Unit 6: Working with Database

Presented By:
Bipin Maharjan

Learning Objective

- Understand the importance of using databases in PHP applications for efficient data storage, retrieval, and management.
- Connecting PHP applications to a MySQL database using MySQLi extensions.
- Acquire skills in executing SQL queries dynamically in PHP to interact with MySQL databases.
- Explore transaction management concepts to ensure data consistency and integrity in multi-step database operations.

Table of Contents

Introduction to MySQL Database Management System

What is a Database?

- A database is an organized collection of data that can be easily accessed, managed, and updated.
- In web applications, data such as user information, products, comments, and transactions must be stored permanently — this is where databases come in.
- A database allows you to:
  - **Store** information permanently on a server.
  - **Retrieve** information quickly when needed.
  - **Update** existing data efficiently.
  - **Delete** data securely when it’s no longer needed.

What is MySQL?

- MySQL is most popular SQL database management system.
- MySQL is a Relational Database Management System (RDBMS) that stores data in tables.
- It is open-source and developed by Oracle Corporation.
- It organizes data into tables consisting of rows and columns.
- Each table represents a specific entity (e.g., users, products, orders).
- MySQL uses Structured Query Language (SQL) to manage and manipulate data.

Features of MySQL

Feature	Description
Open Source	Free to use and modify.
Relational	Uses relationships between tables using primary and foreign keys.
Fast and Reliable	Optimized for read and write operations, ideal for web apps.
Secure	Supports user privileges and password encryption.
Scalable	Can handle small projects or large enterprise systems.
Cross-Platform	Runs on Windows, Linux, and macOS.
Supports SQL	Uses Structured Query Language to interact with the data.

Common SQL Commands Used in MySQL

Type	Command	Description
DDL (Data Definition Language)	`CREATE`, `ALTER`, `DROP`	Define and modify database structures.
DML (Data Manipulation Language)	`INSERT`, `UPDATE`, `DELETE`	Add, modify, or remove data.
DQL (Data Query Language)	`SELECT`	Retrieve data from tables.
DCL (Data Control Language)	`GRANT`, `REVOKE`	Manage access permissions.
TCL (Transaction Control Language)	`COMMIT`, `ROLLBACK`	Manage transactions for consistency.

Example Database

```
-- Create a new database
CREATE DATABASE studentdb;

-- Use the database
USE studentdb;

-- Create a table
CREATE TABLE students (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100),
  course VARCHAR(50)
);
```

How PHP Works with MySQL

- PHP acts as a bridge between the user and the MySQL database.
- PHP code sends SQL queries to the MySQL server using extensions like MySQLi or PDO.
- MySQL executes the queries and returns the results to PHP, which then displays them to the user.

Tools for Managing MySQL

Tool	Description
phpMyAdmin	Web-based interface to manage databases (used in XAMPP).
MySQL Workbench	Graphical tool for advanced database design and SQL execution.
Command Line (mysql)	Terminal-based MySQL management.
HeidiSQL / DBeaver	Third-party GUI tools for managing databases.

Advantages of Using MySQL with PHP

- Easy integration with PHP (through MySQLi or PDO).
- High performance for large-scale web apps.
- Secure and supports encryption.
- Excellent documentation and community support.
- Widely used in content management systems (WordPress, Joomla, Drupal).

Connecting PHP with MySQL database

Methods to Connect PHP with MySQL

There are two major mays PHP can connect to MySQL:

Method	Description	Style
MySQLi (MySQL Improved)	Supports both procedural and object-oriented styles; provides better performance and security.	Procedural / OOP
PDO (PHP Data Objects)	Works with multiple database types (MySQL, PostgreSQL, Oracle, etc.).	Object-Oriented only

We’ll yse MySQLi (Procedural style) because it’s simple and ideal for beginners.

MySQLi Connection

Syntax: mysqli_connect(servername, username, password, databasename);

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "studentdb";

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

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

?>
```

MySQLi Connection: Explanation

Code	Meaning
`$servername = "localhost";`	The server where MySQL is running. “localhost” is used for local servers like XAMPP/WAMP.
`$username = "root";`	Default username for local MySQL.
`$password = "";`	Default password (blank in XAMPP).
`$database = "studentdb";`	The name of the database to connect to.
`mysqli_connect()`	Function to open the connection.
`mysqli_connect_error()`	Returns the error message if connection fails.
`die()`	Stops the script and prints the message if the connection fails.

Closing the Database Connection

After finishing your database operations, you should close the connection to free server resources.

Syntax: mysqli_close($conn);

Closing the Database Connection Example

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "studentdb";
// Create a connection
$conn = mysqli_connect($servername, $username, $password, $database);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully!";
// Business Logic and Database Operations Here …
mysqli_close($conn);
echo "Connection closed.";
?>
```

Common Connection Errors

Error Message	Cause	Solution
`Access denied for user 'root'@'localhost'`	Wrong username or password	Check credentials
`Unknown database 'studentdb'`	Database doesn’t exist	Create the database first
`Can't connect to MySQL server`	MySQL service not running	Start MySQL in XAMPP/WAMP
`Connection failed: ...`	Wrong hostname or network issue	Verify server name and status

Using Object-Oriented MySQLi (Optional)

If you prefer OOP style, you can connect as follows:

```
<?php
$conn = new mysqli("localhost", "root", "", "studentdb");

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully (OOP Style)!";
$conn->close();
?>
```

Performing CRUD operations (Create, Read, Update, Delete)

Setting up the Database

First, create a database and a table to work with.

```
CREATE DATABASE studentdb;
USE studentdb;

CREATE TABLE students (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100),
  course VARCHAR(50)
);
```

Database Connection File

Before performing any operation, include a connection file to connect PHP with MySQL.

File: db_connect.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "studentdb";

$conn = mysqli_connect($servername, $username, $password, $database);

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

Note: You'll reuse this connection file in every CRUD script

Create - Insert Data into Database

File: add_student.html

```
<form method="POST" action="insert.php">

  Name: <input type="text" name="name"><br>

  Email: <input type="email" name="email"><br>

  Course: <input type="text"
name="course"><br>

  <input type="submit" value="Add Student">

</form>
```

File: insert.php

```
<?php
include("db_connect.php");

$name = $_POST['name'];
$email = $_POST['email'];
$course = $_POST['course'];

$sql = "INSERT INTO students (name, email,
course) VALUES ('$name', '$email', '$course')";

if (mysqli_query($conn, $sql)) {
    echo "Student added successfully!";
} else {
    echo "Error: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Read – Retrieve and Display Data

File: display_students.php

```
<?php
include("db_connect.php");

$sql = "SELECT * FROM students";
$result = mysqli_query($conn, $sql);

echo "<h3>Student List</h3>";

if (mysqli_num_rows($result) > 0) {
    echo "<table border='1' cellpadding='5'>
            <tr><th>ID</th><th>Name</th><th>Email</th><th>Course</th></tr>";
    while ($row = mysqli_fetch_assoc($result)) {
        echo "<tr>
                <td>{$row['id']}</td>
                <td>{$row['name']}</td>
                <td>{$row['email']}</td>
                <td>{$row['course']}</td>
              </tr>";
    }
    echo "</table>";
} else {
    echo "No records found.";
}

mysqli_close($conn);
?>
```

Update – Modify Existing Data

File: update_student.php

```
<?php
include("db_connect.php");

$id = 1; //student ID to update
$new_course = "CSIT";

$sql = "UPDATE students SET course='$new_course' WHERE id=$id";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully.";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Delete – Remove a Record

File: delete_student.php

```
<?php
include("db_connect.php");

$id = 2; // The student ID to delete

$sql = "DELETE FROM students WHERE id=$id";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully.";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Executing SQL queries using PHP

What is an SQL Query?

An SQL (Structured Query Language) query is a command that interacts with the database.

It is used to:

- Retrieve data (SELECT)
- Insert data (INSERT)
- Update data (UPDATE)
- Delete data (DELETE)

Executing Queries in PHP

PHP uses the mysqli_query() function to execute SQL statements.

Syntax:

```
mysqli_query(connection, query);
```

- connection: The link returned by mysqli_connect()
- query: The SQL statement to be executed

Types of SQL Queries: Non-SELECT Queries (INSERT, UPDATE, DELETE)

These queries do not return data — only a success or failure status.

Example

```
$sql = "INSERT INTO students (name, email, course) VALUES ('John Doe',
'john@example.com', 'BSc IT')";
if (mysqli_query($conn, $sql)) {
    echo "Record inserted successfully!";
} else {
    echo "Error: " . mysqli_error($conn);
}
```

Types of SQL Queries: SELECT Queries

These queries return data from the database.

Example:

```
$sql = "SELECT * FROM students";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result)) {
        echo "ID: " . $row["id"] . " | Name: " . $row["name"] . "<br>";
    }
} else {
    echo "No records found!";
}
```

Common MySQLi Functions Used

Function	Description
`mysqli_query()`	Executes a query against the database
`mysqli_num_rows()`	Counts number of rows returned by a SELECT query
`mysqli_fetch_assoc()`	Fetches each row as an associative array
`mysqli_fetch_array()`	Fetches row as both numeric and associative array
`mysqli_error()`	Returns description of last error
`mysqli_close()`	Closes the database connection

User Registration and Login

Why User Registration and Login?

Almost every dynamic web application requires user accounts to:

- Identify users uniquely.
- Personalize content.
- Restrict access to certain pages.
- Track activity or store user preferences.

A login system ensures that only authenticated users can access protected resources.

Database Table for Users

Create a users table in the studentdb database:

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) NOT NULL UNIQUE,
  email VARCHAR(100) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL
);
```

Database Connection File: db_connect.php

Use the connection file db_connect.php:

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$database = "studentdb";

$conn = mysqli_connect($servername, $username, $password, $database);

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

Register User: register.php

```php
<?php
// register.php
include('db_connect.php');

$error = '';
$success = '';

// Initialize form variables
$username = '';
$email = '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Retain input values
    $username = trim($_POST['username']);
    $email    = trim($_POST['email']);
    $pass     = $_POST['password'];

    // --- Validation Section ---
    if ($username === '' || $email === '' || $pass === '') {
        $error = 'All fields are required.';
    } elseif (strlen($username) < 3 || strlen($username) > 50) {
        $error = 'Username must be between 3 and 50 characters.';
    } elseif (strlen($email) < 5 || strlen($email) > 100) {
        $error = 'Email must be between 5 and 100 characters.';
    } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $error = 'Please enter a valid email address.';
    }
else {
        // --- Check if user already exists ---
        $check_stmt = mysqli_prepare($conn, "SELECT id FROM users WHERE
username = ? OR email = ?");
        mysqli_stmt_bind_param($check_stmt, "ss", $username, $email);
        mysqli_stmt_execute($check_stmt);
        mysqli_stmt_store_result($check_stmt);

        if (mysqli_stmt_num_rows($check_stmt) > 0) {
            $error = 'Username or email already exists. Please try another.';
        } else {
            // --- Insert new user ---
            $password_hash = password_hash($pass, PASSWORD_DEFAULT);
            $insert_stmt = mysqli_prepare($conn, "INSERT INTO users (username, email,
password) VALUES (?, ?, ?)");
            mysqli_stmt_bind_param($insert_stmt, "sss", $username, $email,
$password_hash);

            if (mysqli_stmt_execute($insert_stmt)) {
                // Redirect to login page on success
                header("Location: login.php?registered=1");
                exit();
            } else {
                $error = 'Database error: ' . mysqli_error($conn);
            }
            mysqli_stmt_close($insert_stmt);
        }

        mysqli_stmt_close($check_stmt);
    }
}
?>
```

User Registration Form: register.php

```
<!DOCTYPE html>
<html>
<head>
    <title>User Registration</title>
</head>
<body>
<h2>Register</h2>

<?php if ($error): ?>
    <p style="color:red;"><?php echo htmlspecialchars($error); ?></p>
<?php endif; ?>

<form method="POST" action="">
    <label>Username:</label><br>
    <input type="text" name="username" maxlength="50"
           value="<?php echo htmlspecialchars($username); ?>" required><br><br>

    <label>Email:</label><br>
    <input type="email" name="email" maxlength="100"
           value="<?php echo htmlspecialchars($email); ?>" required><br><br>

    <label>Password:</label><br>
    <input type="password" name="password" required><br><br>

    <input type="submit" value="Register">
</form>

<p>Already registered? <a href="login.php">Login here</a></p>
</body>
</html>
```

User Login Form: login.php

```php
<?php
// login.php
session_start();
include('db_connect.php');

// If already logged in, go direct to welcome
if (isset($_SESSION['username'])) {
    header("Location: welcome.php");
    exit();
}

$error = '';
$registered = isset($_GET['registered']) ? true :
false;

if ($_SERVER['REQUEST_METHOD'] === 'POST')
{
    $username = trim($_POST['username']);
    $password = $_POST['password'];

  if ($username === '' || $password === '') {
      $error = 'Please enter username and password.';
   } else {
      // Prepared statement to avoid SQL injection
      $stmt = mysqli_prepare($conn, "SELECT id, username, password FROM users WHERE
username = ?");
      if ($stmt) {
         mysqli_stmt_bind_param($stmt, "s", $username);
         mysqli_stmt_execute($stmt);
         mysqli_stmt_store_result($stmt);

         if (mysqli_stmt_num_rows($stmt) === 1) {
            mysqli_stmt_bind_result($stmt, $id, $db_username, $db_password_hash);
            mysqli_stmt_fetch($stmt);

            if (password_verify($password, $db_password_hash)) {
               // Successful login: regenerate id, set session and redirect
               session_regenerate_id(true);
               $_SESSION['username'] = $db_username;
               $_SESSION['user_id']  = $id;

               header("Location: welcome.php");
               exit(); // Important: stop executing after redirect
            } else {
               $error = 'Invalid password.';
            }
         } else {
            $error = 'User not found.';
         }
         mysqli_stmt_close($stmt);
      } else {
         $error = 'Database error: ' . mysqli_error($conn);
      }
   }
}
?>
```

Welcome Page: welcome.php

```
<?php
// welcome.php
session_start();
if (!isset($_SESSION['username'])) {
    // Not logged in — redirect to login
    header("Location: login.php");
    exit();
}
?>
<!DOCTYPE html>
<html>
<head><title>Welcome</title></head>
<body>
<h2>Welcome, <?php echo htmlspecialchars($_SESSION['username'], ENT_QUOTES, 'UTF-8'); ?>!</h2>
<p>This is your protected welcome/dashboard page.</p>
<p><a href="logout.php">Logout</a></p>
</body>
</html>
```

Logout: logout.php

```
<?php
session_start();
session_destroy();

// Redirect back to login
header("Location: login.php");
exit();
```

Error Handling and Transaction Management

What is Error Handling?

Error handling refers to detecting and managing problems that occur while the program runs — for example:

- Database connection failure
- SQL query syntax errors
- Missing data
- Permission or constraint violations

Proper error handling:

- Helps identify issues quickly
- Prevents application crashes
- Keeps the user informed in a user-friendly way

Common Types of Errors in PHP–MySQL

Type	Example	Description
Connection Error	Server not reachable, wrong credentials	Fails to connect to DB
Query Error	Syntax mistake in SQL	Wrong SQL statement
Constraint Error	Duplicate key, foreign key violation	Data rule violated

Handling Errors in MySQLi

You can detect and display errors using these functions:

Function	Purpose
`mysqli_connect_error()`	Returns error from failed connection
`mysqli_error($conn)`	Returns last query error
`mysqli_errno($conn)`	Returns numeric error code

Handling Errors in MySQLi: Example

```
$conn = mysqli_connect("localhost", "root", "",
"studentdb");

if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

$sql = "SELECT * FORM students"; // intentional
typo (FORM instead of FROM)
$result = mysqli_query($conn, $sql);

if (!$result) {
    echo "Query error: " . mysqli_error($conn);
}
```

Output:

Query error: You have an error in your SQL syntax; check the manual...

What is a Transaction?

A transaction is a group of SQL operations executed as a single logical unit of work.

It ensures that either all operations succeed, or none do — maintaining data integrity.

Example scenario:

Transferring money between two accounts:

- Debit from account A
- Credit to account B

If one fails, both should be rolled back to keep balances correct.

Transaction Control Commands

Command	Purpose
`START TRANSACTION`	Begins a new transaction
`COMMIT`	Saves all changes permanently
`ROLLBACK`	Reverts all changes if something goes wrong

Transaction Management in PHP: Example

Here’s a simple PHP demo where two SQL operations must either both succeed or both fail.

Firstly, create “bankdb” having table “accounts”.

```
CREATE TABLE accounts (
  id INT AUTO_INCREMENT PRIMARY KEY,
  holder_name VARCHAR(50) NOT NULL,
  balance DECIMAL(10,2) NOT NULL
);
INSERT INTO accounts (holder_name, balance) VALUES ('Alice', 2000.00), ('Bob', 1500.00);
```

transaction.php

```
<?php
$conn = mysqli_connect("localhost", "root", "",
"bankdb");

if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

echo "<h3>Bank Transaction Example</h3>";

// Start transaction
mysqli_begin_transaction($conn);
```

```
try {
    // Step 1: Debit account A
    $debit = "UPDATE accounts SET balance = balance - 500 WHERE id = 1";
    if (!mysqli_query($conn, $debit)) {
        throw new Exception("Error debiting account A: " . mysqli_error($conn));
    }

    // Step 2: Credit account B
    $credit = "UPDATE accounts SET balance = balance + 500 WHERE id =
2";
    if (!mysqli_query($conn, $credit)) {
        throw new Exception("Error crediting account B: " . mysqli_error($conn));
    }

    // If both queries succeed
    mysqli_commit($conn);
    echo "<p style='color:green;'>✅ Transaction successful! Amount
transferred.</p>";

} catch (Exception $e) {
    // Rollback changes on failure
    mysqli_rollback($conn);
    echo "<p style='color:red;'>❌ Transaction failed: " . $e->getMessage() .
"</p>";
}

mysqli_close($conn);
?>
```

Why Transactions are Important?

- Prevent partial updates when something fails
- Ensure data consistency
- Useful in financial apps, inventory updates, and multi-table operations

Any Question?