Lesson Plan: Creating a Maintainable Database-Driven Web Application with PHP and MySQL

Lesson Overview

This lesson guides students through creating and structuring a database-driven web application using PHP and MySQL. By the end, students will understand how to build a database with XAMPP, access it using PHP, and organize database code to simplify future maintenance and revisions.

Prerequisites

- Basic knowledge of HTML and PHP
- Understanding of object-oriented programming (OOP) concepts in PHP
- Basic understanding of SQL

Setup

- Install XAMPP (provides Apache server, MySQL database, and PHP environment)
- Set up XAMPP and launch the Apache and MySQL services

Part 1: Setting Up a Database in XAMPP

Step 1: Launch phpMyAdmin

1. Open XAMPP and start the **Apache** and **MySQL** modules.
2. Open a browser and go to http://localhost/phpmyadmin/.

Step 2: Create a New Database

1. In phpMyAdmin, click on **Databases** in the top menu.
2. Name the database (e.g., art_gallery_db) and click **Create**.

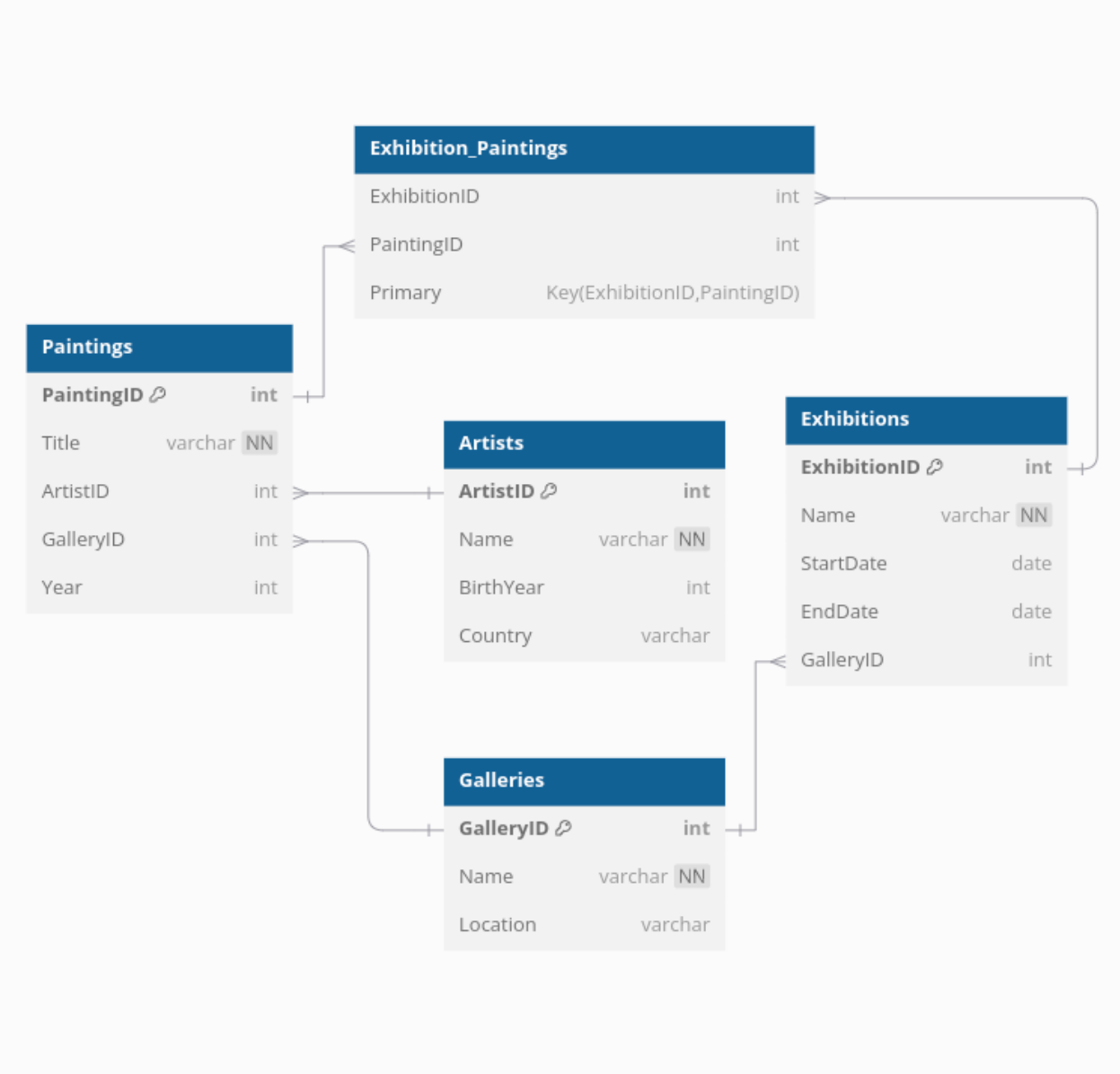
Step 3: Creating Tables

There are two primary ways to create tables:

- **Using phpMyAdmin's Interface**:
  - Click on the database you created (art_gallery_db).
  - Click **New** to add a table.
  - Name the table (e.g., Paintings) and specify the number of columns.
  - Add columns such as PaintingID, Title, ArtistID, and GalleryID.
  - Click **Save** to create the table.
- **Using SQL Queries**:
  - In phpMyAdmin, go to the **SQL** tab.
  - Enter the following SQL command to create a table for Paintings:

```
CREATE TABLE Paintings (
 PaintingID INT AUTO_INCREMENT PRIMARY KEY,
 Title VARCHAR(255) NOT NULL,
 ArtistID INT NOT NULL,
 GalleryID INT NOT NULL
);
```

3. **Add Data** to the table by navigating to the Insert tab and filling in values for each column.
4. Now, likewise, **implement** the following schema:



Part 2: Setting Up Database Connection in PHP

Step 1: Understand the Problem of Dependency

- New programmers often use PDO (PHP Data Objects) connections directly in each PHP page. However, this creates multiple dependencies and makes future changes harder.
- Instead, create a single database helper class to manage connections and queries.

Step 2: Create a Database Helper Class

1. Under htdocs (if you are using xampp), create a new folder for this small project (call it anything, a good name could be art_gallery_website).
2. Create a PHP file called DatabaseHelper.php.
3. Add the following code:

```
class DatabaseHelper {
  public static function createConnection($values = array()) {
    $connString = $values[0];
    $user = $values[1];
    $password = $values[2];
    $pdo = new PDO($connString, $user, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE,
PDO::FETCH_ASSOC);
    return $pdo;
  }

  public static function runQuery($pdo, $sql, $parameters = array()) {
    if (!is_array($parameters)) {
      $parameters = array($parameters);
    }
    $statement = null;
    if (count($parameters) > 0) {
      $statement = $pdo->prepare($sql);
      $executedOk = $statement->execute($parameters);
      if (!$executedOk) throw new PDOException;
    } else {
      $statement = $pdo->query($sql);
      if (!$statement) throw new PDOException;
    }
    return $statement;
  }
}
```

Step 3: Using the Database Helper Class

1. In your main PHP file (e.g., index.php), include the DatabaseHelper.php file.
2. Create a connection using:

```
try {
    $conn =
DatabaseHelper::createConnection(['mysql:host=localhost;dbname=art_gallery_d
b', 'root', '']);
    $sql = "SELECT * FROM Paintings";
    $paintings = DatabaseHelper::runQuery($conn, $sql);
    foreach ($paintings as $p) {
        echo $p["Title"];
    }
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}
```

Part 3: Designing for Future Scalability with Table Gateway Pattern

Step 1: Create a Table Gateway Class

1. In the DatabaseHelper.php file, add a new class for each table you need to interact with. Here's an example for the Paintings table:

```
class PaintingDB {
    private static $baseSQL = "SELECT * FROM Paintings ";
    private $pdo;

    public function __construct($connection) {
        $this->pdo = $connection;
    }

    public function getAll() {
        $sql = self::$baseSQL;
        $statement = DatabaseHelper::runQuery($this->pdo, $sql, null);
```

```
        return $statement->fetchAll();
    }

    public function findById($id) {
        $sql = self::$baseSQL . " WHERE PaintingID=?";
        $statement = DatabaseHelper::runQuery($this->pdo, $sql, array($id));
        return $statement->fetch();
    }
}
```

Step 2: Use the Gateway Class in a PHP Page

Use PaintingDB in your main PHP file:

```
try {
    $conn =
DatabaseHelper::createConnection(['mysql:host=localhost;dbname=art_gallery_d
b', 'root', '']);
    $paintingGateway = new PaintingDB($conn);

    // Retrieve all paintings
    $paintings = $paintingGateway->getAll();
    foreach ($paintings as $painting) {
        echo $painting["Title"];
    }
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}
```