Unit 6: Database and PHP

Introduction to database

A database is a structured collection of data that is organized in a way that allows for efficient storage, retrieval, and manipulation of data. Databases are used to store large amounts of information that can be accessed, managed, and updated by multiple users and applications.

Databases can be organized in different ways, but most databases use a relational model, which organizes data into tables or relations. Each table consists of a set of rows or records, with each row containing a set of columns or fields that represent specific types of data. Tables can be related to each other through keys or references, which allow for complex queries and relationships to be established between tables.

Create, Retrieve, Update and Delete operation in database

There are the SQL commands that correspond to the CRUD operations:

Create:

To insert a new row of data into a table in a database, you can use the SQL INSERT command. The basic syntax for the command is as follows:

INSERT INTO table_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

For example, to insert a new customer into a customer table in a database, you could use the following SQL command:

```
INSERT INTO customer (name, email, phone)
VALUES ('John Doe', 'johndoe@example.com', '555-1234');
```

Select:

To select one or more rows of data from a table in a database, you can use the SQL SELECT command. The basic syntax for the command is as follows:

SELECT column1, column2, column3, ... FROM table_name; For example, to retrieve all customers from a customer table in a database, you could use the following SQL command:

SELECT * FROM customer;

Update:

To modify existing data in a database, you can use the SQL UPDATE command. The basic syntax for the command is as follows:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

For example, to update the email address of a customer in a customer table in a database, you could use the following SQL command:

```
UPDATE customer SET email = 'newemail@example.com' WHERE name = 'John Doe';
```

Delete:

To delete one or more rows of data from a table in a database, you can use the SQL

DELETE command. The basic syntax for the command is as follows:

DELETE FROM table_name WHERE condition;

For example, to delete a customer from a customer table in a database, you could use the following SQL command:

```
DELETE FROM customer WHERE name = 'John Doe'
```

These SQL commands are commonly used for performing CRUD operations on a database. They can be executed using a variety of tools and interfaces, including commandline interfaces, graphical user interfaces, and programming languages with database A.

Connecting to database through PHP mysqli_connect()

In PHP, you can connect to a database using the mysqli_connect() function. This function establishes a new connection to a MySQL database server and returns a mysqli object that can be used to execute queries and interact with the database. Here's an example of how to connect to a database using mysqli_connect():

```
<?php
// Database configuration
$host = "localhost";
$username = "root";
$password = "password";
$dbname = "mydatabase";
// Connect to the database
$conn = mysqli_connect($host, $username, $password, $dbname);
// Check if the connection was successful
if (!$conn) {
 die("Connection failed: " . mysqli_connect_error());
}
// Connection successful, do something with the database
// ...
// Close the connection
```

```
mysqli_close($conn);
?>
```

In this example, the mysqli_connect() function is used to establish a connection to a MySQL database running on the local machine. The host, username, password, and dbname variables are used to specify the details of the database server and the database to connect to.

Executing Queries with mysqli_query()

In PHP, you can execute SQL queries on a database using the mysqli_query() function. This function is used to execute SQL statements, such as SELECT, INSERT, UPDATE, and DELETE statements, on a MySQL database. Here's an example of how to use mysqli_query() to execute a SELECT statement and fetch the results:

```
<?php
// Connect to the database
$conn = mysqli_connect("localhost", "root", "password",
"mydatabase");
// Check if the connection was successful
if (!$conn) {
 die("Connection failed: " . mysqli_connect_error());
}
// Execute a SELECT statement and fetch the results
$sql = "SELECT * FROM customers";
$result = mysqli_query($conn, $sql);
// Check if the query was successful
if (!$result) {
 die("Error executing query: " . mysqli_error($conn));
}
// Process the results
while ($row = mysqli_fetch_assoc($result)) {
 echo "Name: " . $row["name"] . ", Email: " . $row["email"] . "<br>";
}
// Free up the result set
mysqli_free_result($result);
// Close the connection
mysqli_close($conn);  ?>
```

In this example, the mysqli_query() function is used to execute a SELECT statement that retrieves all rows from a customers table in a database. The $conn variable holds the database connection object returned by mysqli_connect().

Create a PHP file (e.g., insert_student.php) with the following code:

```
<?php
// Replace with your database connection details
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "your_database_name";
// Create a connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check the connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}
// Prepare and bind
$stmt = $conn->prepare("INSERT INTO student (id, first_name,
last_name, age, grade_level) VALUES (?, ?, ?, ?, ?)");
$stmt->bind_param("isssi", $id, $first_name, $last_name, $age,
$grade_level);
// Set the values and execute
$id = 1;
$first_name = "John";
$last_name = "Doe";
$age = 18;
$grade_level = "12th";
if ($stmt->execute()) {
 echo "New record created successfully";
} else {
 echo "Error: " . $stmt->error;
}     // Close the statement and the connection
```

```
$stmt->close();
$conn->close();
?>
```

Fetching data with mysqli_fetch_assoc() and mysqli_fetch_array()

mysqli_fetch_assoc() and mysqli_fetch_array() are used to fetch data from the result set returned by a SELECT query in a MySQL database using the MySQLi extension in PHP. mysqli_fetch_assoc() fetches a result row as an associative array, while mysqli_fetch_array() can fetch a result row as an associative array, a numeric array, or both. Here's an example using both mysqli_fetch_assoc() and mysqli_fetch_array():

Create a PHP file (e.g., fetch_data.php) with the following code:

```
<?php
// Replace with your database connection details
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "your_database_name";
// Create a connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check the connection
if ($conn->connect_error) {
 die("Connection failed: " . $conn->connect_error);
}
// Execute a SELECT query
$sql = "SELECT id, first_name, last_name, age, grade_level FROM student";
$result = $conn->query($sql);
// Using mysqli_fetch_assoc()
if ($result->num_rows > 0) {
 echo "<h3>Using mysqli_fetch_assoc()</h3>";
 while($row = mysqli_fetch_assoc($result)) {
 echo "ID: " . $row["id"] . " - Name: " . $row["first_name"] . " " .
```

```
$row["last_name"] . " - Age: " . $row["age"] . " - Grade Level: " . $row["grade_level"] .
"<br>";
 }
} else {
 echo "0 results";
}
// Execute the SELECT query again (required because the result set's internal pointer
has reached the end)
$result = $conn->query($sql);
// Using mysqli_fetch_array()
if ($result->num_rows > 0) {
 echo "<h3>Using mysqli_fetch_array()</h3>";
 while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)) {
 echo "ID: " . $row["id"] . " - Name: " . $row["first_name"] . " " .
$row["last_name"] . " - Age: " . $row["age"] . " - Grade Level: " . $row["grade_level"] .
"<br>";
 }
} else {
 echo "0 results";
}
// Close the connection
$conn->close();
?>
```

Replace the values for $servername, $username, $password, and $dbname with your actual database connection details. Run the script by accessing the fetch_data.php file in your browser, e.g., http://yourdomain.com/fetch_data.php. If everything is set up correctly, you should see the fetched data displayed using both mysqli_fetch_assoc() and mysqli_fetch_array().

Login form

To create a user login system in PHP using mysqli_connect() and mysqli_query() functions, follow these steps:

Create an HTML form for user login (e.g., login.html):

```
<!DOCTYPE html>
<html>
<head>
 <title>User Login</title>
</head>
<body>
 <h1>User Login</h1>
<form action="login.php" method="post">
 <label for="username">Username:</label>
 <input type="text" id="username" name="username" required>
<br><br>
 <label for="password">Password:</label>
 <input type="password" id="password" name="password"
required><br><br>
 <input type="submit" value="Login">
 </form>
</body>
</html>
```

Create the PHP script to process the login form and authenticate the user (e.g., login.php):

```
<?php
// Replace with your database connection details
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "your_database_name";
// Create a connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check the connection
if (!$conn) {
 die("Connection failed: " . mysqli_connect_error());
}   // Check if the form was submitted
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
 $user = mysqli_real_escape_string($conn, $_POST["username"]);
 $password = mysqli_real_escape_string($conn, $_POST["password"]);
 // Retrieve the user's password from the database
 $sql = "SELECT password FROM users WHERE username = '$user'";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 $row = mysqli_fetch_assoc($result);
 $stored_password = $row["password"];
 // Verify the submitted password against the stored hash
 if (password_verify($password, $stored_password)) {
 echo "Login successful!";
 // Here you can start a session and store user information if needed
 } else {
 echo "Incorrect password!";
}
 } else {
 echo "User not found!";
 }
}
// Close the connection
mysqli_close($conn);
?>
```

Replace the values for $servername, $username, $password, and $dbname with your actual database connection details. Upload both the login.html and login.php files to your web server. Access the login form in your browser by visiting the login.html file, e.g., http://yourdomain.com/login.html, and test the user login system.

Remembering users with cookies and session

Cookies are small text files that are stored on a user's computer by a website. They are commonly used for tracking user activity and preferences, such as remembering login information, tracking shopping cart items, and personalizing website content. In PHP, you can use the setcookie() function to set a cookie. The setcookie() function takes up to six arguments:

➢ name: The name of the cookie.

➢ value: The value of the cookie.

➢ expire: The expiration time of the cookie. This is expressed as a Unix timestamp.

➢ path: The path on the server where the cookie will be available.

➢ domain: The domain that the cookie is available to.

➢ secure: Indicates whether the cookie should only be transmitted over a secure HTTPS connection.

```
<?php
// Set a cookie with a name of "username" and a value of "John"
setcookie("username", "John");
// Retrieve the value of the "username" cookie
$username = $_COOKIE["username"];
// Output the value of the "username" cookie
echo "Hello " . $username;
d. ?>
```

Converting database table to CSV file using fputcsv()

here's an example PHP program that retrieves data from a MySQL database table and converts it to a CSV file using the fputcsv() function:

```
<?php
// Connect to the MySQL database
$host = "localhost";
$username = "yourusername";
$password = "yourpassword";
$dbname = "yourdatabase";
$conn = mysqli_connect($host, $username, $password, $dbname);
// Check the connection
if (!$conn) {
 die("Connection failed: " . mysqli_connect_error());
}
// Query the database
```

```
$sql = "SELECT * FROM yourtable";
$result = mysqli_query($conn, $sql);
// Create a file pointer connected to a CSV file
$fp = fopen('output.csv', 'w');
// Output the column headers to the CSV file
$headers = array('ID', 'Name', 'Email');
fputcsv($fp, $headers);
// Loop through the query results and output each row to the CSV file
while ($row = mysqli_fetch_assoc($result)) {
 fputcsv($fp, $row);
}
// Close the file pointer
fclose($fp);
// Close the database connection
mysqli_close($conn);
?>
```

In this program, we first connect to the MySQL database using the mysqli_connect() function. Then, we query the database using the mysqli_query() function to retrieve all data from the specified table. Next, we create a file pointer connected to a new CSV file using the fopen() function. We then output the column headers to the CSV file using the fputcsv() function with an array of header names. Finally, we loop through the query results using mysqli_fetch_assoc() function, and output each row to the CSV file using the fputcsv() function.

Reading CSV file and reflecting the contents in database

Here's an example PHP program that reads a CSV file and inserts its contents into a MySQL database table:

```
<?php
// Connect to the MySQL database
$host = "localhost";
$username = "yourusername";
$password = "yourpassword";
$dbname = "yourdatabase";
$conn = mysqli_connect($host, $username, $password, $dbname);
```

```
// Check the connection
if (!$conn) {
 die("Connection failed: " . mysqli_connect_error());
}
// Open the CSV file for reading
$fp = fopen('input.csv', 'r');
// Loop through the file line by line
while (($data = fgetcsv($fp)) !== FALSE) {
 // Insert the data into the database table
 $sql = "INSERT INTO yourtable (id, name, email) VALUES ('$data[0]',
'$data[1]', '$data[2]')";
 if (mysqli_query($conn, $sql)) {
 echo "Record inserted successfully";
 } else {
 echo "Error inserting record: " . mysqli_error($conn);
 }
}
// Close the file pointer
fclose($fp);
// Close the database connection
mysqli_close($conn);
?>
```

In this program, we first connect to the MySQL database using the mysqli_connect() function. Then, we open the CSV file for reading using the fopen() function, and loop through each line of the file using the fgetcsv() function. For each line, we construct an INSERT SQL statement that inserts the data into the specified database table using the mysqli_query() function. If the insert is successful, we output a success message. If there is an error, we output an error message along with the MySQL error using the mysqli_error() function.

Once the program finishes executing, the data from the CSV file will be inserted into the specified database table. Note that the CSV file should have the same number of columns as the database table and that the order of the columns in the CSV file should match the order of the columns in the table.