PHP & MYSQL

- MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).
- The Structured Query Language (SQL) is a standard language for specifying accesses and modifications to relational databases.
- Database queries: A query is a question or a request.
- We can query a database for specific information and have a recordset returned.
- **mysqli_query()** is the function that executes the SQL queries.
- The **mysqli_error()** function returns the last error description for the most recent function call, if any.

Connecting to server

- Before we can access data in the MySQL database, we need to be able to connect to the server.
- The PHP function **mysql_connect** connects a script to a MySQL server.
- This function takes three parameters, all of which are optional.
- The first is the host that is running MySQL; the default is localhost (the machine on which the script is running).
- The second parameter is the username for MySQL; the default is the username in which the PHP process runs.
- The third parameter is the password for the database; the default is blank

$connection = mysql_connect(*host,username,password*);
e.g: $connection=mysqli_connect("localhost"," root","");
[mysql_connect is deprecated , so use mysqli_connect]

- the connect operation could fail, in which case the value returned would be false (if true it returns a reference to the database). Therefore, the call to mysql_connect usually is used in conjunction with die.
- The connection to a database is terminated with the **mysql_close** function
- **mysql_select_db** function is used to select a particular database on the server.

e.g: $db=mysqli_select_db($connection,*databasename*);

Example: connect.php

```php
<?php
    $username="root";
    $password="";
    $host="localhost";

    $connection=mysqli_connect($host,$username,$password);
    if(!$connection){
        echo "error connecting to server!".mysqli_connect_error();
    }
    else{
        echo "connected to server<br/>";
    }
?>
```

Creating database and table

- The CREATE DATABASE statement is used to create a database in MySQL.
- Query syntax for creating database:

"CREATE DATABASE *database_name*"

Example:

```php
<?php
require_once("connect.php");

$query="CREATE DATABASE license";

$query_run=mysqli_query($connection,$query);

if($query_run){
    echo "database created successfully";
}
else{
    echo mysqli_error($connection);
}

?>
```

- A database table has its own unique name and consists of columns and rows.
- Query syntax for creating a table in the database:

"CREATE TABLE *table_name*(table *column names* and their *attributes)*"

For example the below query on execution creates a table with 3 columns(ID, name and email)

"CREATE TABLE student(id INT(6) AUTO_INCREMENT PRIMARY KEY, name TEXT(20),email VARCHAR(100) NOT NULL);"

Where,

- id, name & email are table student's column names.
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT.
- INT, TEXT, VARCHAR represents data types of values to be stored.
- NOT NULL - Each row must contain a value for that column, null values are not allowed

Each table should have a primary key column. Its value must be unique for each record in the table.

Example:

```php
<?php
require_once("connect.php");

$db=mysqli_select_db($connection,"license");

if($db){
    //echo "database selected succesfully";
    $query="CREATE TABLE applicants (id INT(5) AUTO_INCREMENT PRIMARY KEY,firstname VARCHAR(30)
        NOT NULL,lastname VARCHAR(30),email VARCHAR(50),address VARCHAR(100),blood_group
        VARCHAR(5),gender VARCHAR(5),category VARCHAR(40),photo VARCHAR(300))";

    $query_run =mysqli_query($connection,$query);
    if($query_run){
        echo "table created successfully!";
    }
    else{
        echo mysqli_error($connection);
    }
}
?>
```

Inserting values into table

- The INSERT INTO statement is used to add new records to a MySQL table:
- Syntax:

"INSERT INTO table_name (column1, column2, column3,...)VALUES (value1, value2, value3,...)"

- Rules to remember:
    - The SQL query must be quoted in PHP
    - String values inside the SQL query must be quoted
    - Numeric values must not be quoted

```php
require_once("connect.php");
$db=mysqli_select_db($connection,"license");
if($db){

    $query="INSERT INTO applicants(id,firstname,lastname,email,addres
    s,blood_group,gender,category,photo) VALUES('','$firstname','$
        lastname','$email','$address','$blood','$gender','$c','$
        photoname')";
    if(mysqli_query($connection,$query)){
        echo "insertion successfull";
    }
    else{
        echo mysqli_error($connection);
    }

}
```

Fetching/Retrieving values from database

- The **SELECT** statement is used to select data from one or more tables.
  SELECT column_name(s) FROM table_name;
  SELECT * FROM table_name; // to select all columns use *
- The **mysqli_fetch_array()** function fetches a result row as an associative array, a numeric array, or both.
  Syntax: mysqli_fetch_array(*query_run_result* , MYSQLI_ASSOC)
- The **mysql_fetch_assoc()** function fetches a result row as an associative array.
  Syntax: mysqli_fetch_assoc(*query_run_result*)

Example:

```
<?php
require_once("connect.php");
$db=mysqli_select_db($connection,"license");
if($db){

    $query="SELECT * FROM applicants";
    $query_result=mysqli_query($connection,$query);
    if($query_result){

        while($row=mysqli_fetch_assoc($query_result)){
            $photo=$row['photo'];
            $src="files/".$photo;
?>
            <img src="<?php echo $src ?>" width="100" height="100" />
<?php
            $fname=$row['firstname'];
            $lname=$row['lastname'];
            $email=$row['email'];
            $address=$row['address'];
            $blood=$row['blood_group'];
            $gender=$row['gender'];
            $category=$row['category'];

            echo "<br/>$fname $lname<br/>$email<br/>$address</br>$blood</br>$gender<br/>$
                category<br/><br/>";
        }
    }
}
?>
```

Updating values in database

- The UPDATE statement is used to update existing records in a table.
- Syntax:

  UPDATE table_name SET column1=value, column2=value2,... WHERE some_column=some_value

- The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!
  Example:

```php
<?php

$connection=mysqli_connect("localhost","root","","license");

if($connection){
    $query="UPDATE applicants SET email='a@gmail.com' WHERE id=2";
    if(mysqli_query($connection,$query)){
        echo "update succesfull";
    }
    else{
        echo "update failed";
    }
}
else{
    echo mysqli_error($connection);
}
?>
```

Deleting values from database

- The DELETE statement is used to delete records from a table
- Syntax:

  DELETE FROM table_name WHERE some_column = some_value

- The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Example:

```php
<?php

$connection=mysqli_connect("localhost","root","","license");

if($connection){
    $query="DELETE FROM applicants WHERE id=2";
    if(mysqli_query($connection,$query)){
        echo "delete succesfull";
    }
    else{
        echo "delete failed";
    }
}
else{
    echo mysqli_error($connection);
}
?>
```

Deleting a table and database

- The DROP TABLE statement is used to delete an existing table in a database.
- Syntax:

  DROP TABLE table_name;

Example:

```php
<?php

$connection=mysqli_connect("localhost","root","","license2");

if($connection){
    $query="DROP TABLE applicants";
    if(mysqli_query($connection,$query)){
        echo "table delete succesfull";
    }
    else{
        echo "delete failed";
    }
}
else{
    echo mysqli_error($connection);
}
?>
```

- The DROP DATABASE statement is used to drop an existing SQL database.
- Syntax:

  DROP DATABASE databasename;

Example:

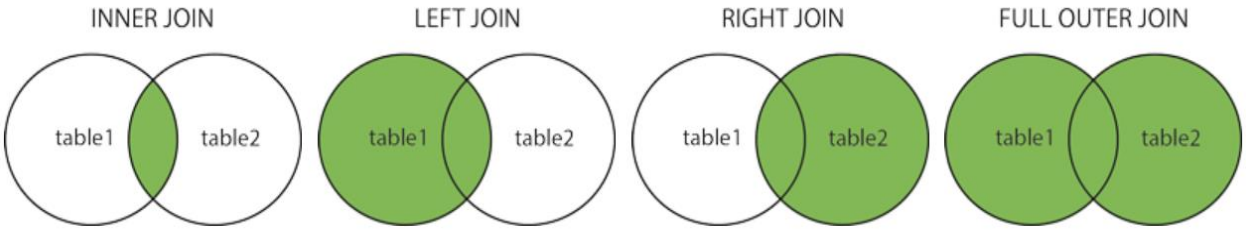
```php
<?php

$connection=mysqli_connect("localhost","root","","license2");

if($connection){
    $query="DROP DATABASE license2";
    if(mysqli_query($connection,$query)){
        echo "database delete succesfull";
    }
    else{
        echo "delete failed";
    }
}
else{
    echo mysqli_error($connection);
}
?>
```

SQL JOINS

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
- The different types of the JOINs in SQL:
  - (INNER) JOIN: Returns records that have matching values in both tables
  - LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
  - RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
  - FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



Example:

Consider two tables users and address

id	name
1	ram
2	laxman
4	sita

id	location
1	ktm
2	pokhara
3	chitwan
5	dharan

1. INNER JOIN
   The INNER JOIN keyword selects records that have matching values in both tables.
   Syntax:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

   Example:

   SQL:

```
SELECT * FROM users INNER JOIN address ON users.id=address.id
```

Output:

id	name	id	location
1	ram	1	ktm
2	laxman	2	pokhara

2. LEFT JOIN
The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.
Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

Example:

SQL:

```
SELECT * FROM users LEFT JOIN address ON users.id=address.id
```

Output:

id	name	id	location
1	ram	1	ktm
2	laxman	2	pokhara
4	sita	*NULL*	*NULL*

3. RIGHT JOIN
The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.
Syntax:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Example:

SQL:

```
SELECT * FROM users RIGHT JOIN address ON users.id=address.id
```

Output:

id	name	id	location
1	ram	1	ktm
2	laxman	2	pokhara
NULL	NULL	3	chitwan
NULL	NULL	5	dharan

4. FULL OUTER JOIN
   - The FULL OUTER JOIN keyword return all records when there is a match in left (table1) or right (table2) table records.
   - FULL OUTER JOIN can potentially return very large result-sets
   - FULL OUTER JOIN and FULL JOIN are the same.

   Syntax:

   SELECT column_name(s)

   FROM table1

   FULL OUTER JOIN table2

   ON table1.column_name = table2.column_name

   Example:

table_A

A	M
1	m
2	n
4	o

table_B

A	N
2	p
3	q
5	r

SQL Code:

```
1  SELECT * FROM table_A
2  FULL OUTER JOIN table_B
3  ON table_A.A=table_B.A;
```

Output:

A	M	A	N
2	n	2	p
1	m	-	-
4	o	-	-
-	-	3	q
-	-	5	r