Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Базы данных

Лабораторная работа № 4

Тема: «Демонстрация команд DDL, DML и INNER JOIN»

Выполнил:

студент группы РИС–23–2б

Борисов Н.А.

Проверил:

доцент кафедры ИТАС

Петренко А.А.

г. Пермь – 2024

**ВВЕДЕНИЕ**

**Цель:** разработать веб–страничку, демонстрирующую работу команд SQL: DDL, DML и INNER JOIN.

**Задачи:**

1. Изучить принципы работы с командами SQL:
   1. DDL;
   2. DML;
   3. INNER JOIN.
2. Разработать дизайн веб–страницы:
   1. Продумать, какие блоки будут на сайте и как их лучше разместить;
   2. Сделать разметку HTML и задать стили CSS.
3. Подключить логику обработки данных между клиентом и сервером с помощью JS и PHP.

**ЭТАПЫ ВЫПОЛНЕНИЯ РАБОТЫ**

**Принципы работы с командами DDL и их внедрение на веб–страницу.**

DDL (Data Definition Language) – это набор команд SQL, предназначенных для определения структуры базы данных. Команды DDL включают CREATE, ALTER, DROP и TRUNCATE, которые позволяют создавать и модифицировать таблицы, удалять или очищать их содержимое.

CREATE – используется для создания новых объектов, таких как таблицы и базы данных. В команде CREATE TABLE задаются названия столбцов и их типы данных.

ALTER – позволяет изменять уже существующие таблицы (добавлять/удалять столбцы или изменять их типы данных).

DROP – удаляет таблицу или базу данных целиком. В отличие от DELETE, DROP полностью убирает объект, включая его структуру.

TRUNCATE – быстро очищает таблицу, удаляя все строки, но сохраняя ее структуру.

Для внедрения команд DDL на веб–сайт необходимо создать интерфейс, позволяющий пользователю взаимодействовать с базой данных через графический интерфейс. Например, создать выпадающие списки и текстовые поля для параметров. Таким образом, для команды CREATE пользователь вводит название таблицы и определяет её столбцы и типы данных. Для ALTER выбирается таблица, и задаются новые параметры для модификации.

Каждая команда будет инициализироваться кнопкой, которая запускает JavaScript–функцию для отправки данных на сервер, используя fetch и JSON для передачи параметров из формы. PHP–скрипты для выполнения SQL. На сервере PHP–скрипты принимают параметры и выполняют соответствующую команду DDL, используя соединение с базой данных MySQL.

**Принципы работы с командами DML и их внедрение на сайт.**

DML (Data Manipulation Language) – это подмножество SQL–команд, предназначенных для управления данными в базе. Основные команды DML включают INSERT, UPDATE, DELETE и SELECT, которые позволяют добавлять, изменять, удалять и получать данные из таблиц. Эти команды работают с содержимым таблиц, а не с их структурой, что отличает их от команд DDL.

INSERT - добавляет новые строки в таблицу. В команде INSERT INTO задается имя таблицы, список столбцов и значения, которые будут добавлены.

UPDATE – изменяет существующие записи в таблице на основе заданных условий. UPDATE позволяет обновлять значения в одном или нескольких столбцах выбранных записей.

DELETE – удаляет записи из таблицы, которые соответствуют указанным условиям.

SELECT – извлекает данные из таблицы, что позволяет получить нужные записи и отобразить их в удобном для пользователя виде.

Для работы с командами DML на сайте будет создан интерфейс, позволяющий пользователю выполнять вставку, обновление, удаление и выборку данных из базы данных через интуитивный графический интерфейс. Пользователь вводит параметры и условия, после чего команды DML обрабатываются и отправляются на сервер.

Основные элементы внедрения DML–команд: использование выпадающих списков; поля ввода для значений и условий. Например, для команды INSERT пользователь выбирает таблицу, вводит значения и имена столбцов для вставки. Для UPDATE и DELETE задаются условия, чтобы применить изменения только к выбранным записям.

Например, для INSERT PHP–скрипт может получать от клиента название таблицы, список столбцов и значения, а затем формирует SQL–запрос для выполнения вставки.

**Принципы работы с командой INNER JOIN и её внедрение на сайт.**

Команда INNER JOIN в SQL используется для объединения данных из двух таблиц на основе заданного условия соединения, которое должно быть выполнено для каждой пары строк. Результатом выполнения INNER JOIN является таблица, содержащая только те строки, у которых найдено соответствие в обеих таблицах.

В SQL–синтаксисе INNER JOIN применяется следующим образом:

SELECT column\_list

FROM table1

INNER JOIN table2

ON table1.common\_field = table2.common\_field;

Здесь table1 и table2 – это соединяемые таблицы, а common\_field – это поле, по которому происходит соединение строк из обеих таблиц. В результат включаются только строки, у которых значения в common\_field совпадают.

Для того чтобы позволить пользователю выполнять INNER JOIN на веб–странице, создается раздел интерфейса, в котором пользователь выбирает две таблицы, вводит условие соединения и нажимает кнопку для выполнения запроса. Эта информация затем передается серверному скрипту для выполнения запроса и возвращения результатов на сайт. В случае успешного выполнения запроса данные выводятся в виде таблицы, представляя объединенные строки из обеих выбранных таблиц.

**Разработка дизайна веб–страницы.**

1. Блок, в котором перечислен список названий существующих таблиц в базе данных.
2. Блок команд DDL – для создания и удаления таблиц. Включает функционал для выполнения команд CREATE и DROP TABLE, позволяя пользователям управлять структурами базы данных.
3. Блок команд DML – для операций вставки, обновления и удаления данных. В этом блоке реализуются команды INSERT, UPDATE, DELETE для редактирования содержимого таблиц.
4. Блок INNER JOIN – для выполнения запросов INNER JOIN, объединяющих данные из нескольких таблиц на основе указанного условия.
5. Блок отображения результатов – секция, где отображаются результаты выполненных SQL–запросов, включая вывод данных и сообщения об успехе или ошибке выполнения команды.

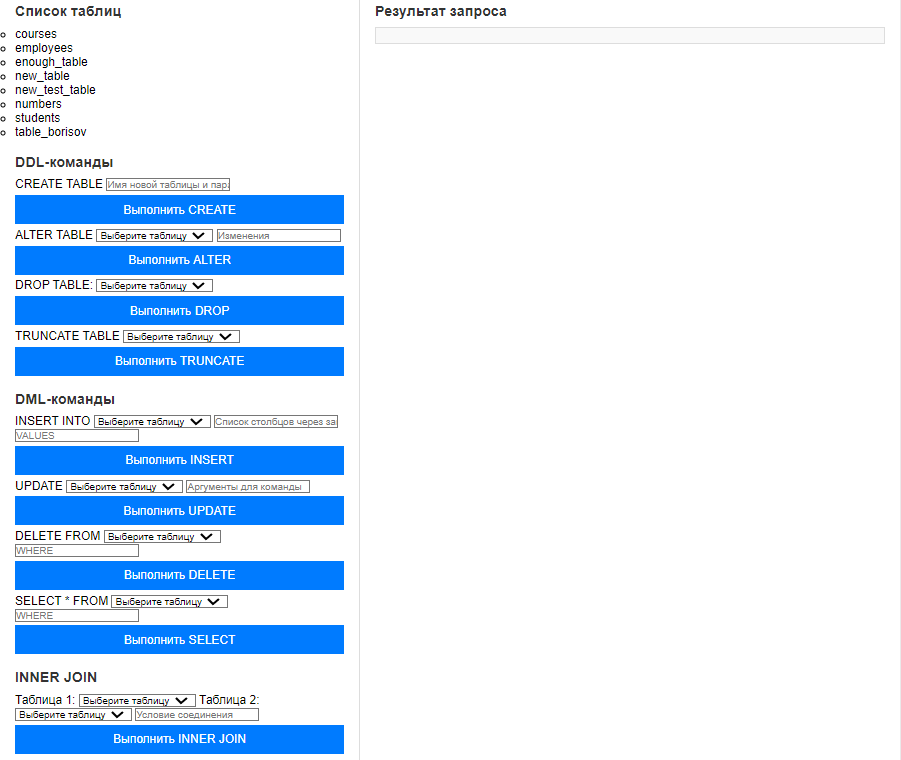
Расположение элементов на странице сделано так, чтобы пользователи могли последовательно переходить от одной команды к другой. Структура страницы состоит из двух панелей:

* Левая панель – содержит все основные блоки команд, организованные в виде вертикального списка. Каждый блок включает элементы управления, такие как выпадающие списки, текстовые поля для ввода значений и условия, а также кнопки для выполнения соответствующих SQL–запросов.
* Правая панель – выделена для отображения результатов запросов. Здесь результаты выводятся в таблице или виде сообщений об успехе или ошибке.

HTML–разметка страницы была спроектирована с использованием семантической структуры, которая упрощает навигацию по сайту. Каждый блок команды был оформлен в виде отдельного <section>, содержащего необходимые элементы управления:

* Для выполнения команд DDL и DML используются элементы <select>, <input>, и <button>.
* Команды INNER JOIN включают два выпадающих списка для выбора таблиц и поле ввода для указания условия соединения.

Для повышения удобства использования и визуального восприятия страницы были разработаны стили CSS, которые делают страницу четкой, структурированной и приятной на вид.



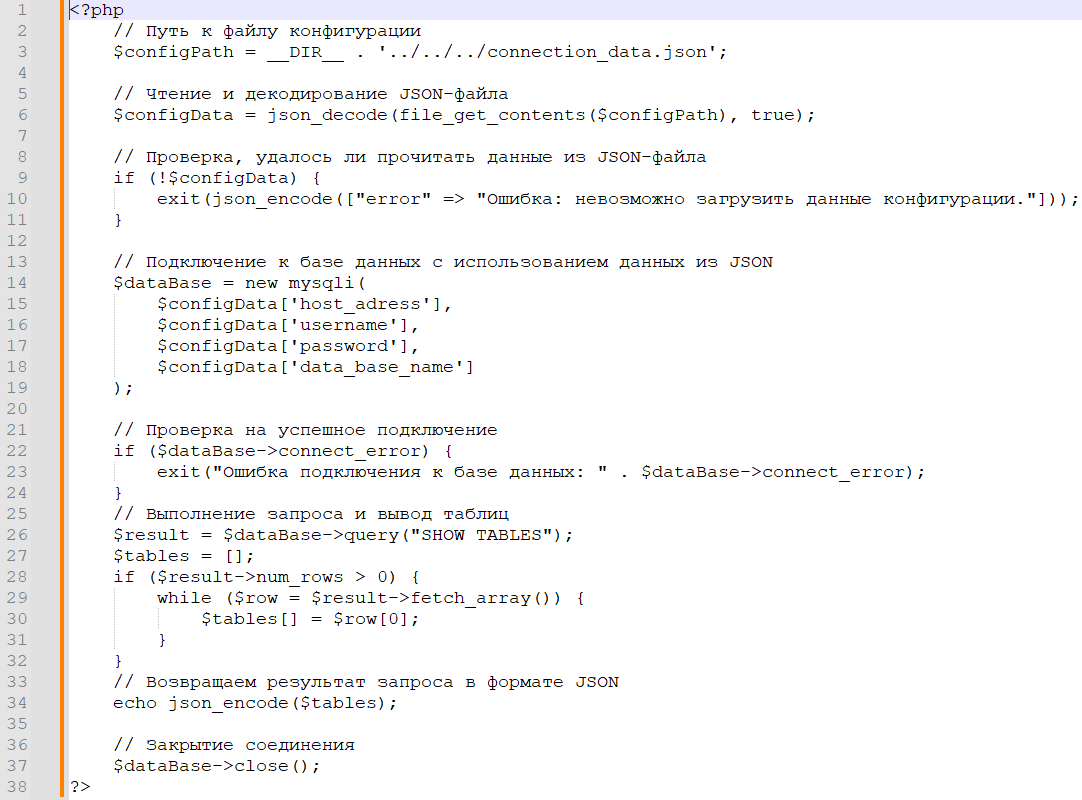
**Блок со списком существующих таблиц в базе данных.**

JavaScript–код реализует запрос к серверу для получения списка таблиц из базы данных. Скрипт fetchData выполняет асинхронный запрос к PHP–скрипту на сервере и обрабатывает ответ. Рассмотрим логику работы более подробно.



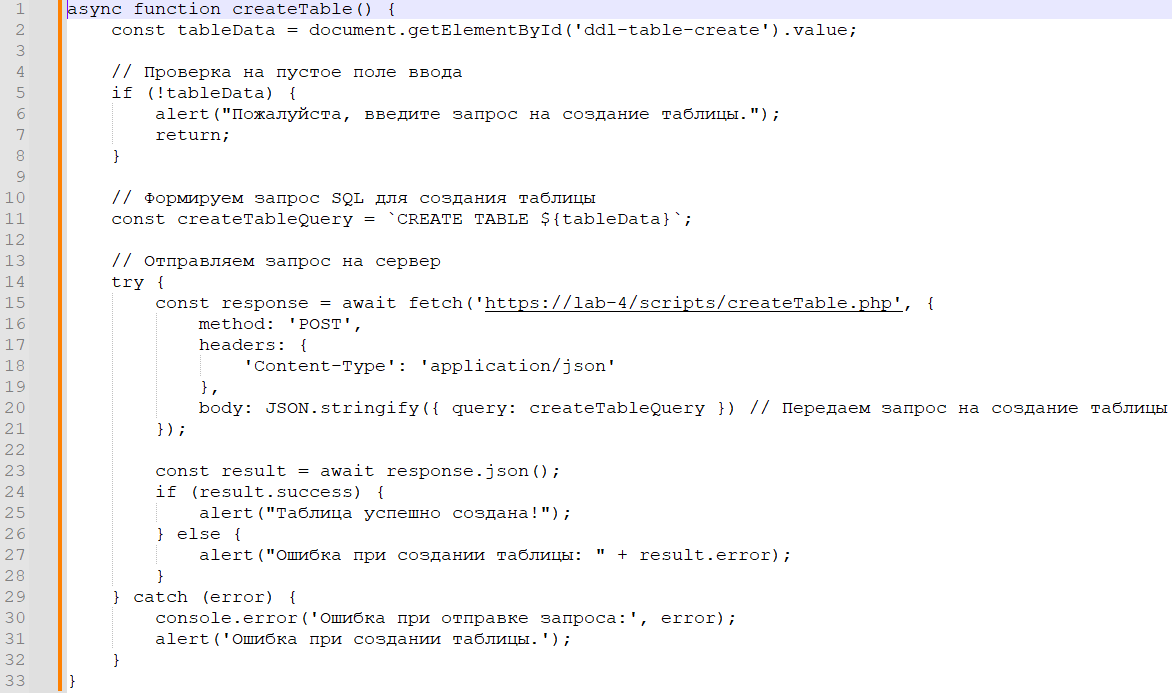
1. Асинхронный запрос. fetchData отправляет запрос на сервер по URL https://lab–4/scripts/table\_names/get\_data.php. Используется метод fetch, который выполняет асинхронные HTTP–запросы.
2. Обработка ответа. После успешного выполнения запроса response.json() преобразует ответ от сервера в формат JSON.
3. Отображение данных. Данные, возвращенные в JSON–формате, используются для формирования HTML–списка таблиц, который затем вставляется в элемент с ID table–names.
4. Обработка ошибок. При возникновении ошибки запрос выводит сообщение об ошибке в консоль, что помогает отладке и диагностике ошибок.

PHP–скрипт get\_data.php выполняет SQL–запрос к базе данных для получения списка всех таблиц, а затем возвращает результат в JSON–формате. Ниже описан процесс обработки запроса в скрипте get\_data.php.

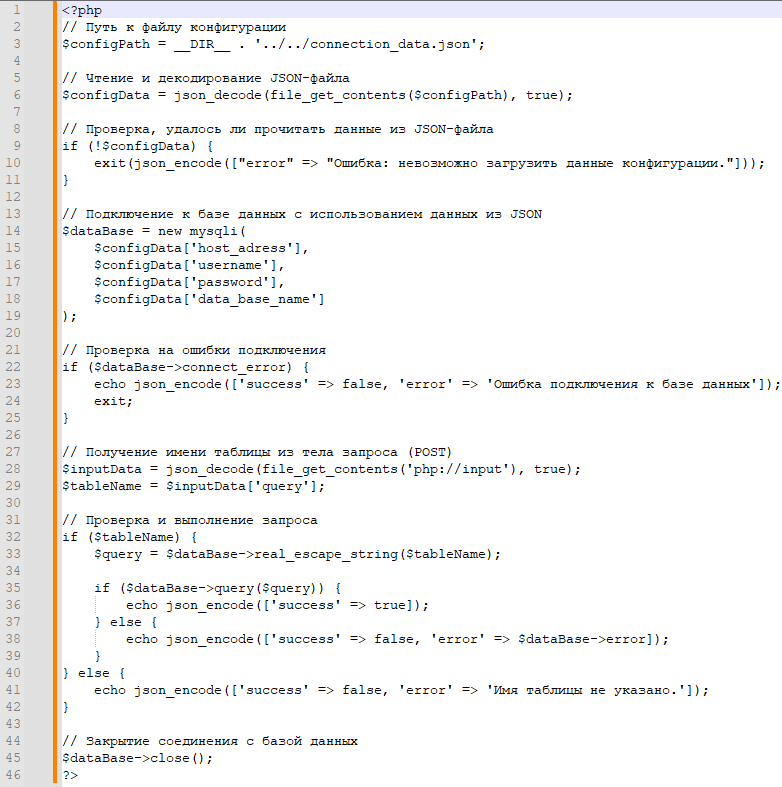


1. Чтение конфигурации. Скрипт загружает файл конфигурации connection\_data.json, который содержит параметры подключения к базе данных (хост, имя пользователя, пароль и название базы данных).
2. Подключение к базе данных. Используя данные конфигурации, скрипт создает подключение к базе данных MySQL с помощью класса mysqli.
3. Проверка подключения. Если подключение не удалось, возвращается ошибка, и выполнение скрипта останавливается.
4. Запрос к базе данных. Скрипт выполняет команду SHOW TABLES, которая возвращает список всех таблиц в базе данных. Результаты сохраняются в массиве $tables.
5. Форматирование и возврат результата. Список таблиц преобразуется в JSON и отправляется в ответ на клиентский запрос.
6. Закрытие соединения. Для освобождения ресурсов подключение к базе данных закрывается.

**Рассмотрение одного из блока команд DLL на примере CREATE.**



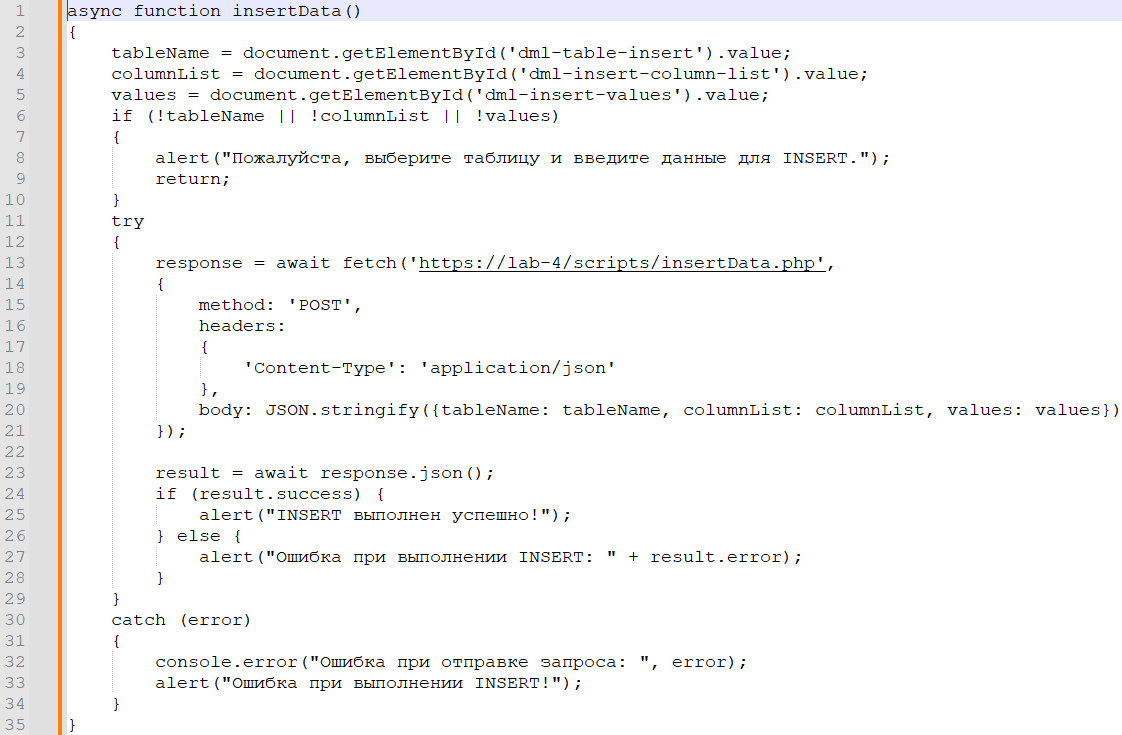
1. Сбор данных. Получаем SQL–запрос для создания таблицы от пользователя из текстового поля ddl–table–create.
2. Проверка заполненности. Если поле ввода пустое, отображается предупреждение, и выполнение функции прекращается.
3. Формирование запроса. Создаем строку SQL–команды для создания таблицы, используя введенные данные.
4. Асинхронный запрос. Отправляем запрос на сервер с помощью fetch, передавая SQL–команду в формате JSON.
5. Обработка ответа. После получения ответа проверяем result.success, чтобы определить, был ли запрос успешным. В случае успеха отображается сообщение об успешном создании таблицы; в противном случае – сообщение об ошибке.



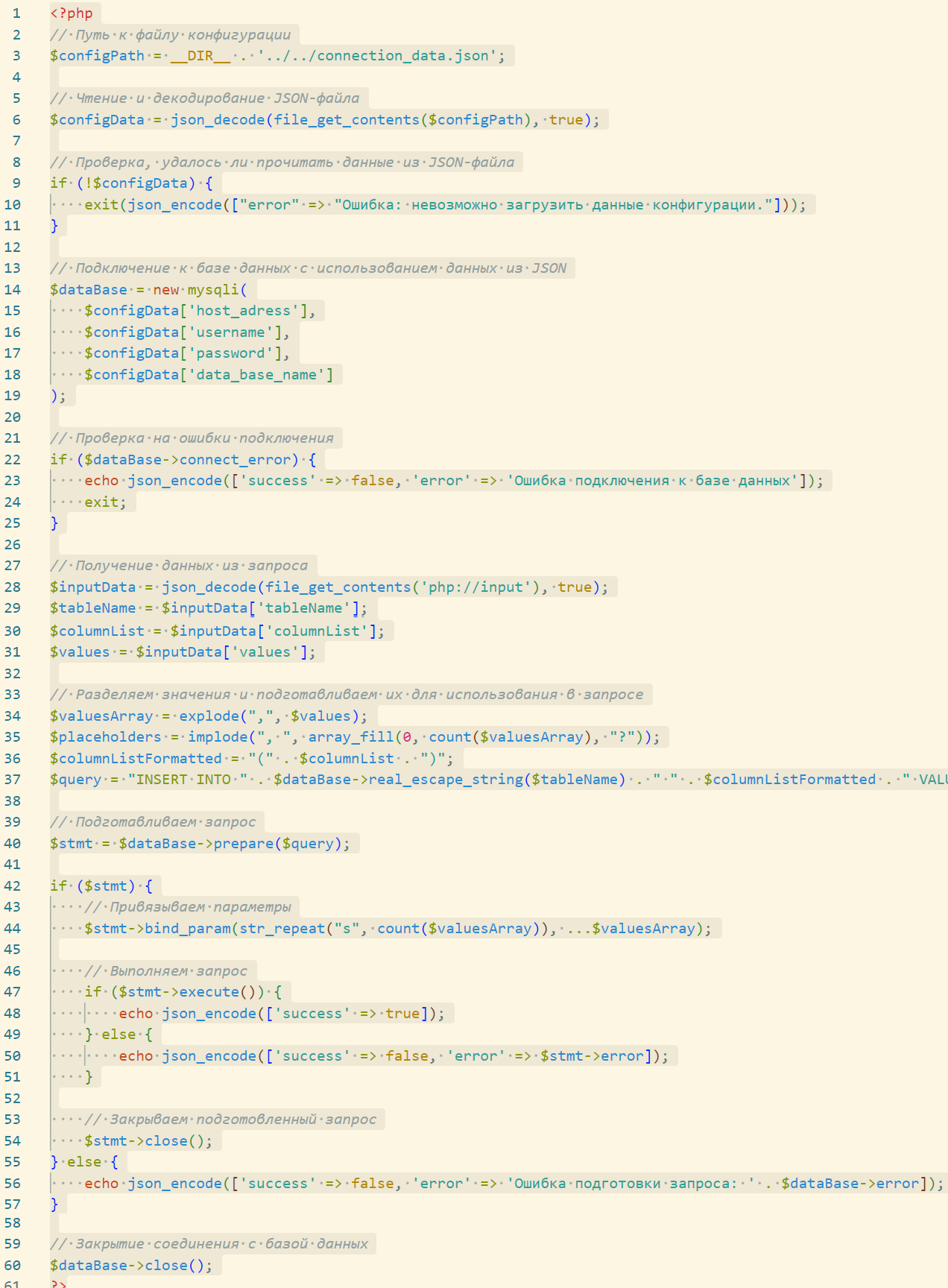
1. Загрузка конфигурации. PHP–скрипт читает файл connection\_data.json, чтобы загрузить параметры для подключения к базе данных.
2. Подключение к базе данных. На основе данных конфигурации скрипт подключается к базе данных MySQL.
3. Получение запроса. Получаем SQL–команду для создания таблицы, отправленную с клиента в JSON–формате.
4. Проверка и выполнение запроса. Если команда присутствует, выполняем её с помощью $dataBase–>query($query). Результат выполнения возвращается клиенту в JSON–формате с флагом success и возможным текстом ошибки.
5. Закрытие соединения. После выполнения запроса соединение с базой данных закрывается для освобождения ресурсов.

Эта структура легко адаптируется для других команд DDL, таких как ALTER TABLE и DROP TABLE.

**Рассмотрение одного из блока команд DML на примере INSERT.**



1. Сбор данных. Получаем данные из полей dml–table–insert, dml–insert–column–list, и dml–insert–values, которые содержат таблицу, список столбцов и значения для вставки.
2. Проверка данных. Если какое–либо поле не заполнено, отображается предупреждение, и функция завершает выполнение.
3. Асинхронный запрос. С помощью функции fetch отправляем POST–запрос на сервер, передавая данные в формате JSON.
4. Обработка ответа. Если сервер сообщает об успешном выполнении команды, отображается сообщение об успешной вставке; если произошла ошибка, выводится её описание.



1. Загрузка конфигурации. Загружаем параметры подключения к базе данных из файла connection\_data.json.
2. Подключение к базе данных. Устанавливаем соединение с базой данных с помощью MySQLi.
3. Получение данных. Получаем параметры запроса из тела POST–запроса, в том числе название таблицы, столбцы и значения для вставки.
4. Формирование и выполнение подготовленного запроса.
5. Возврат результата. Результат выполнения запроса передается обратно клиенту в формате JSON.

**Блок с INNER JOIN.**

HTML–разметка предоставляет пользователю интерфейс для выбора двух таблиц, которые необходимо соединить, и указания условия соединения. Поля select используются для выбора таблиц, а текстовое поле для ввода условия соединения.



JavaScript–функция отправляет запрос на сервер, передавая выбранные таблицы и условие соединения. После получения результата от сервера функция displayJoinResult отображает данные в виде таблицы.



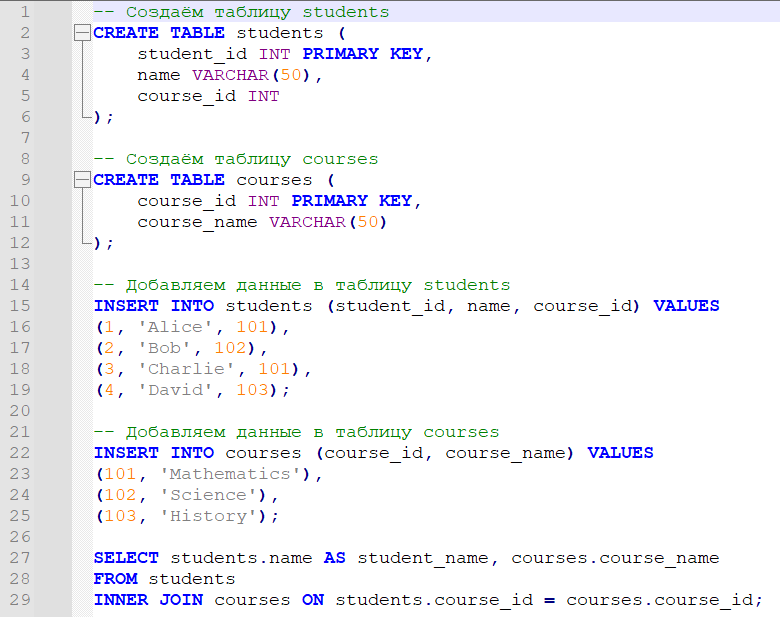
1. Сбор данных. JavaScript собирает значения выбранных таблиц и условия соединения.
2. Асинхронный запрос. Данные отправляются на сервер с помощью fetch, где POST–запрос включает JSON–объект с именами таблиц и условием соединения.
3. Обработка ответа. Результаты, возвращённые сервером, обрабатываются функцией displayJoinResult, которая динамически создаёт HTML–таблицу для отображения данных.



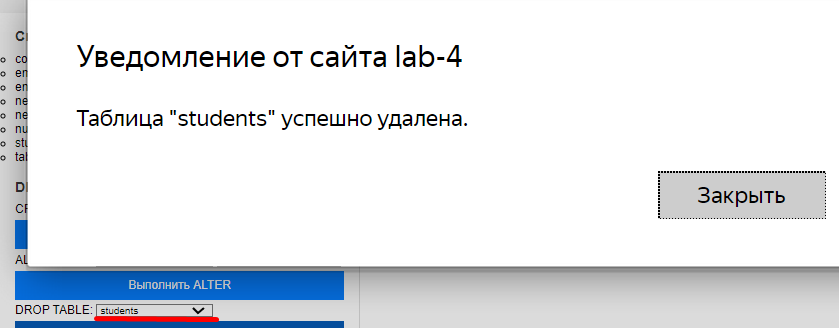
1. Загрузка конфигурации. Конфигурация для базы данных загружается из файла connection\_data.json.
2. Подключение к базе данных. Устанавливаем соединение с базой данных и обрабатываем ошибки.
3. Формирование и выполнение запроса.
4. Получаем таблицы и условие соединения из запроса, формируем SQL–команду INNER JOIN.
5. Выполняем SQL–запрос с соединением таблиц по заданному условию.
6. Форматирование и отправка ответа. Если результат запроса содержит строки, данные отправляются клиенту в JSON–формате. В случае отсутствия данных или ошибки запроса возвращается сообщение об ошибке.

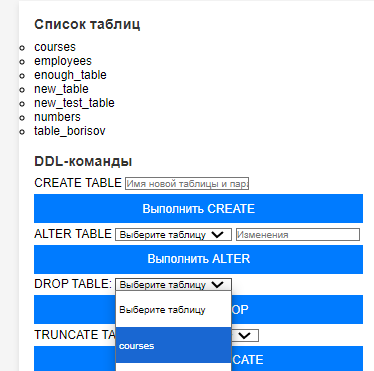
**ТЕСТИРОВАНИЕ ВОЗМОЖНОСТЕЙ ВЕБ–СТРАНИЦЫ**

Попробуем выполнить следующие действия, записанные на языке SQL, но в рамках использования сайта.



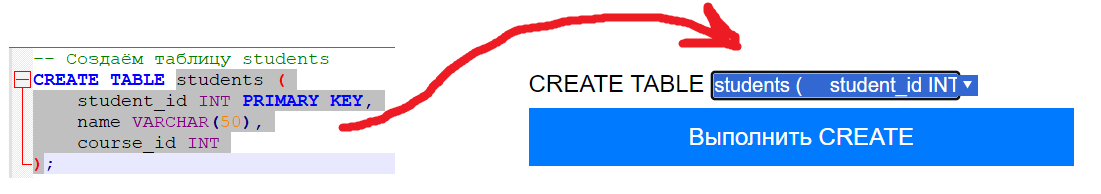
Для начала убедимся, что в базе данных нет таблиц с похожими именами. И если есть, их следует удалить с помощью DROP.



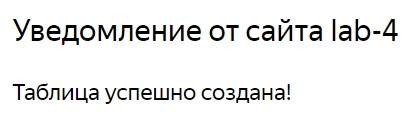




Для создания новой таблицы воспользуемся интерфейсом для команды CREATE TABLE. Конечный запрос формируется, начиная с ключевых слов названия блока, а затем идут аргументы, которые указать пользователь.

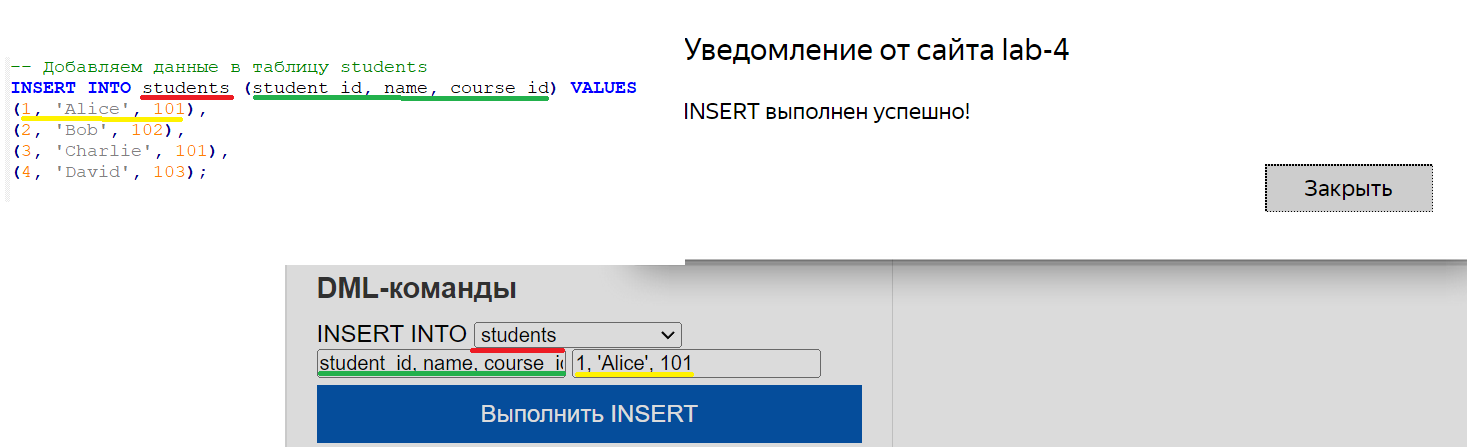


Точно также поступаем и со второй таблицей.



На этом этапе в базе данных вновь появились таблицы students и courses, которые связаны по внешнему ключу, через столбец course\_id.

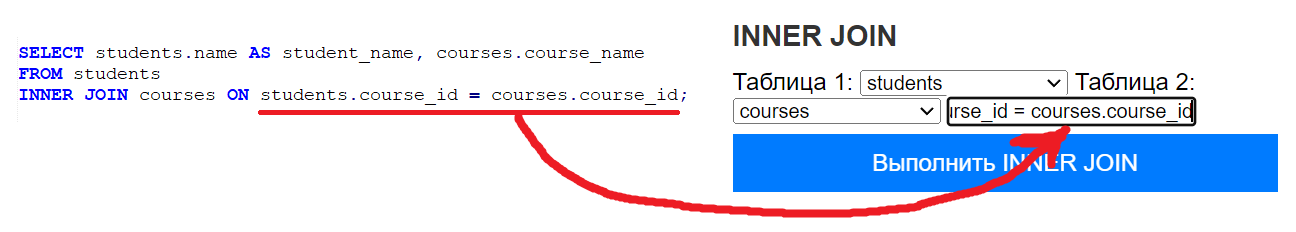
Для заполнения таблиц воспользуемся интерфейсом для INSERT INTO. Добавление каждого значения происходит за одно действие.



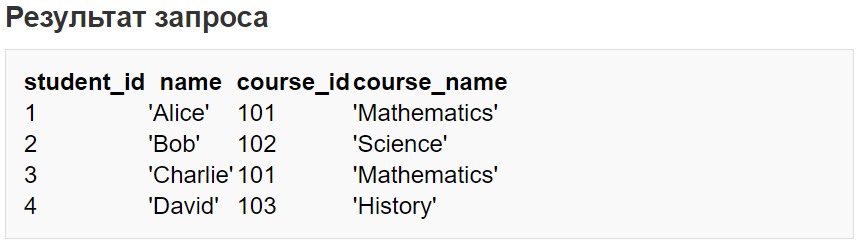
Остаётся лишь убедиться, что таблицы в базе данных заполнились корректно. Для этого нужно воспользоваться блоком SELECT \* FROM, выбрав таблицу из выпадающего списка названий. Поле WHERE используется для выборки, поэтому его стоит оставить пустым, чтобы получить все значения из таблиц.



Чтобы совершить SELECT с INNER JOIN, нужно выбрать две таблицы и указать условие для соединения.



Разница между получившимися таблицами заключается в неполноте настройки запроса, который можно осуществить на сайте. У SQL–запроса из примера будет всего лишь два столбца student\_name и course\_name.



**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной работы была создана веб-приложение с функциями управления базой данных, поддерживающее выполнение команд DDL, DML и INNER JOIN в удобном интерфейсе. Проект позволяет пользователям выполнять основные операции с базой данных, такие как создание и изменение таблиц, вставка, обновление, удаление данных и соединение таблиц, непосредственно из браузера.

В проекте были использованы асинхронные запросы и динамическое обновление интерфейса, что позволило сделать его более отзывчивым и интерактивным. Дополнительное внимание было уделено обработке ошибок, чтобы пользователи получали понятные сообщения о проблемах, возникающих в процессе работы.

**ВЫВОД**

Данное приложение служит хорошим примером системы взаимодействия клиента и сервера для работы с базой данных в реальном времени. Применение асинхронного взаимодействия и организация кода с помощью PHP и JavaScript обеспечивают простоту и удобство использования, что особенно полезно для управления базой данных в обучающих и практических целях.