Лекция 8 - Основы работы с базами данных при создании веб-сайтов и веб-приложений

Введение

Базы данных являются критически важными компонентами большинства веб-сайтов, поскольку они позволяют хранить, извлекать и управлять данными, которые приложение использует в своей работе.

Что такое база данных?

База данных – это организованная коллекция данных. В контексте веб-разработки данные обычно включают информацию о пользователях, продуктах, транзакциях и многое другое. Данные хранятся таким образом, чтобы их можно было легко доступать, управлять и обновлять.

Роль базы данных в веб-разработке

В веб-разработке база данных используется для хранения информации, необходимой для работы сайта или приложения, такой как пользовательские данные, информация о продуктах, данные о заказах и многое другое.

Реляционные и нереляционные базы данных

Базы данных бывают реляционными (SQL) и нереляционными (NoSQL).

Реляционные базы данных (SQL):

- Основаны на предположении, что данные связаны друг с другом.
- Данные организованы в таблицы с строгой схемой.
- Примеры: MySQL, PostgreSQL, SQLite.

Нереляционные базы данных (NoSQL):

- Предназначены для хранения данных в более свободной и гибкой форме.
- Поддерживают различные модели данных: документо-ориентированную, графовую, ключ-значение и т. д.
- Примеры: MongoDB, Redis, Cassandra.

Основные концепции баз данных

- **Таблицы:** В реляционных базах данных информация хранится в виде таблиц, которые состоят из строк и столбцов.
- **Запись (строка):** Конкретная строка таблицы, содержащая уникальные данные.
- **Поле (столбец):** Столбец в таблице, представляющий определенный тип информации.
- **Первичный ключ (Primary Key):** Уникальный идентификатор для каждой записи в таблице.
- **Внешний ключ (Foreign Key):** Поле в одной таблице, которое связано с первичным ключом другой таблицы.
- **Индекс:** Специальная структура данных, которая ускоряет выборку данных из таблицы.

- **SQL (Structured Query Language):** Язык программирования, используемый для управления и манипулирования данными в реляционных базах данных.

Ключевые операции SQL

- SELECT для выборки данных
- INSERT для вставки данных
- UPDATE для обновления существующих данных
- DELETE для удаления данных

Основы CRUD операций

CRUD — это аббревиатура, которая обозначает четыре основные функции, используемые в работе с базами данных:

- Create (создание)
- Read (чтение)
- Update (обновление)
- Delete (удаление)

Жизненный цикл веб-приложения и база данных

1. **Планирование**: Определение структуры базы данных, схемы, таблиц и отношений между ними.
2. **Разработка:**
   - Моделирование сущностей и их взаимодействий.
   - Создание базы данных и таблиц.
   - Настройка связей и ограничений целостности данных.
3. **Тестирование:** Проверка работоспособности базы данных, запросов и транзакций.
4. **Деплоймент:** Размещение базы данных на сервере.
5. **Мониторинг и оптимизация:** Отслеживание производительности и по необходимости оптимизация запросов и структуры базы данных.

Разработка веб-приложения с базой данных

Шаг 1: Проектирование базы данных

- Определите сущности и их атрибуты, которые будут представляться таблицами и полями.
- Определите отношения между этими сущностями.
- Разработайте схему базы данных с использованием ER-диаграммы (сущность-связь).

Шаг 2: Выбор СУБД

- Выберите между реляционной и нереляционной СУБД на основе требований проекта.
- Учтите масштабируемость, производительность и удобство использования.

Шаг 3: Реализация базы данных

- Создайте базу данных и таблицы согласно схеме.

- Напишите SQL-запросы для создания таблиц и ограничений.
- Заполните таблицы начальными данными, если необходимо.

Шаг 4: Работа с базой данных в приложении

- Используйте язык программирования, например PHP, Python, Java или JavaScript (Node.js), для подключения к базе данных.
- Создайте запросы SQL для вставки, обновления, удаления и выборки данных.
- Разработайте функции и классы для работы с данными (например, модели в MVC-фреймворках).

Шаг 5: Безопасность базы данных

- Используйте шифрование для хранения чувствительных данных.
- Предотвращайте SQL-инъекции с помощью подготовленных выражений и параметризованных запросов.
- Обеспечьте резервное копирование и восстановление данных.

Лучшие практики

- Избегайте избыточности данных с помощью нормализации базы данных.
- Используйте транзакции для группировки нескольких операций.
- Мониторьте и анализируйте запросы к базе данных, чтобы выявить и оптимизировать узкие места.
- Обеспечьте масштабируемость приложения, используя кеширование и репликацию данных.
- Документируйте структуру базы данных и изменения в ней.

Пример подключения к базе данных MySQL с PHP:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "my_database";

// Создаем соединение
$conn = new mysqli($servername, $username, $password, $dbname);

// Проверяем соединение
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Выполнение SELECT запроса с PHP

```
// Подключение к базе данных осуществляется через объект $mysqli.
// Отправляем запрос к базе данных на выборку всех записей из таблицы users.
$result = $mysqli->query("SELECT * FROM users");

// Используем цикл while
```

Пример создания таблицы с использованием SQL:

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

Пример вставки данных в таблицу с PHP и MySQL:

```
<?php
$sql = "INSERT INTO users (username, password, email) VALUES (?, ?, ?)";

$stmt = $conn->prepare($sql);

// Привязываем параметры к подготовленному выражению
$stmt->bind_param("sss", $username, $password, $email);

// Устанавливаем параметры и выполняем
$username = "newuser";
$password = password_hash("userpassword", PASSWORD_DEFAULT); // Хэшируем пароль
$email = "user@example.com";
$stmt->execute();

echo "New record created successfully";

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

Пример выборки данных с использованием PHP и MySQL:

```
<?php
$sql = "SELECT id, username, email FROM users";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // Вывод данных каждой строки
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["username"]. " - Email: " . $row["email"]. "<br>";
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

Пример обновления данных в таблице:

```
UPDATE users SET email = 'newemail@example.com' WHERE id = 1;
```

Пример удаления данных из таблицы:

```
DELETE FROM users WHERE id = 1;
```

Безопасность: Предотвращение SQL-инъекций с PDO (PHP Data Objects):

```
<?php
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username AND password = :password");
$stmt->execute(array(":username" => $username, ":password" => $password));
$user = $stmt->fetch();
?>
```

Эти примеры являются основами работы с базами данных в контексте веб-разработки и предназначены для иллюстрации базовых операций CRUD (Создание, Чтение, Обновление и Удаление). В реальных проектах эти операции будут более сложными и могут включать дополнительные уровни абстракции, такие как ORM (Object-Relational Mapping).

Работа с базой данных — это ключевая часть разработки веб-приложений, требующая знаний и внимательности к деталям. Используя реляционные или нереляционные СУБД в зависимости от нужд проекта, разработчик должен уметь эффективно создавать, запрашивать, изменять и удалять данные. Безопасность должна быть на первом месте, и регулярное обслуживание базы данных поможет обеспечить стабильность и производительность веб-сайта или приложения.

Контрольные вопросы

1. Что такое SQL и для чего он используется?
2. Назовите четыре основные операции CRUD, выполняемые с использованием SQL.
3. Каковы различия между реляционными и нереляционными базами данных?
4. Какие типы связей могут быть установлены между таблицами в реляционных базах данных?
5. Какие существуют нормальные формы, и для чего они необходимы?
6. Какие методы можно использовать для предотвращения SQL-инъекций?
7. Почему важно шифровать конфиденциальные данные в базе данных?
8. Что такое индекс в контексте баз данных и для чего он используется?
9. Как кеширование может улучшить производительность веб-приложения?

Практическое задание
Задание 1: Создание и заполнение базы данных

Создайте базу данных webstore и таблицу products с полями id, name, description, price, и in_stock. Добавьте в таблицу несколько записей.

Sql

```
-- SQL команды для создания базы данных и таблицы
CREATE DATABASE webstore;
USE webstore;

CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    in_stock BOOLEAN DEFAULT TRUE
);

-- SQL команды для вставки данных в таблицу
INSERT INTO products (name, description, price, in_stock) VALUES
('Product 1', 'Description for product 1', 19.99, TRUE),
('Product 2', 'Description for product 2', 29.99, FALSE),
('Product 3', 'Description for product 3', 39.99, TRUE);
```

Задание 2: Выборка данных
Напишите SQL запрос для выборки всех товаров, которые есть в наличии (in_stock = TRUE).

Задание 3: Обновление данных
Напишите SQL запрос для обновления цены товара с id 2.

Задание 4: Удаление данных
Напишите SQL запрос для удаления товара из таблицы products, у которого цена меньше 20.

Задание 5: Предотвращение SQL-инъекций
Допишите PHP скрипт для безопасной выборки данных из таблицы users, используя пользовательский ввод.