databases in DBMS

- A database in a DBMS is a place to store data.
- Data in a database is stored in tables.
- Each table should contain information about *one and only one thing*.
- For example, you may have a table of products, a table of customers, or a table of users.
- A table is are also called an *entity*.
- These table based databases are called relational databases.

Tables in a Database

- A table in a database has a number of *rows*.
- Each row contains data for one occurrence of the entity.
- For example, in a CUSTOMER table, each row would contain all the information about *one* customer.
- Each row in a database table is also called a record.

Tables in a Database

Example: Customer Table

Number	FName	LName	Phone
34987	Mary	Tan	9355 7621
76231	Felicity	Smith	9332 6652
98232	Jung	Pradhan	9353 8745
64593	Phoebe	Murphy	9330 6649

Tables in a Database

- Each row in a database table is composed of one or more columns.
- Each column is a single piece of information that you want to store about the entity.
- In the example on the previous slide, the columns are customer `Number`, customer `FName`, customer `LName` and customer `Phone` number.
- Each column is also called a field or an attribute.

Tables in a Database

- This unit will not cover the design of databases (especially normalisation), as this will take too long.
- However, if you find when you create a table that you want to store multiple occurrences of a field in the same table, then you should create another table.
- For example, if you are storing information about customers and orders, there could be multiple orders for each customer (duplicating the customer's information).
- When this occurs, the information about each order should be stored in a separate order table, not in the customer table.

Tables in a Database

- Each column in a database table needs to be given a name and a data type.
- Depending on the data type, you may also need to specify a length (size) for the column.
- Different DBMSs allow you to have different data types. We will discuss the possibilities in MySQL later.

Indexing

- The user can request the DBMS to create an index for a column (or a combination of columns).
- An index is like the index at the end of a book. It allows us to quickly find the pages containing the desired content in the book.
- The DBMS would create a file containing the index mapping each each column value to the address of the corresponding row or rows.
- An index is also called a key for the database table.
- By creating an index, the DBMS can quickly retrieve a single row or group of rows from the table using its key.

Primary Key

- In general, though not necessarily, each table in a database would have a column, or a combination of columns, whose value uniquely identifies a row of the table.
- For example, in the CUSTOMER table, the the customer `Number` would uniquely identifies each customer.
- Most database queries would involve the values in such a column or columns.
- To speed up the database queries, we should create an index for those column or columns.

Primary Key

- When creating a table, we can define those column or columns as the primary key of the table.
- The DBMS would automatically create an index for the column (or columns).
- Database queries would be sped up substantially due to the index.
- However, we must make sure that for the column designated as primary key, its value for each row in the table is always present (not NULL) and is unique in the table.

SQL Language

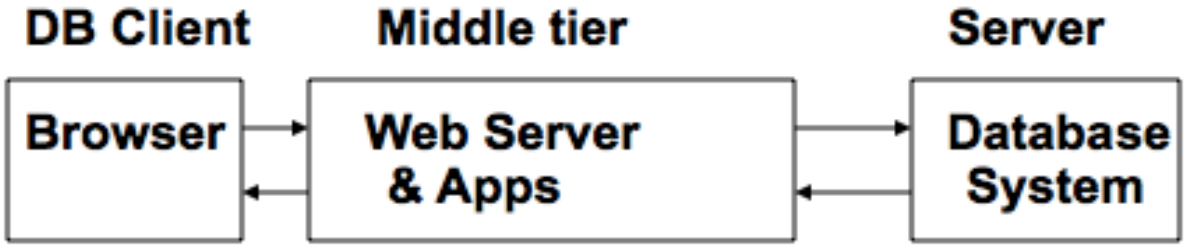
- We use SQL (Structured Query Language) to create, query and modify relational databases.
- There is a standard syntax for SQL, and most DBMSs adhere to this standard. There are usually small variations, usually extensions, for the syntax between different DBMSs, but these are very minor.
- SQL reserved words are case insensitive.
- In this topic, we will only discuss CREATE TABLE, ALTER TABLE, DROP TABLE, SELECT, INSERT, UPDATE and DELETE commands.

Database Access

- Client-Server architecture.
- Client tasks:
  - Provide a way for users to submit queries
  - Run applications that use the results of queries
  - Display the results of queries
- Server tasks:
  - Implement a data manipulation language, e.g., SQL, which can directly access and update the databases

Web-Based Database Access

- For web-based database access, the web client is not directly connected to the database server. Instead, the web client is connected to the web server and the web server is connected with the database server.
- The web server runs the applications (e.g., PHP scripts) and the web clients get the results from the web server.



SQL In MySQL

- In this topic, we will look at SQL in MySQL to perform the basic operations.
- This is only a small subset of the things you can do in MySQL, but they are the things that you will use most frequently.
- First we need to get into MySQL...

Accessing MySQL On Ceto Server

- You have each been created a MySQL account on the MySQL DBMS running on ceto.
- For convenience, your account userid, password and database name are all the same: the letter X followed by your student number. For instance, if your student number is 12345678, then your userid, password and the database name are all X12345678.
- You will need to use a secure shell (like putty or ssh) to login to ceto first and from there you can access your MySQL account.

Accessing MySQL From Linux Terminal

- After you have logged into ceto, at the shell prompt, start MySQL client by typing the following command:

  `mysql –u` *`userid`* `–p`

  … you will be prompted for your MySQL password.

- To use the database that has been created for you, type:

  `use` *`databasename`*`;`

  Note: each command typed from MySQL prompt must end with semicolon “;”.

Creating A Table In MySQL

- A simplified version of the general form of the SQL command to create a table is:

```
CREATE TABLE tablename (attributeName
        datatype, ... );
```

- Example:

```
CREATE TABLE Customer (
    CustNo int,
    CustName varchar(255)
);
```

View Tables In MySQL

- To display a list of tables in your current database (don't forget ";" at the end):

```
SHOW TABLES;
```

- To see the description of a table (columns):

```
DESCRIBE tablename;
```

Character Data Types

Data Type	Example	Description
char	char(4)	A fixed length character field; maximum 255 characters. The example defines a string of length 4.
varchar	varchar(10)	A variable length character field; maximum 255 characters. The example defines a string up to length 10.
text	text	A variable length character field; maximum 65,535 characters.

Numeric Data Types

Data Type	Example	Description
int integer	int(4)	A non decimal number. In this example, the integer is displayed in 4 digits, padding with leading spaces may be needed.
float	float(5,2)	A decimal number. In this example, the total display length is 5 characters including 2 digits after the decimal point.

Other Data Types

Data Type	Example	Description
date	date	A date in the format YYYY-MM-DD.
blob	blob	Binary data storage.

There are many other data types available in MySQL. You can research these from the textbook or the Web

Defining Primary Key

- You can define a primary key for a table while creating the table or else add it later on.
- It is best to define the structure of the database first before adding any data. Otherwise there can be problems if the data does not conform to the changes you want to make. For example, you cannot make a field a primary key if there are any NULL values, or non-unique values.

Defining Primary Key

- To create a primary key whilst creating the table use:

```
CREATE TABLE Customer (
     CustNo int PRIMARY KEY,
     CustName varchar(255)

);
```

Or

```
CREATE TABLE Customer (
     CustNo int,
     CustName varchar(255),
     PRIMARY KEY (CustNo)
);
```

Alter Tables

- The ALTER TABLE command can be used to change the structure of a table, including adding a primary key. For example:

```
ALTER TABLE Customer
    ADD PRIMARY KEY (CustNo);
```

Other `ALTER TABLE` Commands

CLAUSE	Description
ADD COLUMN	Adds a new column to the end of the table.
DROP COLUMN	Removes a column from a table including all of the data.
RENAME AS	Changes the name of the table
ADD INDEX	Adds a new index on a field(s) of the table.
DROP INDEX	Removes an existing index.

The general form of the `ALTER TABLE` statement is:

`ALTER TABLE` *`tablename clause`*`;`

Dropping Tables

- To delete a table and all its data use the `DROP TABLE` command:

```
DROP TABLE tablename;
```

Adding Data To Tables

- Now we have created the structure of the table, we can add some data – the records or rows. To do this use the insert command:

```
INSERT INTO tablename
     VALUES (fieldvalue, . . . );
```

- For example:

```
INSERT INTO Customer
     VALUES (1234, 'Hong');
```

Deleting Data From Tables

- To delete data from a table use the delete command:

```
DELETE FROM tablename WHERE condition;
```

Note: if you do not specify a `WHERE` clause, All data in the table will be deleted, but the table will still exist.

- For example:

```
DELETE FROM Customer
      WHERE CustNo = 1234;
```

Updating Tables

- To update content of a table use the update command:

```
UPDATE tablename
SET col_name1 = value1,
. . . . . .
SET col_namen = valuen
WHERE condition;
```

  - The WHERE clause select all rows that satisfy the condition using the primary key.
  - The SET clause change the values of those columns of the selected rows.

Updating Tables

- Example

```
UPDATE Customer
SET CustName = 'John'
WHERE CustNo = 1234;
```

Making Queries

- We store information in a database so that we can get the information out at a later date. To do this we use the `SELECT` command. The general format of the `SELECT` command is:

```
SELECT fieldname, …
    FROM tablename, …
        WHERE condition;
```

Example SELECT Statements

- The next few slides contain some examples of the kinds of things that can be done with select. They are by no means exhaustive. You can research the select statement in the textbook or the Web.

Example SELECT Statements

```
SELECT * FROM Customer;
```

Selects all the columns and all the rows in the table `Customer`.

Example SELECT Statements

```
SELECT CustName
   FROM Customer
      WHERE CustNo = 1234;
```

Selects the Customer name for the customer with Number `1234` (if it exists) in the table `Customer`.

Example SELECT Statements

```
SELECT CustName
   FROM Customer
     WHERE CustNo > 5000;
```

Selects the names of those customers whose customer number is greater than `5000` (if any exists) in the table `Customer`.

Example SELECT Statements

```
SELECT CustNo
   FROM Customer
      WHERE CustName IS NULL;
```

Selects the Customer number for all customers with names that are empty in the table `Customer`.

Example SELECT Statements

```
SELECT CustNo
   FROM Customer
      WHERE CustName LIKE 'C%';
```

Selects the Customer number for all customers with names that start with `C` in the table `Customer`.

Here ‘C%’ defines a string pattern. The symbol % is the wildcard.

Example SELECT Statements

```
SELECT CustNo
   FROM Customer
      WHERE (CustName LIKE 'C%')
       OR (CustName LIKE 'B%');
```

Selects the Customer number for all customers with names that start with *C* or *B* in the table Customer.

Example SELECT Statements

```
SELECT CustNo, CustName
   FROM Customer
      WHERE (CustName LIKE 'C%')
         AND (CustNo < 2000);
```

Selects the Customer number and Customer name for all customers with names that start with `C` and numbers that are less than `2000`.

Where To From Here…

- This is only the beginning of the possibilities with the select statement. It is also only the beginning of what is possible with databases.
- These basics should give you enough to complete the second assignment.
- When you do ICT285 you will learn much more about these topics.

Access MySQL In PHP

- Now you can create and populate tables at the command line in MySQL, we will see how to do the same thing in PHP.

Connect To MySQL Server

- Firstly you need to connect to the MySQL server from your PHP script.
- To do so, you need the host name of the server (which is "localhost" as the MySQL server runs on the ceto server), your MySQL account name and password, and the database name. Eg,

```
$host = "localhost";
$user = "X12345678";
$password = "X12345678";
$dbname = "X12345678"
```

Connect To MySQL Server

- To connect to the MySQL server from within PHP you may use the `mysqli_connect` command. For example:

```
<?php
$dbc = mysqli_connect($host, $user,
                      $password, $dbname);
// check connection
if (mysqli_connect_errno()) {
   echo "Failed to connect to MySQL"
        . mysqli_connect_error();
}
?>
```

Select The Database

- To select or change the database (same as the `USE` command from the command line) use the `mysqli_select_db` command. For example:

```
$dbname = "X12345678";

mysqli_select_db($dbc, $dbname);
```

Error Functions

- It is possible that you may encounter errors to the database operations. The following functions can be used to code for these cases.
- `@` in front of a function will suppress any error messages that would be generated.
- `die ('error message');` will display the message in the string and stop the script.

Error Functions

- `mysqli_error()` will display a descriptive message about the error that occurred.
- `mysqli_errno()` will display the error number for the error that occurred.

Example:

```
@mysqli_select_db($dbc, $dbname) OR
    die ('Cannot connect to database '
        . mysqli_error($dbc));
```

PHP Arrays

- A PHP array is a generalisation of the arrays of other languages.
- A PHP array is really a mapping of keys to values, where the keys can be numbers (like traditional arrays) or strings (associative arrays).
- There are two types of arrays; sequential arrays and associative arrays. We will consider sequential arrays first.
- A sequential array is like a traditional array. Its keys are indices.
- An associated array is accessed using its keys.

Sequential Arrays

- You create a sequential array using the array() function. For example:

```
$students=array('Mary', 'Jim', 'Felicity');
```

- This is equivalent to:

```
$students[]='Mary';
$students[]='Jim';
$students[]='Felicity';
```

- When assigning a value like the above, the value is added to the *end* of the array.

Sequential Arrays

- To access elements of an array, use the index. Index starts at 0. For example:

```
$scores=array(75, 50, 100, 25);
$scores[]=50;
$average=($scores[0]+$scores[1]
    +$scores[2]+$scores[3]+$scores[4])/5;
```

Sequential Arrays

- You can set index values explicitly using the => operator. For example:

```
$scores=array(1=>75, 3=>100, 2=>60);
$scores[]=50;
```

In the above example, we will add value to the end of the array; i.e., element at index 4 (`$scores[4]`).

Sequential Arrays

- Often you will want to loop through the values of an array. For example:

```
$scores=array(75, 50, 100, 25, 50);
$average=0;
for ($i=0; $i<count($scores); $i++) {
    $average=$average + $scores[$i]
}
$average=$average / count($scores);
```

Sequential Arrays

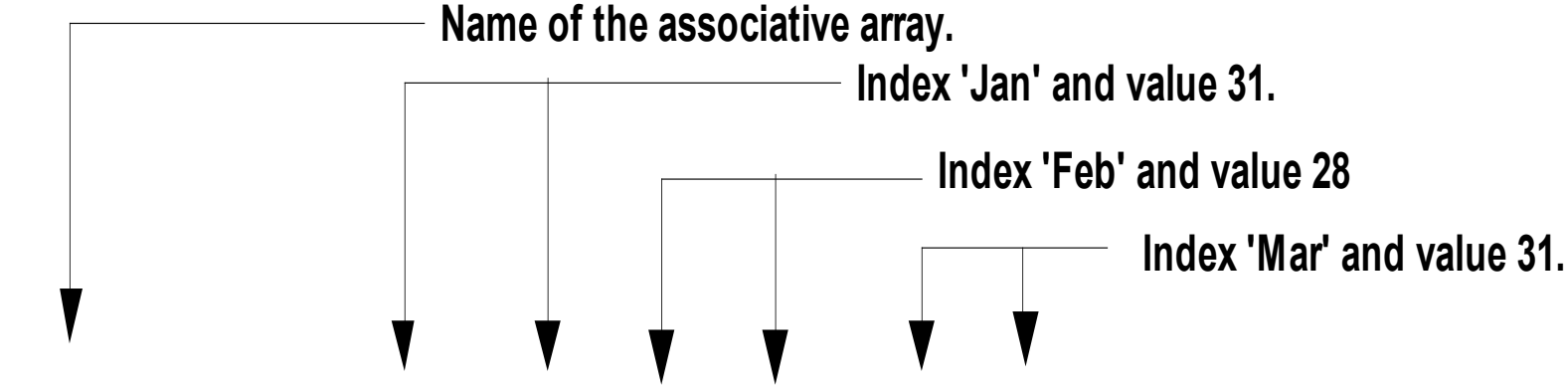
- There is also a foreach loop construct that you can use. For example:

```
$scores=array(75, 50, 100, 25, 50);
$average=0;
foreach ($scores as $item) {
  $average=$average + $item
}
$average=$average / count($scores);
```

In the above loop, the next array element from `$scores` is assigned to variable `$item`.

Associative Arrays

- An associative array is like a sequential array except that the indices (keys) do not have to be sequential numbers. They can be string values. For example:




```
$months = array( 'Jan'=>31, 'Feb'=>28, 'Mar'=>31, 'Apr'=>30,
           'May'=>31, 'Jun'=>30, 'Jul'=>31, 'Aug'=>31,
           'Sep'=>30, 'Oct'=>31, 'Nov'=>30, 'Dec'=>31 );
```

Associative Arrays

- When we want to reference an element of an associative array we use the index. These indices are not restricted to be numerical values.
- In the previous example, the "index" is a string such as “Jan” or “Apr”. These indices are known as the keys.
- For example:

```
$days=$months['Mar'];
```

Will return the number of days in the month of March.

Associative Arrays

- You use a foreach loop to run through the elements of the array. For example:

```
foreach ($months as $key => $value) {
  … process $key and/or $value
}
```

- To add or change a value in an associative array use the key. For example (in a leap year):

```
$months['Feb']=29;
$months['NewMonth']=20;
```

Associative Arrays

- To delete an element from an associative array use the `unset` function. For example:

```
unset($months['NewMonth']);
```

- To check if a key exists in an associative array, use the `isset` function. For example:

```
if (isset($months['Jan']) {
    ...
}
```

Associative Arrays

- To sort an associative array there are two functions;
  - `asort` sorts the array based on the values;
  - `ksort` sorts the array based on the key.;
  - In both cases the relationships between the key and value are maintained.
- For example:

```
asort($months);

ksort($months);
```

Array Functions

There are various functions you can use with arrays:

Function	Description
array_shift()	Removes an item from the beginning of the array.
array_unshift()	Adds an item to the beginning of the array.
array_pop()	Removes an item from the end of the array.
array_push()	Adds an item to the end of the array.

Array Functions

- For example:

```
$scores=array(75, 50, 100, 25, 50);
$first=array_shift($scores);
```

The array $scores would contain 50, 100, 25, and 50.

- The following statement adds 25 to the beginning of the array:

```
array_unshift($scores, 25);
```

More Array Functions

Function	Description
max()	Returns the maximum value of the array.
min()	Returns the minimum value of the array.
array_sum()	Sums the numerical values of the array.
sort()	Sorts the elements of the array.

Array Functions

- For example:

```
$scores=array(75, 50, 100, 25, 50);
$biggest=max($scores);
$smallest=min($scores);
$total=array_sum($scores);
sort($scores);
```

Array Functions

- Another example:

```
$scores=array("2 nights", "5 days",
                    100, "2 more");
$total=array_sum($scores);
```

The variable `$total` contains the value 109 – recall string coercion from Topic 7.

- There are other array functions you can investigate for yourself.

Predefined Arrays

- PHP has a set of available predefined arrays containing variables from the web server (if applicable), the environment, and user input.
- These predefined arrays are associative arrays.
- We have already used some of these arrays:

```
$_GET

$_POST
```

Example Predefined Arrays

Variable Name	Description
`$_COOKIE['lastaccess']`	Cookie with name ‘lastaccess’ - same as a previous example
`$_SESSION['sample_hidden']`	The session variable $sample_hidden.
`$_SERVER['SERVER_NAME']`	The name of the web server running the PHP script.
`$_SERVER['SCRIPT_NAME']`	The name of the current PHP script.
`$_GET['MyName']`	The value of a parameter called “MyName”, passed from a GET request
`$_POST['MyName']`	The value of a parameter called “MyName”, passed from a POST
`$_ENV['LOGNAME']`	The ‘LOGNAME’ environment variable.

Running A Query

- The `mysqli_query` command can be used to run an SQL query.
- Example 1:

```
$query = "SELECT * FROM Customer";
$result = mysqli_query($dbc, $query);
```

- Example 2:

```
$name = "Hong";
$query = "SELECT * FROM Customer WHERE
  CustName = '$name'";
$result = mysqli_query($dbc, $query);
```

Running A Query

- If the query is an INSERT, DELETE or UPDATE, `$result` will contain true or false depending on whether the query worked or not.
  - It can be used to inform the user of the result.
- If the query is a SELECT, `$result` will be a pointer to the resulting data set.

Accessing The Result Of A Query

- After running the `mysqli_query` command, the rows returned can be accessed using a special while loop. Example:

```
while($row=mysqli_fetch_array($result,
                        MYSQLI_ASSOC)){
    // process one row stored in $row
}
```

- Each field of the selected row can be retrieved by using the column name as the key into the associative array `$row`. For example

```
$row['CustNo']
```

Accessing The Result Of A Query

- Alternatively you can use the MYSQLI_NUM constant to get the result array with number indices and then access the columns using indices.
- Example:

```
$query = 'SELECT * FROM Customer';
$result = mysqli_query($dbc, $query);
while($row=mysqli_fetch_array($result,
                    MYSQLI_NUM)){
    // process one row in $row, eg,
    // $row[0] contain the CustNo of each row
    // $row[1] contain the CustName of each row.
}
```

Release Resources

- It is a good practice to close an object and free the resources used by the object when you have finished with them.
- To close the SQL query resource:

```
mysqli_free_result ($result);
```

- To close the connection to MySQL server:

```
mysqli_close($dbc);
```

References

- PHP Tutorial from W3School:

  http://www.w3schools.com/php/default.asp

- MySQL Tutorial from W3Schools:

  http://www.w3schools.com/sql/default.asp

- MySQL Tutorial from MySQL:

  https://dev.mysql.com/doc/mysql-tutorial-excerpt/5.7/en/