# IWT UNIT-5 Notes

1. Explain the significance of PHP in interacting with MySQL databases. Provide examples of basic PHP commands used to establish connections to MySQL servers and select databases.

   Answer:

   PHP is a powerful server-side scripting language renowned for its robust integration with MySQL databases, enabling efficient management and manipulation of data. Its significance in interacting with MySQL databases lies in its ability to establish connections, execute queries, and handle data retrieval and manipulation. Here are examples of basic PHP commands for establishing connections and selecting databases:

   **Establishing a Connection to MySQL Server:**

   **Using MySQLi Extension (Procedural Style):**

```
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
} else {
  echo "Connected successfully!";
}
```

   **Using PDO (PHP Data Objects):**

```
$servername = "localhost";
$username = "username";
$password = "password";

try {
  $conn = new PDO("mysql:host=$servername", $username, $password);
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully!";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
```

   **Selecting a MySQL Database:**

**Using MySQLi Extension:**

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "database_name";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
} else {
   echo "Connected to database successfully!";
}
```

Using PDO:

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "database_name";

try {
   $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   echo "Connected to database successfully!";
} catch(PDOException $e) {
   echo "Connection failed: " . $e->getMessage();
}
```

**Explanation:**

- **mysqli_connect():** Establishes a connection to the MySQL server using the MySQL Improved Extension (mysqli).
- **PDO():** Creates a new PDO object to interact with the database using PHP Data Objects (PDO).
- **mysql:host:** Specifies the host (e.g., localhost) and dbname (database name) in the connection string.

**Significance:**

1. **Data Management:** PHP facilitates the execution of SQL queries, enabling data retrieval, insertion, modification, and deletion within MySQL databases.

2. **Dynamic Content Generation:** Enables PHP scripts to dynamically generate content based on retrieved data, allowing for dynamic and interactive web applications.

3. **Secure Database Interaction:** Provides methods to handle database connections securely, preventing SQL injection and ensuring data integrity.

4. **Scalability:** Allows seamless integration with MySQL databases, supporting scalable and robust web applications.

PHP's robust integration with MySQL databases empowers developers to create dynamic and data-driven web applications by facilitating efficient interaction and manipulation of database content.

2. Outline the step-by-step process of creating a new database using PHP. Include relevant PHP code snippets to illustrate the creation of a database within MySQL. or Write a PHP program for creating a table.

Answer :

Creating a new database using PHP involves connecting to a MySQL server and executing SQL commands to create the database. Here's a step-by-step process with PHP code snippets:

Step 1: Establish a connection to MySQL server using PHP's mysqli extension.

```
<?php

$servername = "localhost"; // Replace with your MySQL server name

$username = "your_username"; // Replace with your MySQL username

$password = "your_password"; // Replace with your MySQL password

// Create connection

$conn = new mysqli($servername, $username, $password);

// Check connection

if ($conn->connect_error) {

  die("Connection failed: " . $conn->connect_error);

}
```

Step 2: Create a new database using SQL command (CREATE DATABASE).

```
// SQL command to create a new database

$databaseName = "new_database"; // Replace with your desired database name

$sql = "CREATE DATABASE $databaseName";

// Execute SQL command

if ($conn->query($sql) === TRUE) {

    echo "Database created successfully";

} else {

    echo "Error creating database: " . $conn->error;

}
```

Step 3: Close the database connection.

```
// Close connection

$conn->close();

?>
```

This code establishes a connection to the MySQL server, creates a new database, and then closes the connection. Remember to replace 'your_username', 'your_password', and 'new_database' with your actual MySQL credentials and the desired database name.

Ensure that your PHP environment has the necessary permissions to create databases on the MySQL server. Also, it's crucial to handle user inputs securely to prevent SQL injection attacks by sanitizing and validating inputs before executing SQL queries.

3. Discuss the procedure for selecting a specific database using PHP and MySQL. Provide PHP code demonstrating how to switch between multiple databases.

Answer : PHP, we can select a specific database using MySQL by connecting to the server and then specifying the database we want to work with. Here's the procedure along with PHP code demonstrating how to switch between multiple databases:

Step 1: Establish a connection to the MySQL server using PHP's mysqli extension.

```
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password
$databaseName = "your_database"; // Replace with your default database name

// Create connection
$conn = new mysqli($servername, $username, $password, $databaseName);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
```

Step 2: Switch to another database by using the **mysqli_select_db()** function.

```
$newDatabaseName = "another_database"; // Replace with the database we want to switch to
// Select another database
if(mysqli_select_db($conn, $newDatabaseName)) {
   echo "Switched to $newDatabaseName database successfully";
} else {
   echo "Error switching to $newDatabaseName database: " . mysqli_error($conn);
}
```

This code snippet connects to the MySQL server and initially selects the default database specified in the connection parameters. Then, using the **mysqli_select_db()** function, it switches to another database named **$newDatabaseName**. If successful, it echoes a confirmation message; otherwise, it displays an error message.

Remember to replace **'your_username'**, **'your_password'**, **'your_database'**, and **'another_database'** with wer actual MySQL credentials, default database name, and the name of the database we want to switch to.

After switching to another database, all subsequent queries using **$conn** will be executed within that selected database context until a different database is selected or a new connection is established.

4. Explain how PHP can retrieve and list the available databases hosted on a MySQL server. Offer PHP code examples to showcase the listing of database names.

Answer: To retrieve and list the available databases hosted on a MySQL server using PHP, you can execute a SQL query to fetch this information. MySQL provides a system database called **information_schema**, which contains metadata about the databases on the server. You can query this database to retrieve the list of databases.

Here's an example PHP code that demonstrates how to fetch and list the available databases:

```php
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// SQL query to retrieve database names
$sql = "SHOW DATABASES";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
   // Output data of each row
   while ($row = $result->fetch_assoc()) {
      echo $row["Database"] . "<br>";
   }
} else {
   echo "No databases found";
}

// Close connection
$conn->close();
?>
```

This PHP code connects to the MySQL server and executes the **SHOW DATABASES** query. It fetches the list of databases available on the server and then iterates through the result set to display the database names.

Replace **'your_username'** and **'your_password'** with our actual MySQL credentials. When we run this PHP script, it will output a list of available databases hosted on our MySQL server.

The user we are connecting with should have the necessary permissions to view the list of databases (**SHOW DATABASES** privilege) on the MySQL server.

5. Detail the process of retrieving and displaying the names of tables within a MySQL database using PHP. Provide code snippets demonstrating how PHP fetches table names.

Answer: To retrieve and display the names of tables within a MySQL database using PHP, you can execute a SQL query against the **information_schema** database. The **information_schema** contains metadata about tables, among other things, on the MySQL server.

Here's an example PHP code that demonstrates how to fetch and display the names of tables within a specific MySQL database:

```
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password
$databaseName = "your_database"; // Replace with your specific database name

// Create connection
$conn = new mysqli($servername, $username, $password, $databaseName);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// SQL query to retrieve table names
$sql = "SHOW TABLES";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // Output data of each row
  while ($row = $result->fetch_row()) {
    echo $row[0] . "<br>"; // The table name is in the first column (index 0)
  }
} else {
  echo "No tables found";
}

// Close connection
$conn->close();
?>
```

This PHP code connects to the specified MySQL database and executes the **SHOW TABLES** query. It fetches the list of tables within that database and then iterates through the result set to display the table names.

Replace **'your_username'**, **'your_password'**, and **'your_database'** with your actual MySQL credentials and the name of the database you want to retrieve table names from.

When you run this PHP script, it will output a list of table names within the specified MySQL database. Ensure that the user you're connecting with has the necessary permissions to access and list tables within the specified database.

6. Elaborate on the PHP-driven creation of tables within a MySQL database. Present examples that show PHP's role in executing SQL queries to create tables.

Answer: PHP can be used to execute SQL queries that create tables within a MySQL database. Here's a step-by-step process along with PHP code examples demonstrating the creation of tables:

Step 1: Connect to the MySQL server using PHP's mysqli extension.

```
<?php

$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password
$databaseName = "your_database"; // Replace with your specific database name

// Create connection
$conn = new mysqli($servername, $username, $password, $databaseName);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
```

Step 2: Execute SQL queries to create tables within the specified database.

Here's an example creating a simple table:

```
// SQL query to create a table
$sql = "CREATE TABLE users (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";

// Execute SQL query to create the table
if ($conn->query($sql) === TRUE) {
    echo "Table 'users' created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}
```

This code creates a table named 'users' with columns for **id**, **firstname**, **lastname**, **email**, and **reg_date**.

We have to replace **'your_username'**, **'your_password'**, **'your_database'** with our actual MySQL credentials and the name of the database we want to create tables in.

Ensure that the user account used to connect to the database has the necessary permissions to create tables. Also, always validate and sanitize user inputs to prevent SQL injection attacks when dynamically creating tables based on user input.

After executing this PHP script, the 'users' table will be created within the specified MySQL database. Adjust the table structure and columns as per your specific requirements.

7. Illustrate the insertion of data into MySQL tables using PHP scripts. Include PHP code exemplifying the insertion of records into specific tables.

Answer: Inserting data into MySQL tables using PHP involves executing SQL queries that insert records into the specified tables. Here's an example of how to insert records into a MySQL table using PHP:

Step 1: Connect to the MySQL server using PHP's mysqli extension.

```
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
```

```
$password = "your_password"; // Replace with your MySQL password
$databaseName = "your_database"; // Replace with your specific database name

// Create connection
$conn = new mysqli($servername, $username, $password, $databaseName);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
```

Step 2: Execute SQL queries to insert records into the specified table.

Here's an example of inserting data into the 'users' table:

```
// Sample data to be inserted
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";


// SQL query to insert data into the 'users' table
$sql = "INSERT INTO users (firstname, lastname, email)
    VALUES ('$firstname', '$lastname', '$email')";


// Execute SQL query to insert data
if ($conn->query($sql) === TRUE) {
  echo "New record inserted successfully";
} else {
  echo "Error inserting record: " . $conn->error;
}
```

This code snippet inserts a new record into the 'users' table with the specified values for **firstname**, **lastname**, and **email**. Replace **'your_username'**, **'your_password'**, **'your_database'** with your actual MySQL credentials and the name of the database you want to insert records into.

Always sanitize and validate user inputs before executing SQL queries to prevent SQL injection attacks.

After executing this PHP script, a new record will be inserted into the 'users' table with the provided data. Adjust the table name, column names, and values to match your specific table structure and data requirements.

8. Discuss the alteration of existing tables within a MySQL database using PHP. Provide examples showcasing PHP's capability to modify table structures through SQL queries.

Answer:

PHP can execute SQL queries to alter existing tables within a MySQL database, allowing modifications to table structures such as adding columns, modifying column definitions, or changing table constraints. Here are examples demonstrating how to alter tables using PHP:

Step 1: Connect to the MySQL server using PHP's mysqli extension.

```
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password
$databaseName = "your_database"; // Replace with your specific database name

// Create connection
$conn = new mysqli($servername, $username, $password, $databaseName);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
```

Step 2: Execute SQL queries to alter the structure of an existing table.

Here are a few examples:

**Example 1: Adding a new column to an existing table**

```
// SQL query to add a new column 'age' to the 'users' table
$sql = "ALTER TABLE users ADD COLUMN age INT(3)";

// Execute SQL query to add a new column
if ($conn->query($sql) === TRUE) {
   echo "Column 'age' added successfully";
} else {
   echo "Error adding column: " . $conn->error;
}
```

**Example 2: Modifying a column in an existing table**

```
// SQL query to modify the 'email' column in the 'users' table
$sql = "ALTER TABLE users MODIFY COLUMN email VARCHAR(100) NOT NULL";


// Execute SQL query to modify the column
if ($conn->query($sql) === TRUE) {
    echo "Column 'email' modified successfully";
} else {
    echo "Error modifying column: " . $conn->error;
}
```

**Example 3: Dropping a column from an existing table**

```
// SQL query to drop the 'age' column from the 'users' table
$sql = "ALTER TABLE users DROP COLUMN age";

// Execute SQL query to drop the column
if ($conn->query($sql) === TRUE) {
    echo "Column 'age' dropped successfully";
} else {
    echo "Error dropping column: " . $conn->error;
}
```

Replace **'your_username'**, **'your_password'**, **'your_database'** with our actual MySQL credentials and the name of the database containing the table we want to alter.

We have to be cautious while altering tables as it can affect existing data. Test any alterations thoroughly, especially on production databases.

9. Explain the role of PHP in executing MySQL queries. Discuss various types of queries (e.g., SELECT, INSERT, UPDATE, DELETE) and provide PHP code snippets for each query type.

Answer:

PHP serves as a bridge between your web application and the MySQL database, allowing you to interact with the database by executing SQL queries. It offers various functions and methods through extensions like mysqli or PDO (PHP Data Objects) to perform SELECT, INSERT, UPDATE, DELETE, and other SQL operations. Here are examples of different query types using PHP:

**SELECT Query:**

```
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password
$databaseName = "your_database"; // Replace with your specific database name

// Create connection
$conn = new mysqli($servername, $username, $password, $databaseName);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// SQL query to fetch data from a table ('users' table in this example)
$sql = "SELECT * FROM users";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // Output data of each row
  while ($row = $result->fetch_assoc()) {
    // Access row data, for example:
    echo "ID: " . $row["id"] . " - Name: " . $row["name"] . "<br>";
  }
} else {
  echo "No records found";
}

// Close connection
$conn->close();
?>
```

**INSERT Query:**

```
<?php
// Assuming a connection is already established (as shown in previous examples)

// Sample data to be inserted
$name = "John Doe";
$email = "john@example.com";

// SQL query to insert data into the 'users' table
$sql = "INSERT INTO users (name, email) VALUES ('$name', '$email')";

if ($conn->query($sql) === TRUE) {
  echo "New record inserted successfully";
} else {
    echo "Error inserting record: " . $conn->error;
}
?>
```

**UPDATE Query:**

```
<?php
// Assuming a connection is already established (as shown in previous examples)

// Sample update data
$newName = "Jane Doe";
$idToUpdate = 1;

// SQL query to update data in the 'users' table
$sql = "UPDATE users SET name='$newName' WHERE id=$idToUpdate";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}
?>
```

**DELETE Query:**

```
 <?php
// Assuming a connection is already established (as shown in previous examples)

$idToDelete = 2;

// SQL query to delete data from the 'users' table
$sql = "DELETE FROM users WHERE id=$idToDelete";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}
?>
```

Replace **'your_username'**, **'your_password'**, **'your_database'**, and adjust table and column names according to our actual database structure. Always validate and sanitize user inputs to prevent SQL injection attacks when using dynamic data in queries.

10. Describe the process of deleting databases, tables, and data records from MySQL using PHP. Include PHP code demonstrating how to delete databases, tables, and specific data entries.

Answer: Deleting databases, tables, and specific data records from MySQL using PHP involves executing SQL queries to perform these operations. Here are examples demonstrating how to delete databases, tables, and data entries:

**Deleting a Database:**

```
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// SQL query to drop/delete a database
$databaseName = "database_to_delete"; // Replace with the database name to be deleted
$sql = "DROP DATABASE $databaseName";

if ($conn->query($sql) === TRUE) {
   echo "Database $databaseName deleted successfully";
} else {
   echo "Error deleting database: " . $conn->error;
}

// Close connection
$conn->close();
?>
```

**Deleting a Table:**

```
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password
$databaseName = "your_database"; // Replace with your specific database name

// Create connection
$conn = new mysqli($servername, $username, $password, $databaseName);
```

```
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// SQL query to drop/delete a table
$tableName = "table_to_delete"; // Replace with the table name to be deleted
$sql = "DROP TABLE $tableName";

if ($conn->query($sql) === TRUE) {
   echo "Table $tableName deleted successfully";
} else {
   echo "Error deleting table: " . $conn->error;
}

// Close connection
$conn->close();
?>
```

**Deleting Specific Data Records:**

```
<?php
$servername = "localhost"; // Replace with your MySQL server name
$username = "your_username"; // Replace with your MySQL username
$password = "your_password"; // Replace with your MySQL password
$databaseName = "your_database"; // Replace with your specific database name

// Create connection
$conn = new mysqli($servername, $username, $password, $databaseName);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// SQL query to delete specific data from a table
$idToDelete = 1; // Replace with the record ID to be deleted
$sql = "DELETE FROM your_table WHERE id = $idToDelete";

if ($conn->query($sql) === TRUE) {
   echo "Record with ID $idToDelete deleted successfully";
} else {
   echo "Error deleting record: " . $conn->error;
}
```

```
// Close connection
$conn->close();
?>
```

Replace **'your_username'**, **'your_password'**, **'database_to_delete'**, **'table_to_delete'**, **'your_database'**, **'your_table'**, and adjust table and column names according to your actual database structure.

We have to be cautious when performing deletion operations as they are irreversible. Always ensure we have the necessary permissions and double-check before deleting any data or structures in a production environment.

## 11. What is File ? Explain how file is created in PHP?

Answer:

In PHP, a file is a resource used to store data persistently on a storage medium, such as a hard disk or a similar storage device. It can contain various types of data, such as text, images, code, or any other information.

Creating a file in PHP involves a few steps:

**Using fopen() to Create a File:**

The **fopen()** function in PHP is used to open a file or create a new one if it doesn't exist. It returns a file pointer/resource if successful.

Here's an example of how you can create a new file using **fopen()**:

```
<?php
$filename = "newfile.txt"; // Replace with your desired file name

// 'w' mode indicates writing; creates the file if it doesn't exist
$file = fopen($filename, "w") or die("Unable to create file!");

echo "File '$filename' created successfully.";

// Remember to close the file pointer when done
fclose($file);
?>
```

**Modes for Opening Files:**

- **w**: Opens the file for writing. If the file doesn't exist, it creates it. If the file exists, it truncates it (removes the contents).
- **a**: Opens the file for writing. If the file doesn't exist, it creates it. If the file exists, it appends the data to the end of the file.
- **x**: Creates a new file for writing. Returns **false** and an error if the file already exists.
- **r+**: Opens the file for reading and writing. Starts at the beginning of the file.
- **a+**: Opens the file for reading and writing. If the file doesn't exist, it creates it. If it exists, it appends data.
- **x+**: Creates a new file for reading and writing. Returns **false** and an error if the file already exists.

  When creating or working with files in PHP, it's important to consider file permissions, error handling, and closing the file pointer (**fclose($file)**) after the operations are complete to free up resources and ensure data integrity.