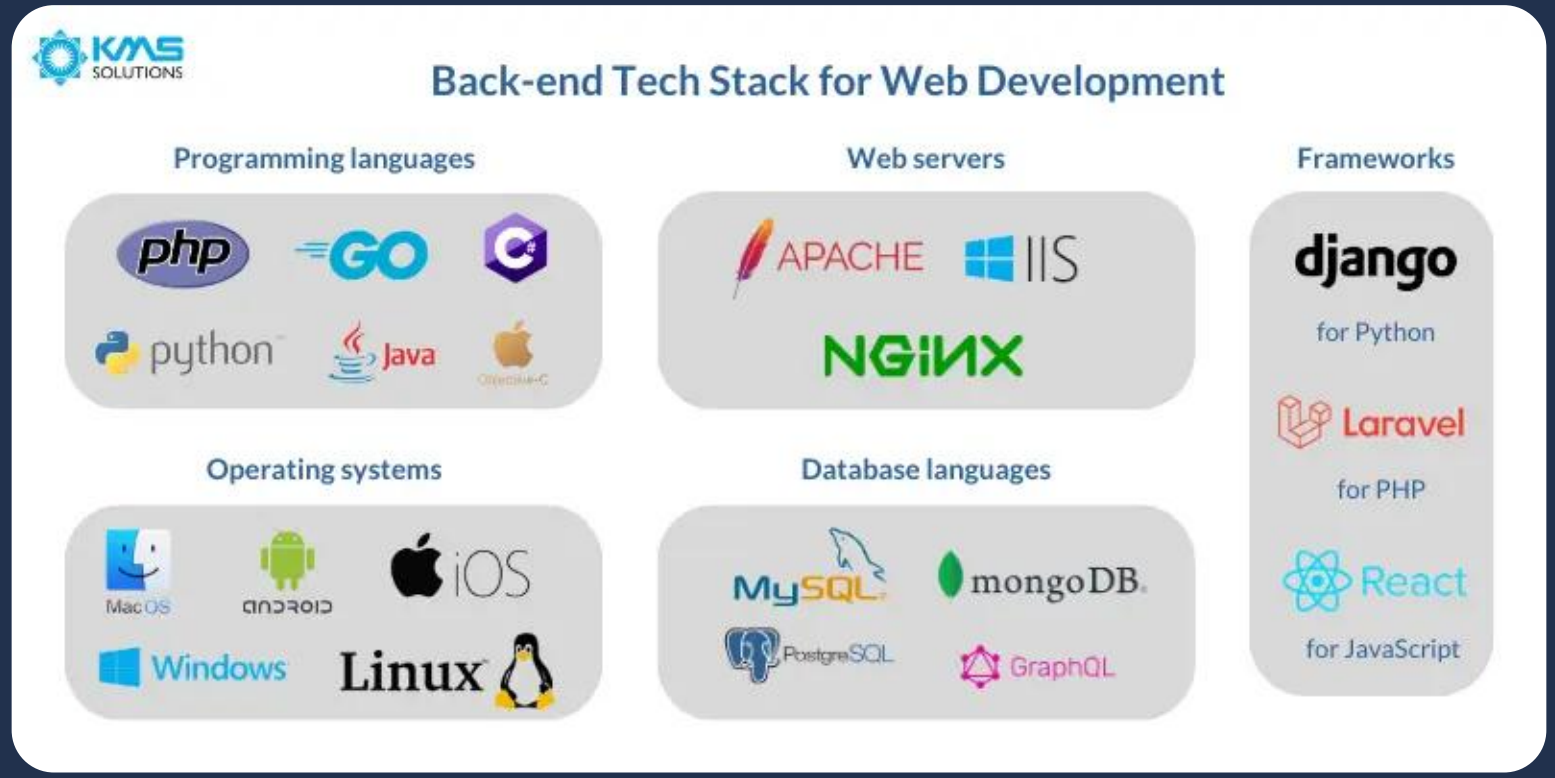
CSE485 – Công nghệ Web

dungkt@tlu.edu.vn

QUI ĐỊNH

- Lớp học KHÔNG DÙNG ĐIỆN THOẠI
- Lớp học KHÔNG MẶC QUẦN ĐÙI đến lớp
- Lớp học CẦN MANG LAPTOP
- Lớp học KHÔNG ĐI MUỘN (*chậm nhất sau 5 phút không vào lớp*)
- Lớp học CẦN LÀM BÀI TẬP ĐẦY ĐỦ, NỘP ĐÚNG HẠN
- Lớp học BỊ PHẠT THEO NHÓM nếu ngồi cùng BÀN với thành viên vi phạm các QUI ĐỊNH
- Máy tính cần dán biển tên để định dang gọi tên SV



Bài 4. PHP và CSDL

NỘI DUNG

1. Giới thiệu về CSDL
2. MySQL/MariaDB và phpMyAdmin
3. SELECT dữ liệu từ CSDL
4. Kết nối CSDL từ PHP
5. Truy vấn dữ liệu từ PHP
6. Tạo trang hiển thị 1 bài viết
7. Thêm bài viết mới
8. SQL Injection
9. Tạo hàm kết nối CSDL

NỘI DUNG

10. Đảm bảo tính hợp lệ của FORM Thêm
11. Phòng ngừa tấn công XSS
12. Chuyển hướng sau khi thêm bài viết
13. Hàm lấy article theo id
14. Tạo Form EDIT article
15. Xóa bài viết
16. Tăng tốc truy vấn
17. Login và Session

1. GIỚI THIỆU VỀ CSDL

- Cơ sở dữ liệu là một tập hợp có cấu trúc các dữ liệu, được tổ chức và lưu trữ để truy xuất được dễ dàng. Nó giống như một thư viện để có thể lưu trữ, tìm kiếm và quản lý thông tin một cách hệ thống.

```
CREATE TABLE articles (
  `title` VARCHAR(255) NOT NULL,
  `content` TEXT NOT NULL,
  `author` VARCHAR(255) NOT NULL,
  `created_at` DATE NOT NULL
);
```

```php
<?php
$articles = [
    [
        'title' => 'Bài viết đầu tiên',
        'content' => 'Đây là nội dung của bài viết đầu
tiên.',
        'author' => 'Nguyễn Văn A',
        'created_at' => '2023-11-20'
    ],
    [
        'title' => 'Bài viết thứ hai',
        'content' => 'Đây là nội dung của bài viết thứ
hai.',
        'author' => 'Trần Thị B',
        'created_at' => '2023-11-21'
    ],
    // ... thêm các bài viết khác vào đây
];
?>
```

1. GIỚI THIỆU VỀ CSDL

- Cơ sở dữ liệu là một tập hợp có cấu trúc các dữ liệu, được tổ chức và lưu trữ để truy xuất được dễ dàng. Nó giống như một thư viện để có thể lưu trữ, tìm kiếm và quản lý thông tin một cách hệ thống.

```
<!DOCTYPE html>
<html>
<head>
    <title>Danh sách bài viết</title>
</head>
<body>
    <h1>Danh sách bài viết</h1>
    <ul>
        <?php foreach ($articles as $article): ?>
            <li>
                <h2><?= $article['title'] ?></h2>
                <p><?= $article['content'] ?></p>
                <p>Tác giả: <?= $article['author'] ?></p>
                <p>Ngày đăng: <?= $article['created_at']
?></p>
            </li>
        <?php endforeach; ?>
    </ul>
</body>
</html>
```

2. MySQL/MariaDB và phpMyAdmin

- **MySQL** được phát triển và duy trì bởi Oracle, trong khi **MariaDB** được phát triển bởi các thành viên cốt lõi của nhóm phát triển MySQL sau khi Oracle mua lại Sun Microsystems (công ty mẹ của MySQL).
  - MariaDB được xem là một phiên bản "fork" của MySQL, tương thích với hầu hết các tính năng của MySQL, nhưng cũng có một số cải tiến và tính năng bổ sung.
- **phpMyAdmin** là một công cụ web-based để quản lý và thao tác với **MySQL/MariaDB**. Nó cung cấp một giao diện đồ họa thân thiện người dùng, cho phép bạn thực hiện các tác vụ quản lý cơ sở dữ liệu như:
  - Tạo, sửa, xóa các cơ sở dữ liệu, bảng, chỉ mục, v.v.
  - Chạy các truy vấn SQLXem và chỉnh sửa dữ liệu trong các bảng
  - Sao lưu và khôi phục cơ sở dữ liệu
  - Quản lý người dùng và quyền truy cập
  - Theo dõi hoạt động của hệ thống
- **phpMyAdmin** được phát triển bằng ngôn ngữ PHP và có thể chạy trên hầu hết các máy chủ web hỗ trợ PHP. Nó là một công cụ rất hữu ích và phổ biến trong việc quản lý các cơ sở dữ liệu MySQL/MariaDB.

2. MySQL/MariaDB và phpMyAdmin

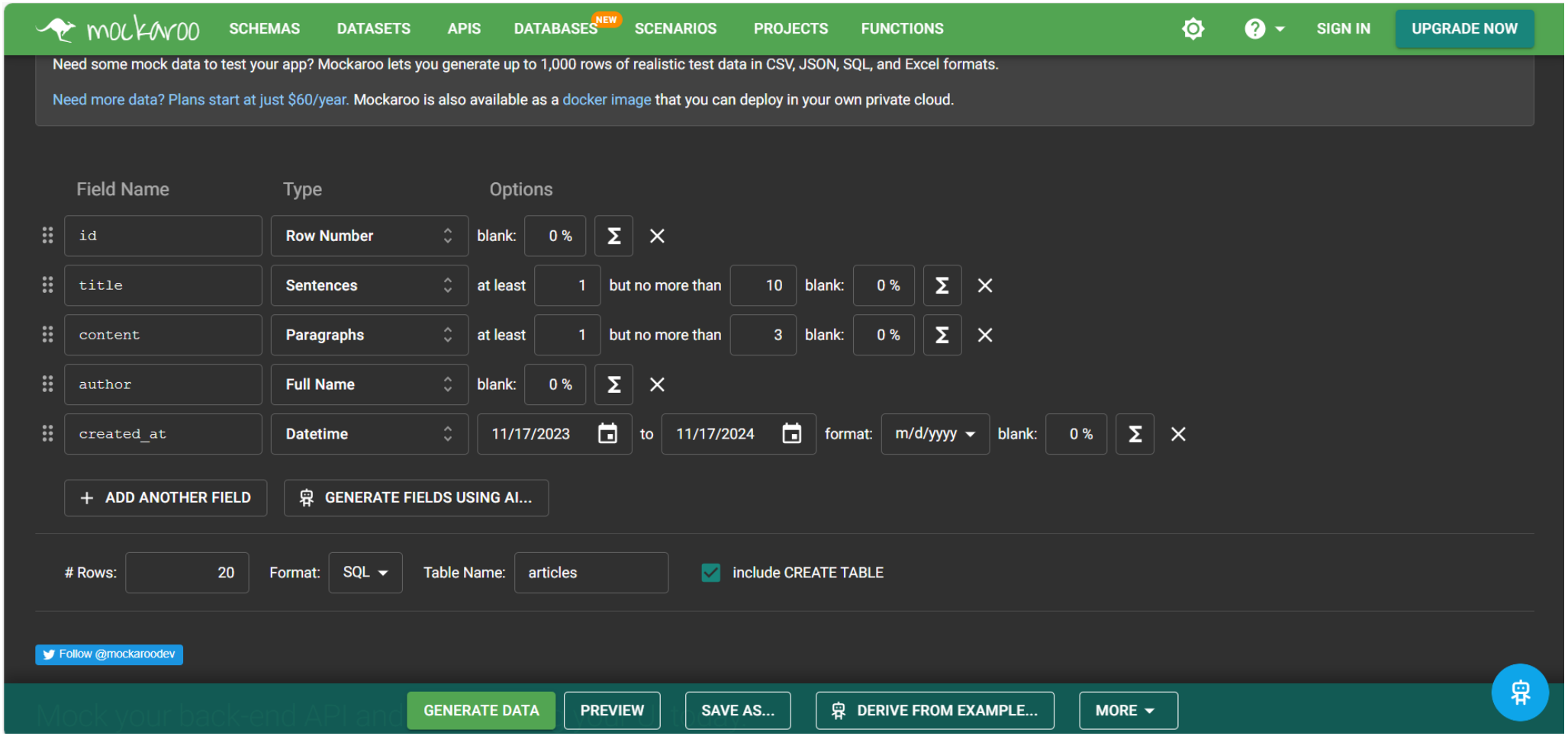
- **Các kiểu dữ liệu thông dụng:**
  - **Số nguyên (Integers)**:
    - TINYINT: -128 đến 127 (8 bits)
    - SMALLINT: -32,768 đến 32,767 (16 bits)
    - MEDIUMINT: -8,388,608 đến 8,388,607 (24 bits)
    - INT (hoặc INTEGER): -2,147,483,648 đến 2,147,483,647 (32 bits)
    - BIGINT: -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807 (64 bits)
  - **Số thực (Floating-point)**:
    - FLOAT: số dấu phẩy động, độ chính xác 7 chữ số
    - DOUBLE: số dấu phẩy động, độ chính xác 15 chữ số
  - **Chuỗi ký tự (Strings)**:
    - CHAR(n): chuỗi ký tự cố định độ dài n (tối đa 255 ký tự)
    - VARCHAR(n): chuỗi ký tự có độ dài thay đổi, tối đa n ký tự
    - TEXT: chuỗi ký tự có độ dài tối đa 65,535 ký tự
    - LONGTEXT: chuỗi ký tự có độ dài tối đa 4,294,967,295 ký tự

2. MySQL/MariaDB và phpMyAdmin

- **Các kiểu dữ liệu thông dụng:**
  - **Ngày tháng (Dates and Times):**
    - DATE: định dạng YYYY-MM-DD
    - TIME: định dạng HH:MM:SS
    - DATETIME: định dạng YYYY-MM-DD HH:MM:SS
    - TIMESTAMP: định dạng YYYY-MM-DD HH:MM:SS, tự động cập nhật khi dữ liệu thay đổi
  - **Các kiểu dữ liệu khác**:
    - BOOLEAN: lưu trữ giá trị true/false
    - ENUM: cho phép lựa chọn một giá trị từ một tập hợp các giá trị xác định trước
    - SET: cho phép lựa chọn một hoặc nhiều giá trị từ một tập hợp các giá trị xác định trước

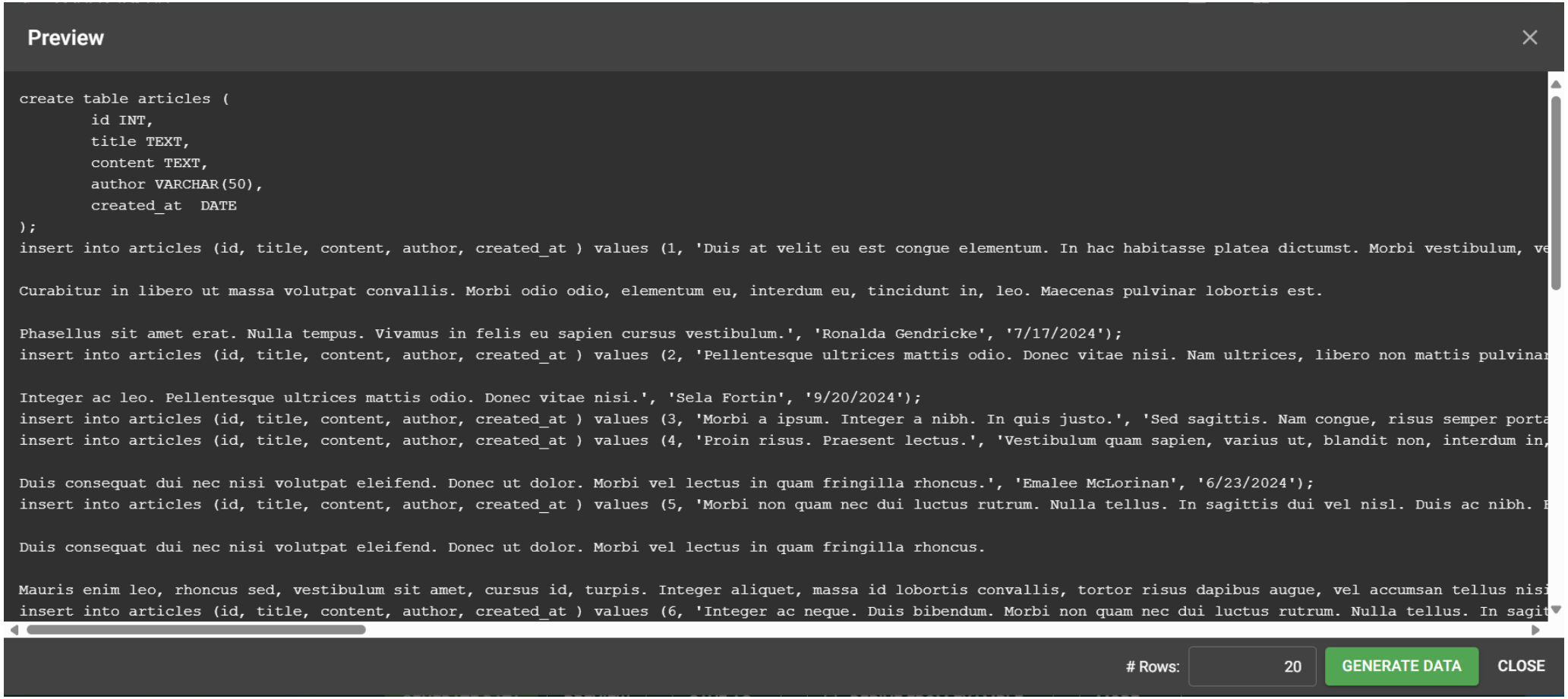
2. MySQL/MariaDB và phpMyAdmin

- Sinh dữ liệu fake (dữ liệu minh họa) cho CSDL



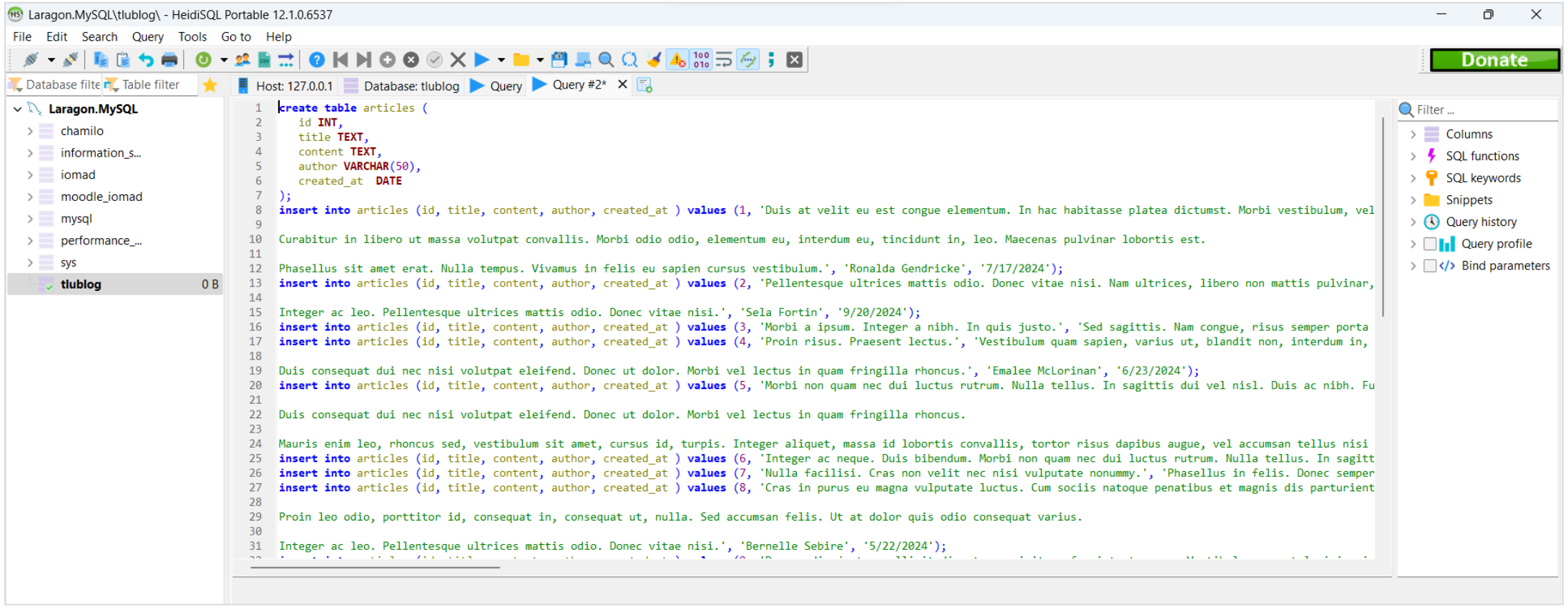
2. MySQL/MariaDB và phpMyAdmin

- Sinh dữ liệu fake (dữ liệu minh họa) cho CSDL



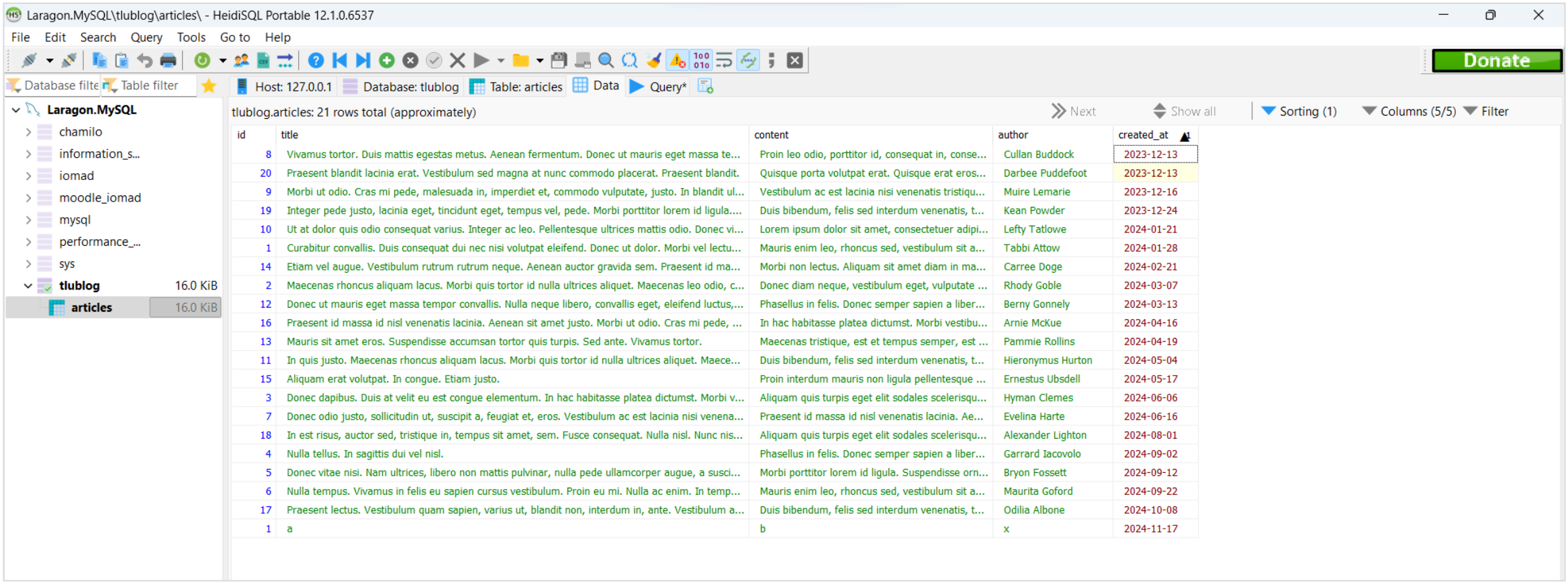
2. MySQL/MariaDB và phpMyAdmin

- Sinh dữ liệu fake (dữ liệu minh họa) cho CSDL



2. MySQL/MariaDB và phpMyAdmin

- Sinh dữ liệu fake (dữ liệu minh họa) cho CSDL

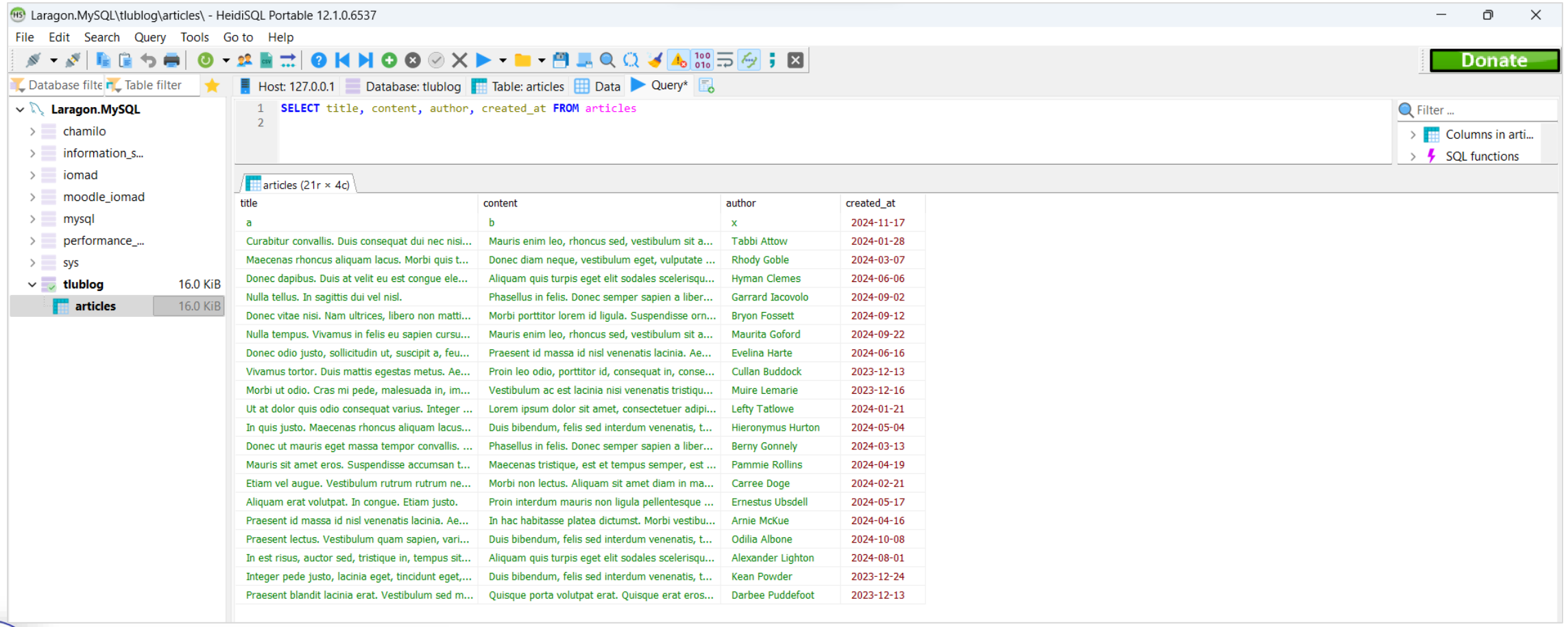


id	title	content	author	created_at
8	Vivamus tortor. Duis mattis egestas metus. Aenean fermentum. Donec ut mauris eget massa te...	Proin leo odio, porttitor id, consequat in, conse...	Cullan Buddock	2023-12-13
20	Praesent blandit lacinia erat. Vestibulum sed magna at nunc commodo placerat. Praesent blandit.	Quisque porta volutpat erat. Quisque erat eros...	Darbee Puddefoot	2023-12-13
9	Morbi ut odio. Cras mi pede, malesuada in, imperdiet et, commodo vulputate, justo. In blandit ul...	Vestibulum ac est lacinia nisi venenatis tristiqu...	Muire Lemarie	2023-12-16
19	Integer pede justo, lacinia eget, tincidunt eget, tempus vel, pede. Morbi porttitor lorem id ligula....	Duis bibendum, felis sed interdum venenatis, t...	Kean Powder	2023-12-24
10	Ut at dolor quis odio consequat varius. Integer ac leo. Pellentesque ultrices mattis odio. Donec vi...	Lorem ipsum dolor sit amet, consectetuer adipi...	Lefty Tatlowe	2024-01-21
1	Curabitur convallis. Duis consequat dui nec nisi volutpat eleifend. Donec ut dolor. Morbi vel lectu...	Mauris enim leo, rhoncus sed, vestibulum sit a...	Tabbi Attow	2024-01-28
14	Etiam vel augue. Vestibulum rutrum rutrum neque. Aenean auctor gravida sem. Praesent id ma...	Morbi non lectus. Aliquam sit amet diam in ma...	Carree Doge	2024-02-21
2	Maecenas rhoncus aliquam lacus. Morbi quis tortor id nulla ultrices aliquet. Maecenas leo odio, c...	Donec diam neque, vestibulum eget, vulputate ...	Rhody Goble	2024-03-07
12	Donec ut mauris eget massa tempor convallis. Nulla neque libero, convallis eget, eleifend luctus,...	Phasellus in felis. Donec semper sapien a liber...	Berny Gonnely	2024-03-13
16	Praesent id massa id nisl venenatis lacinia. Aenean sit amet justo. Morbi ut odio. Cras mi pede, ...	In hac habitasse platea dictumst. Morbi vestibu...	Arnie McKue	2024-04-16
13	Mauris sit amet eros. Suspendisse accumsan tortor quis turpis. Sed ante. Vivamus tortor.	Maecenas tristique, est et tempus semper, est ...	Pammie Rollins	2024-04-19
11	In quis justo. Maecenas rhoncus aliquam lacus. Morbi quis tortor id nulla ultrices aliquet. Maece...	Duis bibendum, felis sed interdum venenatis, t...	Hieronymus Hurton	2024-05-04
15	Aliquam erat volutpat. In congue. Etiam justo.	Proin interdum mauris non ligula pellentesque ...	Ernestus Ubsdell	2024-05-17
3	Donec dapibus. Duis at velit eu est congue elementum. In hac habitasse platea dictumst. Morbi v...	Aliquam quis turpis eget elit sodales scelerisqu...	Hyman Clemes	2024-06-06
7	Donec odio justo, sollicitudin ut, suscipit a, feugiat et, eros. Vestibulum ac est lacinia nisi venena...	Praesent id massa id nisl venenatis lacinia. Ae...	Evelina Harte	2024-06-16
18	In est risus, auctor sed, tristique in, tempus sit amet, sem. Fusce consequat. Nulla nisl. Nunc nis...	Aliquam quis turpis eget elit sodales scelerisqu...	Alexander Lighton	2024-08-01
4	Nulla tellus. In sagittis dui vel nisl.	Phasellus in felis. Donec semper sapien a liber...	Garrard Iacovolo	2024-09-02
5	Donec vitae nisi. Nam ultrices, libero non mattis pulvinar, nulla pede ullamcorper augue, a susci...	Morbi porttitor lorem id ligula. Suspendisse orn...	Bryon Fossett	2024-09-12
6	Nulla tempus. Vivamus in felis eu sapien cursus vestibulum. Proin eu mi. Nulla ac enim. In temp...	Mauris enim leo, rhoncus sed, vestibulum sit a...	Maurita Goford	2024-09-22
17	Praesent lectus. Vestibulum quam sapien, varius ut, blandit non, interdum in, ante. Vestibulum a...	Duis bibendum, felis sed interdum venenatis, t...	Odilia Albone	2024-10-08
1	a	b	x	2024-11-17

3. SELECT dữ liệu từ CSDL

- Câu lệnh SELECT

```
SELECT title, content, author, created_at FROM articles
```



4. KẾT NỐI CSDL TỪ PHP

- Có 3 phong cách kết nối. Mặc định, bài giảng sử dụng phong cách **PHP PDO**

```php
<?php
// Thông tin kết nối đến cơ sở dữ liệu MySQL
$servername = "localhost"; // Địa chỉ máy chủ
$username = "your_username"; // Tên người dùng MySQL
$password = "your_password"; // Mật khẩu MySQL
$dbname = "your_database"; // Tên cơ sở dữ liệu

try {
    // Tạo kết nối
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

    // Thiết lập chế độ báo lỗi
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    echo "Kết nối thành công!";
} catch (PDOException $e) {
    echo "Kết nối thất bại: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
?>
```

5. TRUY VẤN DỮ LIỆU TỪ PHP

- Có 3 phong cách kết nối. Mặc định, bài giảng sử dụng phong cách **PHP PDO**

```php
<?php
    // Kết nối đến cơ sở dữ liệu MySQL
    ...
    // Thực hiện truy vấn SELECT
    $sql = "SELECT title, content, author, created_at FROM articles";
    $stmt = $conn->prepare($sql);
    $stmt->execute();

    // Lấy tất cả dữ liệu
    $results = $stmt->fetchAll(PDO::FETCH_ASSOC);

    // Kiểm tra và hiển thị kết quả
    if ($results) {
        foreach ($results as $row) {
            echo "Tiêu đề: " . htmlspecialchars($row['title']) . "<br>";
            echo "Nội dung: " . nl2br(htmlspecialchars($row['content'])) . "<br>";
            echo "Tác giả: " . htmlspecialchars($row['author']) . "<br>";
            echo "Ngày tạo: " . htmlspecialchars($row['created_at']) . "<br><br>";
        }
    } else {
        echo "Không có dữ liệu.";
    }
} catch (PDOException $e) {
    echo "Lỗi: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
```

6. TẠO TRANG HIỂN THỊ 1 BÀI VIẾT

- view_article.php

```php
<?php
// Thông tin kết nối đến cơ sở dữ liệu MySQL
..
// Lấy ID bài viết từ URL (ví dụ: view_article.php?id=1)
$article_id = isset($_GET['id']) ? (int)$_GET['id'] : 0;

try {
    ...
    // Truy vấn bài viết theo ID
    $sql = "SELECT title, content, author, created_at FROM articles WHERE id = :id";
    $stmt = $conn->prepare($sql);
    $stmt->bindParam(':id', $article_id, PDO::PARAM_INT);
    $stmt->execute();

    // Lấy dữ liệu
    $article = $stmt->fetch(PDO::FETCH_ASSOC);

    // Kiểm tra và hiển thị bài viết
    if ($article) {
        echo "<h1>" . htmlspecialchars($article['title']) . "</h1>";
        echo "<p>" . nl2br(htmlspecialchars($article['content'])) . "</p>";
        echo "<p><strong>Tác giả:</strong> " . htmlspecialchars($article['author']) . "</p>";
        echo "<p><strong>Ngày tạo:</strong> " . htmlspecialchars($article['created_at']) . "</p>";
    } else {
        echo "Bài viết không tồn tại.";
    }
} catch (PDOException $e) {
    echo "Lỗi: " . $e->getMessage();
}

// Đóng kết nối
$conn = null;
?>
```

7. THÊM BÀI VIẾT MỚI

- add_article.php

```php
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Thêm Bài Viết</title>
</head>
<body>
    <h1>Thêm Bài Viết Mới</h1>

    <?php if ($message): ?>
        <p><?php echo htmlspecialchars($message); ?></p>
    <?php endif; ?>

    <form method="post" action="">
        <label for="title">Tiêu đề:</label><br>
        <input type="text" id="title" name="title" required><br><br>

        <label for="content">Nội dung:</label><br>
        <textarea id="content" name="content" required></textarea><br><br>

        <label for="author">Tác giả:</label><br>
        <input type="text" id="author" name="author" required><br><br>

        <input type="submit" value="Thêm Bài Viết">
    </form>
</body>
</html>
```

7. THÊM BÀI VIẾT MỚI

- add_article.php

```php
<?php
// Thông tin kết nối đến cơ sở dữ liệu MySQL
...
// Biến để lưu thông báo
$message = "";
// Xử lý form khi người dùng gửi dữ liệu
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $title = $_POST['title'];
    $content = $_POST['content'];
    $author = $_POST['author'];

    try {
        // Tạo kết nối
        $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
        // Thiết lập chế độ báo lỗi
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        // Chuẩn bị câu lệnh SQL để chèn dữ liệu
        $sql = "INSERT INTO articles (title, content, author) VALUES (:title, :content,
:author)";
        $stmt = $conn->prepare($sql);

        // Liên kết tham số
        $stmt->bindParam(':title', $title);
        $stmt->bindParam(':content', $content);
        $stmt->bindParam(':author', $author);

        // Thực hiện câu lệnh
        $stmt->execute();

        $message = "Bài viết đã được thêm thành công!";
    } catch (PDOException $e) {
        $message = "Lỗi: " . $e->getMessage();
    }
    // Đóng kết nối
    $conn = null;
}
?>
```

8. SQL Injection

- SQL Injection (SQLi) là một trong những lỗ hổng bảo mật phổ biến nhất trong các ứng dụng web. Nó xảy ra khi một ứng dụng cho phép người dùng nhập dữ liệu mà không kiểm tra hoặc xử lý đúng cách, cho phép kẻ tấn công chèn các câu lệnh SQL độc hại vào truy vấn của ứng dụng.
- SQL Injection thường xảy ra do:
  - Thiếu kiểm tra dữ liệu đầu vào: Ứng dụng không kiểm tra hoặc làm sạch dữ liệu mà người dùng nhập vào.
  - Sử dụng câu lệnh SQL động: Khi ứng dụng xây dựng câu lệnh SQL bằng cách nối chuỗi mà không sử dụng tham số.
- Cách thức hoạt động: Khi một kẻ tấn công nhập dữ liệu độc hại vào trường nhập liệu (như form đăng nhập, tìm kiếm, v.v.), kẻ tấn công có thể thay đổi cấu trúc của câu lệnh SQL mà ứng dụng thực hiện. Ví dụ:

```
SELECT * FROM users WHERE username = '$username' AND password = '$password';
```

  - Nếu kẻ tấn công nhập vào:username: admin' --password: (bỏ trống)

```
SELECT * FROM users WHERE username = 'admin' --' AND password = '';
```

8. SQL Injection

- Hệ quả của SQL Injection
  - Truy cập trái phép: Kẻ tấn công có thể truy cập vào các tài khoản người dùng mà không cần mật khẩu.
  - Lộ dữ liệu nhạy cảm: Kẻ tấn công có thể lấy được thông tin cá nhân, mật khẩu, và dữ liệu nhạy cảm khác từ cơ sở dữ liệu.
  - Xóa hoặc thay đổi dữ liệu: Kẻ tấn công có thể xóa hoặc thay đổi dữ liệu trong cơ sở dữ liệu.
  - Tấn công từ chối dịch vụ (DoS): Kẻ tấn công có thể làm cho ứng dụng không hoạt động bằng cách gửi các truy vấn nặng.
- Phòng chống SQL Injection
  - Sử dụng Prepared Statements

```php
<?php
$stmt = $conn->prepare("SELECT * FROM users WHERE username = :username AND password =
:password");
$stmt->bindParam(':username', $username);
$stmt->bindParam(':password', $password);
$stmt->execute();
```

8. SQL Injection

- Phòng chống SQL Injection
  - Sử dụng Prepared Statements
  - Kiểm tra và làm sạch dữ liệu đầu vào: Kiểm tra dữ liệu do người dùng nhập vào và loại bỏ các ký tự không hợp lệ.
  - Sử dụng ORM (Object-Relational Mapping): Sử dụng các thư viện ORM để tương tác với cơ sở dữ liệu, giúp tự động hóa các biện pháp bảo vệ.
  - Giới hạn quyền truy cập cơ sở dữ liệu: Chỉ cung cấp quyền tối thiểu cho các tài khoản cơ sở dữ liệu mà ứng dụng sử dụng.
  - Theo dõi và ghi lại hoạt động: Ghi lại các hoạt động truy cập cơ sở dữ liệu để phát hiện và phân tích các hành vi bất thường.

9. TẠO HÀM KẾT NỐI CSDL

- db_connection.php

```php
<?php
function connectDatabase($servername, $username, $password, $dbname) {
    try {
        // Tạo kết nối
        $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

        // Thiết lập chế độ báo lỗi
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        return $conn; // Trả về đối tượng kết nối
    } catch (PDOException $e) {
        die("Kết nối thất bại: " . $e->getMessage());
    }
}

// Ví dụ sử dụng hàm
$servername = "localhost"; // Địa chỉ máy chủ
$username = "your_username"; // Tên người dùng MySQL
$password = "your_password"; // Mật khẩu MySQL
$dbname = "your_database"; // Tên cơ sở dữ liệu

$conn = connectDatabase($servername, $username, $password, $dbname);

// Đừng quên đóng kết nối khi không cần thiết
// $conn = null;
?>
```

9. TẠO HÀM KẾT NỐI CSDL

- Sử dụng hàm đã định nghĩa kết nối ở tệp khác

```php
<?php
require 'db_connection.php';

// Bây giờ bạn có thể sử dụng $conn để thực hiện các truy vấn
$sql = "SELECT * FROM articles";
$stmt = $conn->prepare($sql);
$stmt->execute();
$results = $stmt->fetchAll(PDO::FETCH_ASSOC);

// Hiển thị kết quả
foreach ($results as $row) {
    echo "Tiêu đề: " . htmlspecialchars($row['title']) . "<br>";
}
?>
```

10. ĐẢM BẢO TÍNH HỢP LỆ CỦA FORM THÊM

- Để thực hiện xác thực (validation) cho form thêm bài viết mới trong PHP, kiểm tra dữ liệu đầu vào trước khi lưu vào cơ sở dữ liệu.

```php
<?php
// Xử lý form khi người dùng gửi dữ liệu
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $title = trim($_POST['title']);
    $content = trim($_POST['content']);
    $author = trim($_POST['author']);

    // Xác thực dữ liệu đầu vào
    if (empty($title)) {
        $errors[] = "Tiêu đề không được để trống.";
    }
    if (empty($content)) {
        $errors[] = "Nội dung không được để trống.";
    }
    if (empty($author)) {
        $errors[] = "Tác giả không được để trống.";
    }
    ...
}
?>
```

```php
<!DOCTYPE html>
<html lang="vi">
<head>
    ...
</head>
<body>
    <h1>Thêm Bài Viết Mới</h1>

    <?php if ($message): ?>
        <p><?php echo htmlspecialchars($message); ?></p>
    <?php endif; ?>

    <?php if ($errors): ?>
        <ul>
            <?php foreach ($errors as $error): ?>
                <li style="color: red;"><?php echo htmlspecialchars($error); ?></li>
            <?php endforeach; ?>
        </ul>
    <?php endif; ?>

    <form method="post" action="">
        <label for="title">Tiêu đề:</label><br>
        <input type="text" id="title" name="title" value="<?php echo isset($title) ? htmlspecialchars($title) : ''; ?>"
required><br><br>

        <label for="content">Nội dung:</label><br>
        <textarea id="content" name="content" required><?php echo isset($content) ? htmlspecialchars($content) : '';
?></textarea><br><br>

        <label for="author">Tác giả:</label><br>
        <input type="text" id="author" name="author" value="<?php echo isset($author) ? htmlspecialchars($author) : '';
?>" required><br><br>

        <input type="submit" value="Thêm Bài Viết">
    </form>
</body>
</html>
```

11. PHÒNG NGỪA TẤN CÔNG XSS

- **Cross-Site Scripting (XSS)** là một loại lỗ hổng bảo mật cho phép kẻ tấn công chèn mã JavaScript độc hại vào một trang web, gây ra các hậu quả nghiêm trọng cho người dùng như đánh cắp cookie, thông tin đăng nhập, hoặc thực hiện các hành động không mong muốn từ phía người dùng. Dưới đây là một số phương pháp hiệu quả để phòng ngừa tấn công XSS:
  - Xử lý và làm sạch dữ liệu đầu vào
  - Sử dụng htmlspecialchars()
  - Sử dụng Content Security Policy (CSP)
  - Tránh sử dụng các thuộc tính HTML có thể dẫn đến XSS
  - Sử dụng các thư viện xác thực
  - Xác thực và phân quyền người dùng
  - Kiểm tra và cập nhật thường xuyên
  - Sử dụng các công cụ bảo mật

```
<!-- Tránh -->
<button onclick="alert('XSS!')">Click me</button>

<!-- Thay vào đó, sử dụng sự kiện trong JavaScript -->
<button id="myButton">Click me</button>
<script>
    document.getElementById('myButton').addEventListener('click',
function() {
        alert('Button clicked!');
    });
</script>
```

12. CHUYỂN HƯỚNG SAU KHI THÊM BÀI VIẾT

- Để thực hiện chuyển hướng (**redirect**) người dùng sau khi thêm một bài viết mới vào cơ sở dữ liệu, bạn có thể sử dụng hàm **header()** trong PHP. Dưới đây là cách thực hiện điều này trong mã PHP của bạn.

```php
<?php
// Thông tin kết nối đến cơ sở dữ liệu MySQL
...
// Xử lý form khi người dùng gửi dữ liệu
If ($_SERVER[“REQUEST_METHOD”] == “POST”) {
    ...
            // Chuẩn bị câu lệnh SQL để chèn dữ liệu
            $sql = “INSERT INTO articles (title, content, author) VALUES (:title, :content, :author)”;
            $stmt = $conn->prepare($sql);
            // Liên kết tham số
            $stmt->bindParam(':title', $title);
            $stmt->bindParam(':content', $content);
            $stmt->bindParam(':author', $author);
            // Thực hiện câu lệnh
            $stmt->execute();
            // Chuyển hướng sau khi thêm thành công
            header("Location: success.php"); // Đường dẫn đến trang thành công
            exit; // Dừng thực thi để tránh việc gửi thêm dữ liệu
        } catch (PDOException $e) {
            $message = "Lỗi: " . $e->getMessage();
        }
        // Đóng kết nối
        $conn = null;
    }
}
?>
```

13. HÀM LẤY ARTICLE THEO ID

- Định nghĩa hàm

```php
<?php
// Thông tin kết nối đến cơ sở dữ liệu MySQL
...
function getArticleById($id) {
    global $servername, $username, $password, $dbname; // Sử dụng biến
toàn cục

    try {
        // Tạo kết nối
        $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        // Chuẩn bị câu lệnh SQL
        $sql = "SELECT * FROM articles WHERE id = :id";
        $stmt = $conn->prepare($sql);

        // Liên kết tham số
        $stmt->bindParam(':id', $id, PDO::PARAM_INT);

        // Thực hiện câu lệnh
        $stmt->execute();

        // Lấy kết quả
        $article = $stmt->fetch(PDO::FETCH_ASSOC);

        return $article; // Trả về bài viết
    } catch (PDOException $e) {
        echo "Lỗi: " . $e->getMessage();
        return null; // Trả về null nếu có lỗi
    } finally {
        // Đóng kết nối
        $conn = null;
    }
}

...
?>
```

13. HÀM LẤY ARTICLE THEO ID

- Sử dụng hàm

```php
<?php
...
// Ví dụ sử dụng hàm
$id = 1; // Thay đổi ID theo nhu cầu
$article = getArticleById($id);

if ($article) {
    echo "<h2>" . htmlspecialchars($article['title']) . "</h2>";
    echo "<p>" . nl2br(htmlspecialchars($article['content'])) . "</p>";
    echo "<p><em>Người viết: " . htmlspecialchars($article['author']) .
"</em></p>";
} else {
    echo "Bài viết không tồn tại.";
}
?>
```

```php
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chỉnh Sửa Bài Viết</title>
</head>
<body>
    <h1>Chỉnh Sửa Bài Viết</h1>

    <?php if ($message): ?>
        <p><?php echo htmlspecialchars($message); ?></p>
    <?php endif; ?>

    <?php if ($errors): ?>
        <ul>
            <?php foreach ($errors as $error): ?>
                <li style="color: red;"><?php echo htmlspecialchars($error); ?></li>
            <?php endforeach; ?>
        </ul>
    <?php endif; ?>

    <form method="post" action="">
        <input type="hidden" name="id" value="<?php echo htmlspecialchars($id); ?>">

        <label for="title">Tiêu đề:</label><br>
        <input type="text" id="title" name="title" value="<?php echo htmlspecialchars($title); ?>" required><br><br>

        <label for="content">Nội dung:</label><br>
        <textarea id="content" name="content" required><?php echo htmlspecialchars($content); ?></textarea><br><br>

        <label for="author">Tác giả:</label><br>
        <input type="text" id="author" name="author" value="<?php echo htmlspecialchars($author); ?>" required><br><br>

        <input type="submit" value="Cập Nhật Bài Viết">
    </form>
</body>
</html>
```

14. TẠO FORM EDIT ARTICLE

- **edit_article**.php

```php
<?php
// Nếu không có lỗi, thực hiện cập nhật bài viết
if (empty($errors)) {
    try {
        $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "UPDATE articles SET title = :title, content = :content, author =
:author WHERE id = :id";
        $stmt = $conn->prepare($sql);
        $stmt->bindParam(':title', $title);
        $stmt->bindParam(':content', $content);
        $stmt->bindParam(':author', $author);
        $stmt->bindParam(':id', $id, PDO::PARAM_INT);
        $stmt->execute();

        $message = "Bài viết đã được cập nhật thành công!";
    } catch (PDOException $e) {
        $message = "Lỗi: " . $e->getMessage();
    } finally {
        $conn = null;
    }
}
```

15. XÓA BÀI VIẾT

- **delete_article.php**

```php
<?php
// Xử lý khi người dùng xác nhận xóa
if (isset($_GET['id'])) {
    $id = intval($_GET['id']);

    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        try {
            $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

            // Câu lệnh SQL để xóa bài viết
            $sql = "DELETE FROM articles WHERE id = :id";
            $stmt = $conn->prepare($sql);
            $stmt->bindParam(':id', $id, PDO::PARAM_INT);
            $stmt->execute();

            $message = "Bài viết đã được xóa thành công!";
            header("Location: articles.php"); // Chuyển hướng đến trang danh sách bài
viết
            exit;
        } catch (PDOException $e) {
            $message = "Lỗi: " . $e->getMessage();
        } finally {
            $conn = null;
        }
    }
} else {
    $message = "ID bài viết không hợp lệ.";
}
```

15. XÓA BÀI VIẾT

- **delete_article.php**: delete thông qua POST và xác nhận trước khi xóa

```
<a href="#" onclick="confirmDelete(<?php echo $article['id']; ?>)">Xóa</a>

<script>
function confirmDelete(articleId) {
  if (confirm("Bạn có chắc chắn muốn xóa bài viết này?")) {
    // Gửi yêu cầu xóa bằng AJAX hoặc chuyển hướng đến trang xử lý xóa
    window.location.href = "delete.php?id=" + articleId;
  }
}
</script>
```

15. XÓA BÀI VIẾT

- **delete_article.php**: delete thông qua POST và xác nhận trước khi xóa

```php
<?php
require_once 'connect.php'; // Kết nối CSDL

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $articleId = $_POST['id'];

  // Kiểm tra xem ID có hợp lệ hay không

  // Thực hiện truy vấn xóa bài viết
  $sql = "DELETE FROM articles WHERE id = :id";
  $stmt = $pdo->prepare($sql);
  $stmt->execute(['id' => $articleId]);

  // Chuyển hướng về trang danh sách bài viết
  header('Location: list.php');
  exit;
}
?>

<form method="POST" action="">
  <input type="hidden" name="id" value="<?php echo $_GET['id']; ?>">
  <p>Bạn có chắc chắn muốn xóa bài viết này?</p>
  <button type="submit">Xóa</button>
  <a href="list.php">Hủy</a>
</form>
```

16. TĂNG TỐC TRUY VẤN

- Một số phương pháp để tối ưu hóa tốc độ truy vấn và tăng hiệu suất ứng dụng khi làm việc với cơ sở dữ liệu:
  - Sử dụng giới hạn Truy vấn (LIMIT)

```
SELECT * FROM articles LIMIT 10 OFFSET 0;
```

  - Phân trang Dữ liệu: Khi hiển thị danh sách bài viết, có thể chia nhỏ dữ liệu thành các trang. Sử dụng LIMIT kết hợp với OFFSET để lấy dữ liệu cho từng trang.

```php
<?php
$page = isset($_GET['page']) ? (int)$_GET['page'] : 1;
$limit = 10; // Số bản ghi trên mỗi trang
$offset = ($page - 1) * $limit;

// Truy vấn với LIMIT và OFFSET
$sql = "SELECT * FROM articles LIMIT :limit OFFSET :offset";
$stmt = $conn->prepare($sql);
$stmt->bindParam(':limit', $limit, PDO::PARAM_INT);
$stmt->bindParam(':offset', $offset, PDO::PARAM_INT);
```

16. TĂNG TỐC TRUY VẤN

- Một số phương pháp để tối ưu hóa tốc độ truy vấn và tăng hiệu suất ứng dụng khi làm việc với cơ sở dữ liệu:
  - Chỉ lấy các cột cần thiết

```
SELECT id, title, author FROM articles LIMIT 10;
```

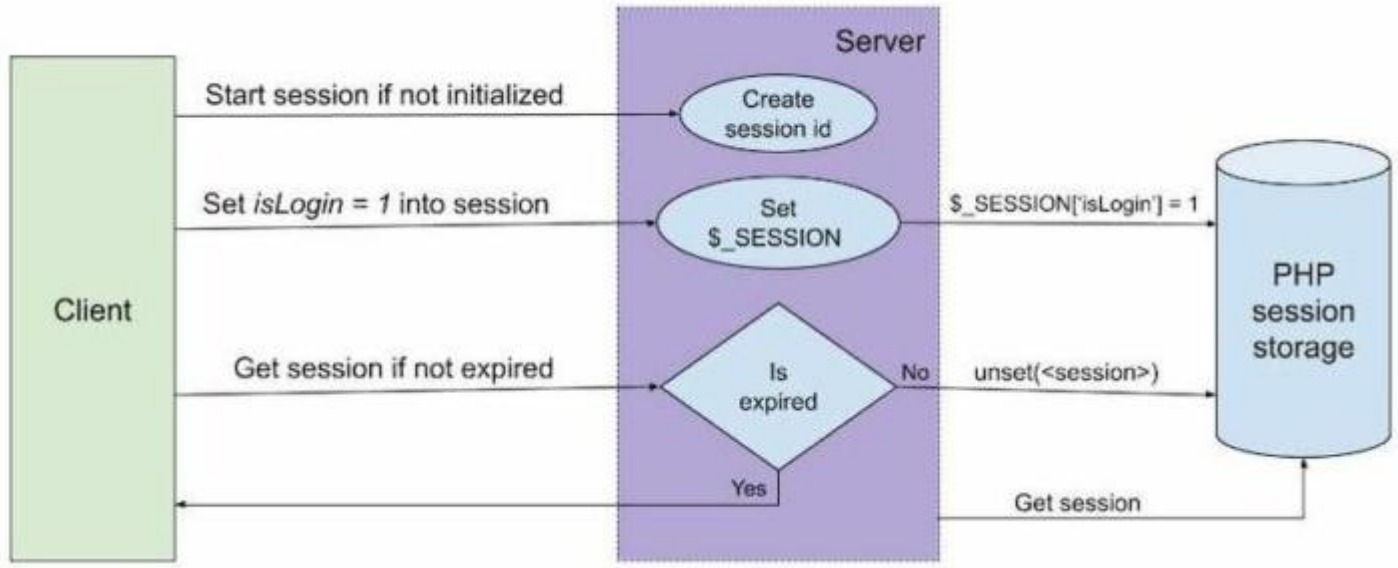
  - Tối ưu hóa cơ sở Dữ liệu: Tạo chỉ số (index) cho các cột thường xuyên được sử dụng trong điều kiện WHERE, ORDER BY, hoặc JOIN. Điều này giúp tăng tốc độ truy vấn.

```
CREATE INDEX idx_title ON articles(title);
```

  - Sử dụng cache
  - Truy vấn không đồng bộ
  - Giá trị mặc định và tránh NULL

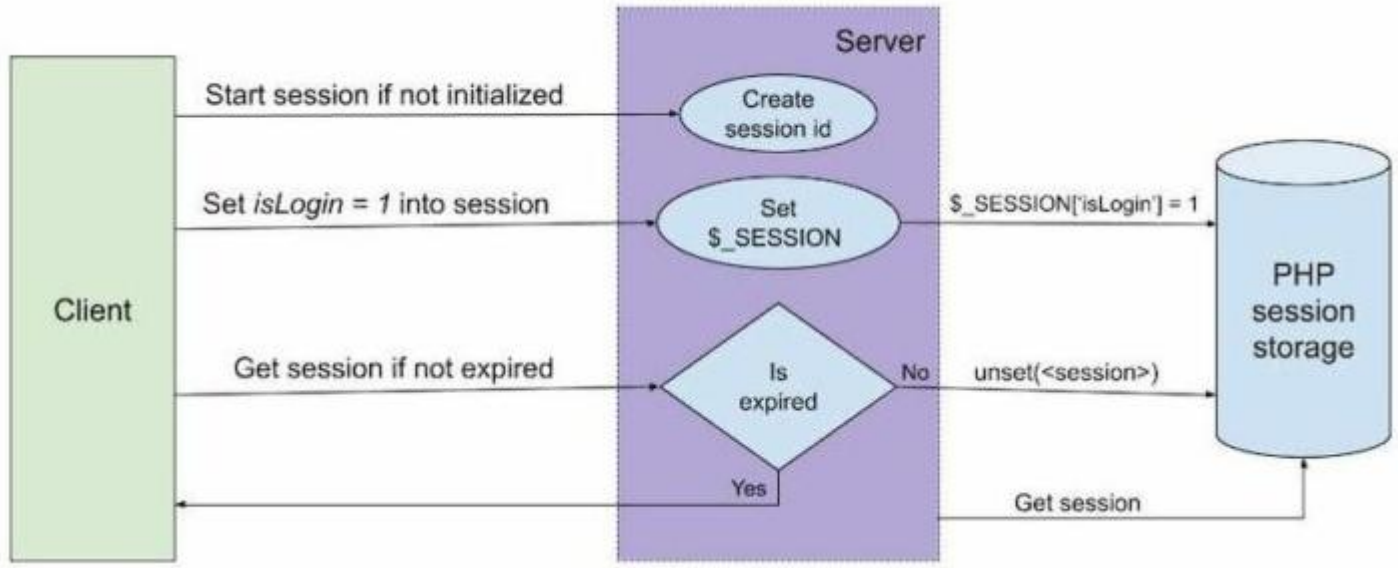
17. LOGIN và SESSION

- **Khái niệm Session**: Session là một cơ chế cho phép lưu trữ thông tin của người dùng trên máy chủ web trong suốt phiên làm việc của họ. Thông tin này có thể được truy cập và sử dụng trên nhiều trang web khác nhau mà không cần phải gửi lại dữ liệu qua lại giữa trình duyệt và máy chủ.
- **Mục đích sử dụng**: Session thường được sử dụng để quản lý trạng thái đăng nhập, lưu trữ giỏ hàng, cá nhân hóa trải nghiệm người dùng, v.v.



17. LOGIN và SESSION

- **Cách thức hoạt động:**
  - Khi người dùng truy cập website, máy chủ sẽ tạo một Session ID duy nhất cho người dùng đó.
  - Session ID này được lưu trữ trong một cookie trên trình duyệt của người dùng.
  - Mỗi khi người dùng gửi yêu cầu đến máy chủ, trình duyệt sẽ gửi kèm theo Session ID.
  - Máy chủ sử dụng Session ID để xác định phiên làm việc của người dùng và truy cập vào dữ liệu Session tương ứng.



17. LOGIN và SESSION

- **Bắt đầu session:**
  - Sử dụng hàm session_start() ở đầu mỗi trang PHP cần sử dụng Session. Hàm này sẽ khởi tạo một Session mới hoặc tiếp tục Session hiện tại dựa trên Session ID.
  - Kiểm tra Session đã tồn tại: Có thể kiểm tra xem Session đã tồn tại hay chưa bằng cách kiểm tra biến $_SESSION.

```php
<?php
  session_start();

  if (isset($_SESSION['username'])) {
    echo "Xin chào, " . $_SESSION['username'];
  } else {
    echo "Bạn chưa đăng nhập.";
  }
?>
```

17. LOGIN và SESSION

- **Lưu trữ dữ liệu trong Session:**
  - Sử dụng biến toàn cục $_SESSION (là một mảng kết hợp) để lưu trữ dữ liệu.

```php
<?php
  session_start();

  $_SESSION['username'] = 'johndoe';
  $_SESSION['email'] = 'johndoe@example.com';
  $_SESSION['role'] = 'admin';
?>
```

- **Truy cập dữ liệu trong Session**
  - Sử dụng $_SESSION['key'] để truy cập dữ liệu trong Session.

```php
<?php
  session_start();

  echo "Tên người dùng: " . $_SESSION['username'];
?>
```

17. LOGIN và SESSION

- **Xóa dữ liệu trong Session:**
  - Xóa 1 biến session.

```php
<?php
  session_start();

  unset($_SESSION['username']);
?>
```

  - Xóa toàn bộ session.

```php
<?php
  session_start();

  session_destroy();
?>
```

```php
<?php
#Ví dụ logout
  session_start();

  session_destroy();

  // Chuyển hướng người dùng đến trang login
  header('Location: login.php');
  exit;
?>
```

17. LOGIN và SESSION

- **Xây dựng FORM login:**
  - Để tạo form login, chúng ta sử dụng HTML với các trường username và password, kết hợp với CSS để tạo giao diện. Phương thức POST được sử dụng để gửi dữ liệu form đến server nhằm đảm bảo bảo mật hơn so với GET (không hiển thị thông tin đăng nhập trên URL).

```
<!DOCTYPE html>
<html>
<head>
  <title>Form Login</title>
  <style>
    /* CSS để tạo giao diện cho form */
  </style>
</head>
<body>
  <h1>Đăng nhập</h1>
  <form method="POST" action="login.php">
    <label for="username">Tên đăng nhập:</label><br>
    <input type="text" id="username" name="username" required><br><br>
    <label for="password">Mật khẩu:</label><br>
    <input type="password" id="password" name="password" required><br><br>
    <button type="submit">Đăng nhập</button>
  </form>
</body>
</html>
```

17. LOGIN và SESSION

- **Xử lý dữ liệu FORM login:**
  - File **login.php** sẽ nhận dữ liệu từ form và xử lý đăng nhập:

```php
<?php
session_start(); // Bắt đầu session

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

  // Kiểm tra dữ liệu đầu vào có hợp lệ hay không
  if (empty($username) || empty($password)) {
    echo "Vui lòng nhập đầy đủ thông tin.";
    exit;
  }
  // Kết nối cơ sở dữ liệu
  require_once 'connect.php';
  // Truy vấn cơ sở dữ liệu để kiểm tra username và password
  // Sử dụng prepared statements để ngăn chặn SQL injection
  $sql = "SELECT * FROM users WHERE username = :username";
  $stmt = $pdo->prepare($sql);
  $stmt->execute(['username' => $username]);
  $user = $stmt->fetch();

  if ($user && password_verify($password, $user['password'])) {
    // Đăng nhập thành công
    $_SESSION['user_id'] = $user['id'];
    $_SESSION['username'] = $user['username'];
    // ... lưu thêm thông tin cần thiết khác

    // Chuyển hướng người dùng đến trang chủ
    header('Location: index.php');
    exit;
  } else {
    // Đăng nhập thất bại
    echo "Tên đăng nhập hoặc mật khẩu không đúng.";
  }
}
?>
```

17. LOGIN và SESSION

- **Tăng bảo mật khi login:**
  - **Sử dụng Prepared Statements**:
    - SQL Injection là một kỹ thuật tấn công lợi dụng lỗ hổng trong việc xử lý dữ liệu đầu vào của ứng dụng để chèn mã SQL độc hại vào câu truy vấn, từ đó chiếm quyền điều khiển cơ sở dữ liệu.
    - Prepared Statements là một cách thức thực thi câu truy vấn SQL an toàn hơn, giúp ngăn chặn SQL Injection bằng cách tách biệt dữ liệu đầu vào khỏi câu truy vấn.
    - Ví dụ: Thay vì ghép trực tiếp dữ liệu username vào câu truy vấn như $sql = "SELECT * FROM users WHERE username = '$username'";, chúng ta nên sử dụng Prepared Statements như sau:

```php
<?php
$sql = "SELECT * FROM users WHERE username = :username";
$stmt = $pdo->prepare($sql);
$stmt->execute(['username' => $username]);
```

17. LOGIN và SESSION

- **Tăng bảo mật khi login:**
  - **Mã hóa mật khẩu**:
    - Không bao giờ lưu mật khẩu dưới dạng plain text (text thô) trong cơ sở dữ liệu.Sử dụng hàm password_hash() để mã hóa mật khẩu trước khi lưu trữ.
    - Hàm này sử dụng thuật toán mã hóa mạnh và tạo ra một chuỗi hash duy nhất cho mỗi mật khẩu.
    - Ví dụ:

```php
<?php
$hashedPassword = password_hash($password,
PASSWORD_DEFAULT);
// Lưu $hashedPassword vào cơ sở dữ liệu

if (password_verify($password, $hashedPassword)) {
    // Đăng nhập thành công
  } else {
    // Đăng nhập thất bại
  }
```

17. LOGIN và SESSION

- **Tăng bảo mật khi login:**
  - **Sử dụng HTTPS**:
    - HTTPS mã hóa dữ liệu truyền tải giữa trình duyệt và máy chủ, giúp ngăn chặn việc nghe lén và đánh cắp thông tin đăng nhập.
    - Đảm bảo website của bạn có chứng chỉ SSL và sử dụng giao thức HTTPS.
  - **Ngăn chặn Brute-force attack**:
    - Brute-force attack là kỹ thuật tấn công bằng cách thử nhiều mật khẩu khác nhau cho đến khi tìm được mật khẩu đúng.
    - Giới hạn số lần đăng nhập sai: Cho phép người dùng đăng nhập sai một số lần nhất định trong một khoảng thời gian nhất định. Sau đó, khóa tài khoản hoặc yêu cầu xác thực bổ sung (ví dụ: captcha).
    - Ghi lại địa chỉ IP: Ghi lại địa chỉ IP của người dùng đăng nhập và chặn các IP có nhiều lần đăng nhập sai.
  - **Sử dụng Captcha**:
    - Captcha là một hình thức kiểm tra để phân biệt người dùng là người thật hay máy tính.
    - Sử dụng Captcha trong form login để ngăn chặn các bot tự động đăng nhập.
    - Có nhiều loại Captcha khác nhau, ví dụ: reCAPTCHA của Google.

17. LOGIN và SESSION

- **Giới hạn truy cập khi chưa login:**
  - Trong nhiều ứng dụng web, có những trang hoặc chức năng chỉ dành cho người dùng đã đăng nhập, ví dụ như trang quản trị, trang thông tin cá nhân, chức năng mua hàng,... Để giới hạn truy cập vào những khu vực này, ta cần kiểm tra trạng thái đăng nhập của người dùng.
  - **Kiểm tra session**:
    - Session là một cơ chế phổ biến để quản lý trạng thái đăng nhập. Sau khi người dùng đăng nhập thành công, thông tin đăng nhập (như user ID, username, role) thường được lưu trữ trong Session.
    - Để kiểm tra xem người dùng đã đăng nhập hay chưa, ta kiểm tra sự tồn tại của các biến Session này. Ví dụ, nếu lưu user ID trong $_SESSION['user_id'], ta có thể kiểm tra như sau:

```php
<?php
session_start();

if (!isset($_SESSION['user_id'])) {
  // Người dùng chưa đăng nhập
  // Chuyển hướng đến trang login
  header('Location: login.php');
  exit;
}

// Người dùng đã đăng nhập, tiếp tục xử lý
// ...
?>
```

17. LOGIN và SESSION

- **Giới hạn truy cập khi chưa login:**
  - Trong nhiều ứng dụng web, có những trang hoặc chức năng chỉ dành cho người dùng đã đăng nhập, ví dụ như trang quản trị, trang thông tin cá nhân, chức năng mua hàng,... Để giới hạn truy cập vào những khu vực này, ta cần kiểm tra trạng thái đăng nhập của người dùng.
  - **Bảo vệ trang admin**:
    - Giả sử ta có trang quản trị admin.php chỉ dành cho người dùng có quyền admin. Ta có thể kiểm tra quyền truy cập như sau:

```php
<?php
session_start();

if (!isset($_SESSION['user_id'])) {
  // Người dùng chưa đăng nhập
  header('Location: login.php');
  exit;
}

// Kiểm tra quyền admin
if ($_SESSION['role'] !== 'admin') {
  // Người dùng không có quyền admin
  echo "Bạn không có quyền truy cập trang này.";
  exit;
}

// Người dùng có quyền admin, hiển thị trang quản trị
// ...
?>
```

“Câu hỏi & Thảo luận”

THE END!