Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Ижорский колледж»

КУРСОВАЯ РАБОТА

ТЕМА: « Платформа для онлайн-курсов – для обучения и сертификации »

по ПМ.11 МДК.11.01 Технология разработки и защиты баз данных

Выполнил

обучающийся группы 231с

(Шолохов Владимир Вячеславович)

Проверил

преподаватель спец. дисциплин

специальности 09.02.07

Информационные системы и программирование

Венедиктов Д.В.

Санкт-Петербург, 2025

ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

1. Изучить теоретические основы технологии разработки баз данных, включая этапы проектирования, нормализации, реализации и тестирования.

2. Рассмотреть методы и средства защиты баз данных от несанкционированного доступа, атак и вредоносных программ.

3. Провести анализ существующих систем управления базами данных (СУБД) с точки зрения их функциональности, безопасности и удобства использования.

4. Разработать проект базы данных для конкретной предметной области, учитывая требования к безопасности и конфиденциальности данных.

5. Реализовать разработанный проект в выбранной СУБД.

6. Протестировать разработанную базу данных на предмет соответствия требованиям безопасности и надёжности.

7. Оценить эффективность разработанной системы защиты базы данных и предложить рекомендации по её улучшению.

8. Оформить результаты работы в виде курсовой работы, включающей введение, основную часть, заключение, список использованных источников и приложения.

В основной части курсовой работы необходимо подробно рассмотреть следующие вопросы:

* основные этапы разработки базы данных;
* методы обеспечения целостности и безопасности данных;
* механизмы аутентификации и авторизации пользователей;
* шифрование данных и защита от вредоносного ПО;
* особенности реализации проекта базы данных в выбранной СУБД;
* результаты тестирования и оценка эффективности системы защиты.

Для выполнения задания рекомендуется использовать современные научные источники, а также практический опыт разработки и защиты баз данных.

ПЛАН-ГРАФИК ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 10.09 | Выбор темы курсовой работы |  |
| 17.09 | Знакомство с методическими рекомендациями по выполнению курсовой работы |  |
| 24.09 | Знакомство с планом-графиком выполнения курсовой работы |  |
| 08.10 | Работа с нормативно-правовыми документами, учебной литературой |  |
| 22.10 | Анализ предметной области. Постановка задачи |  |
| 29.10 | Составление введения к курсовой работе |  |
| 05.11 | Описание предметной области и функции решаемых задач |  |
| 12.11 | Выбор средств для выполнения курсовой работы. Выбор среды разработки |  |
| 19.11 | Концептуально-логическое проектирование. Составление ER-диаграммы |  |
| 10.12 | Создание и заполнение базы данных |  |
| 24.12 | Представления в базе данных |  |
| 21.01 | Процедуры в базе данных |  |
| 04.02 | Создание ролей в базе данных |  |
| 18.02 | Аутентификация пользователей в базе данных |  |
| 25.02 | Импорт и экспорт базы данных |  |
| 04.03 | Разработка стратегии резервного копирования базы данных |  |
| 11.03 | Тестирование базы данных |  |
| 18.03 | Оптимизация базы данных |  |
| 25.03 | Составление заключения к курсовой работе |  |
| 01.04 | Составление списка источников информации, используемых при выполнении курсовой работы |  |
| 08.04 | Подготовка курсовой работы к защите |  |

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 7

ГЛАВА 1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ БАЗЫ ДАННЫХ 8

1.1. Анализ предметной области 8

1.2. Анализ технологий для разработки базы данных 8

1.3. Постановка задачи курсовой работы 10

1.3.1. Цели курсовой работы 10

1.3.2. Задачи курсовой работы 11

1.4.1. Функциональные требования к системе 11

1.4.2. Нефункциональные требования к системе 11

1.5. Выбор программных средств для разработки базы данных 11

1.6. Выбор среды для разработки базы данных 14

ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 17

2.1. Концептуально-логическое моделирование 17

2.1.1 Концептуальное моделирование 17

2.1.2. Логическое моделирование 18

2.2. Описание информационных объектов базы данных 20

2.3.1. Создание базы данных 25

2.3.2. Заполнение базы данных 25

2.4. Представления в базе данных 27

2.5. Процедуры в базе данных 30

2.6. Создание ролей в базе данных 30

2.7. Аутентификация пользователей в базе данных 30

2.8. Импорт и экспорт базы данных 30

2.9. Разработка стратегии резервного копирования базы данных 30

2.10.1. Тестирование базы данных 31

2.10.2. Оптимизация базы данных 31

ЗАКЛЮЧЕНИЕ 32

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33

ВВЕДЕНИЕ

В современном цифровом мире данные представляют собой один из ключевых ресурсов, влияющих на стратегические решения и операционные процессы организаций. Базы данных стали основным инструментом для хранения, обработки и управления информацией, обеспечивая эффективное взаимодействие с данными. Разработка баз данных – это сложный процесс, требующий комплексного подхода, включающего анализ требований, проектирование, реализацию и поддержку.

С учетом растущего объема информации и усложнения бизнес-процессов, значимость качественного проектирования баз данных трудно переоценить. Правильная структура базы данных обеспечивает не только ее функциональность и производительность, но и легкость в управлении данными, что в свою очередь влияет на оперативность принятия решений.

Однако с увеличением объема данных и многообразием технологий, связанных с их хранением, возникает необходимость уделять особое внимание защите данных. Утечки информации и кибератаки могут привести к значительным финансовым потерям и подрыву доверия со стороны клиентов. Поэтому технологии защиты баз данных становятся критически важными для бизнеса всех размеров.

Эта работа сосредоточится на ключевых аспектах разработки и защиты баз данных, включая методы проектирования и элементы безопасности, которые должны быть внедрены для обеспечения надежности и защиты данных в любой организации. Изучение этих тем поможет определить лучшие практики, которые должны применяться для достижения устойчивости и эффективности работы с информацией в условиях динамично меняющейся технологической среды.

ГЛАВА 1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ БАЗЫ ДАННЫХ

1.1. Анализ предметной области

Технология разработки и защиты баз данных охватывает широкий спектр процессов, методов и инструментов, необходимых для создания, управления и обеспечения безопасности баз данных. Это область, которая активно развивается в ответ на растущие требования к хранению и защите информации, что делает её актуальной для различных отраслей, таких как финансы, здравоохранение, образование и многие другие.

1.2. Анализ технологий для разработки базы данных

Разработка баз данных включает в себя использование различных технологий и инструментов, которые помогают осуществлять проектирование, реализацию, управление и оптимизацию баз данных. Рассмотрим основные технологии и подходы, которые широко применяются в этой сфере.

**Системы управления базами данных (СУБД)**

СУБД являются основными инструментами для создания и управления базами данных. Они могут быть разделены на несколько категорий:

* реляционные СУБД (RDBMS): MySQL, PostgreSQL, Oracle, Microsoft SQL Server;
* нереляционные СУБД (NoSQL): MongoDB, Cassandra, Redis;
* гибридные СУБД: Microsoft Azure Cosmos DB.

**Языки программирования**

* SQL (Structured Query Language): основной язык для работы с реляционными базами данных. Позволяет создавать, изменять, запрашивать и управлять данными;
* Python и JavaScript: часто используются для взаимодействия с базами данных в веб-приложениях. Библиотеки, такие как SQLAlchemy или Sequelize, упрощают работу с базами данных;
* Java и C#: используются в крупных корпоративных приложениях, особенно с реляционными базами данных.

**Инструменты для проектирования баз данных**

* ER-моделирование: Инструменты, такие как Lucidchart, Draw.io или ER/Studio, позволяют создавать диаграммы сущностей и взаимосвязей (ER-диаграммы), которые помогают визуализировать структуру базы данных;
* системы управления версиями: Git и другие системы позволяют контролировать изменения в схемах баз данных и процессах миграции.

**Инструменты для обеспечения безопасности баз данных**

* шифрование: использование технологий шифрования данных (например, AES) для защиты информации как «в покое», так и «в движении»;
* аутентификация и авторизация: применение OAuth, OpenID Connect и других протоколов для управления доступом к базе данных;
* мониторинг и аудит: инструменты, такие как Splunk, могут использоваться для отслеживания запросов и анализа событий безопасности.

**Методы оптимизации производительности**

* индексация: создание индексов для ускорения выполнения запросов, особенно в реляционных базах данных.
* кэширование: использование кэшей, таких как Redis или Memcached, для ускорения доступа к часто запрашиваемым данным.
* балансировка нагрузки: распределение запросов между несколькими серверами баз данных для повышения производительности и отказоустойчивости.

Выбор технологий для разработки базы данных зависит от специфики проекта, объема данных, ожидаемых нагрузок и требований к безопасности. Комплексный подход к выбору инструментов и методов способствует созданию эффективных и надежных систем управления данными, которые могут адаптироваться к изменяющимся условиям и требованиям.

1.3. Постановка задачи курсовой работы

С ростом объемов данных, а также увеличением угроз безопасности информации, эффективная разработка и защита баз данных становятся крайне важными задачами для организаций. Безопасность данных имеет критическое значение для решения вопросов конфиденциальности, целостности и доступности информации.

Успешное выполнение работы будет способствовать более глубокому пониманию актуальных проблем и новых подходов в области безопасного управления данными.

1.3.1. Цели курсовой работы

Основная цель заключается в создании эффективной и безопасной базы данных, способной обеспечить надежный доступ к информации и защиту от потери или несанкционированного доступа.

1.3.2. Задачи курсовой работы

* проектирование архитектуры баз данных;
* разработка схемы данных и моделей;
* обеспечение безопасности данных через шифрование, аутентификацию и авторизацию;
* реализация механизмов резервного копирования и восстановления;
* мониторинг и аудит доступа к данным.

1.4.1. Функциональные требования к системе

* возможность создания, обновления и удаления данных;
* поддержка сложных запросов и отчетов;
* обеспечение многопользовательского доступа.

1.4.2. Нефункциональные требования к системе

* высокая производительность и надежность;
* защита от несанкционированного доступа;
* удобный интерфейс для пользователей.

1.5. Выбор программных средств для разработки базы данных

Выбор программных средств для разработки баз данных является ключевым этапом, который влияет на функциональность, производительность и безопасность системы.

При выполнении данной курсовой работы я планирую использовать следующие программные средства:

* система управления базами данных – MySQL;
* языки программирования – SQL, Java;
* средства проектирования базы данных – dbdiagram.io, Liquibase ;
* инструменты для обеспечения безопасности данных – Vormetric, Splunk;
* средства тестирования и оптимизации приложений – JMeter, EXPLAIN .

Выбор программных средств для разработки базы данных зависит от специфических требований проекта, таких как объем данных, тип доступа, требования к безопасности и производительности. Важно учитывать будущую масштабируемость и поддерживаемость решений, чтобы создать надежную и эффективную систему.

1.6. Выбор среды для разработки базы данных

Выбор подходящей среды для разработки базы данных является критически важным этапом проекта, который может оказать значительное влияние на эффективность работы команды и качество конечного продукта.

При выполнении данной курсовой работы я планирую использовать следующие среды и инструменты:

* IDE и редактор кода – MySQL Workbench;
* система контейнеризации – Docker;
* система управления версиями – git;
* фреймворк и библиотека – Hibernate;

Выбор среды для разработки базы данных требует комплексного подхода с учетом специфики проекта, требований к производительности, безопасности и интеграции. Правильный выбор инструментария и технологии существенно влияет на успех проекта, поэтому следует тщательно проанализировать свои потребности и доступные решения.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

2.1. Концептуально-логическое моделирование

Концептуально-логическое моделирование – это важный этап в проектировании базы данных, который позволяет структурировать данные и определить их взаимосвязи. Этот процесс разделяется на два ключевых этапа: концептуальное моделирование и логическое моделирование.

2.1.1 Концептуальное моделирование

Целью концептуального моделирования является определение основных сущностей и их связей в предметной области, не вдаваясь в детали реализации.

**Идентификация сущностей**

Пользователь, Курс, Диплом, Отзывы

**Определение атрибутов**

Пользователь

- ID

- ФИО

- Email

- Телефон

- Пароль

Курс

- ID

- Название

- Описание

- Дата начала

- Дата окончания

- Преподаватели

Диплом

- ID

- Номер диплома

- Дата выдачи

- Пользователь

Отзывы

- ID

- Пользователь

- Курс

- Оценка

- Текст отзыва

- Дата создания

**Установление связей**

Пользователь и Курс

- Один-ко-многим: Один пользователь (студент) может проходить несколько курсов, а курс может быть доступен для множества пользователей.

Пользователь и Диплом

- Один-ко-многим: Один пользователь может получить несколько дипломов (по завершению разных курсов), а каждый диплом выдается только одному пользователю.

Курс и Отзывы

- Один-ко-многим: Один курс может иметь несколько отзывов, каждый отзыв связан только с одним курсом.

Пользователь и Отзывы

- Один-ко-многим: Один пользователь может оставить несколько отзывов, но каждый отзыв написан только одним пользователем.

**Создание ER-диаграммы**

В данной диаграмме прямоугольные блоки отвечают за сущности, овальные – атрибуты, ромбы – связи.

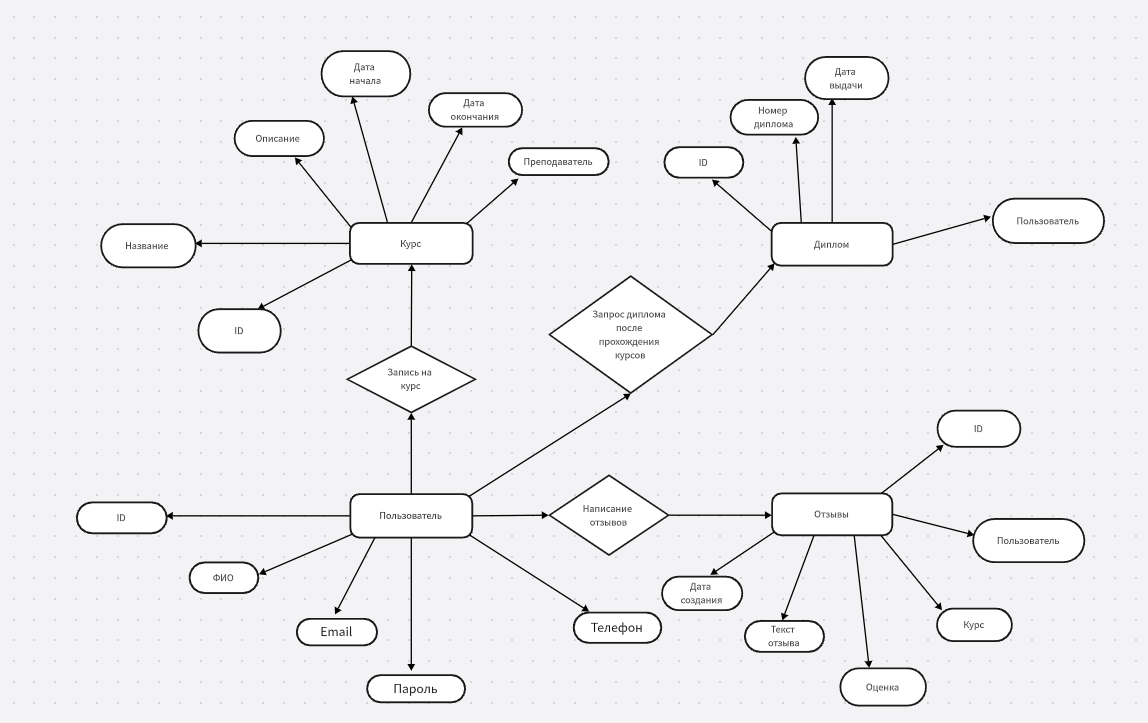


Рисунок 1 – ER-диаграмма разрабатываемой базы данных

2.1.2. Логическое моделирование

Целью логического моделирования является перевод концептуальной модели в логическую, с уточнением структуры базы данных и обязательств для хранения данных.

**Определение первичных ключей**

Пользователь: ID

Курс: ID

Диплом: ID

Отзывы: ID

**Формализация связей**

⦁ Для связи между пользователем и курсом требуется: ID\_Пользователя.

⦁ Для связи между пользователем и дипломом требуется: ID\_Пользователя, ID\_Курса.

⦁ Для связи между пользователем и отзывом требуется: ID\_Пользователя, ID\_Курса.

⦁ Для связи между курсом и дипломом требуется: ID\_Курса.

⦁ Для связи между курсом и отзывом требуется: ID\_Курса, ID\_Пользователя.

**Определение типов данных**

- ID: INTEGER

- ФИО: VARCHAR

- Email: VARCHAR

- Телефон: VARCHAR

- Пароль: VARCHAR

- Название курса: VARCHAR

- Описание курса: VARCHAR

- Дата начала: DATETIME

- Дата окончания: DATETIME

- Преподаватели: VARCHAR

- Номер диплома: VARCHAR

- Дата выдачи: DATETIME

- Пользователь: VARCHAR

- Курс: VARCHAR

- Оценка: VARCHAR

- Текст отзыва: VARCHAR

- Дата создания: DATETIME

**Создание логической схемы базы данных**



Концептуально-логическое моделирование является важным этапом разработки базы данных, который позволяет углубиться в детализацию структуры данных и обеспечить целостность и согласованность информации. Правильное моделирование позволяет избежать проблем на следующих этапах разработки и эксплуатации базы данных.

2.2. Описание информационных объектов базы данных

Информационные объекты базы данных представляют собой основные строительные блоки, на которых базируется структура любой базы данных. Они включают в себя различные сущности и их атрибуты, которые позволяют организовать, хранить и управлять данными.

Таблица 1 – информационные объекты разрабатываемой базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Объект | Атрибуты | Связи | Ограничения | Тип данных |
| Пользователь(users) | ID, ФИО, Email,  Телефон, Пароль | Один Пользователь может записываться на множество курсов.  Один пользователь  Может получать множество дипломов за пройденные курсы.  Один пользователь может оставлять множество отзывов. | Номер телефона должен быть уникальным,  Пароль должен быть обязательно | INTEGER, VARCHAR |
| Курс (curs) | ID, Название, Описание, Дата Начала, Дата окончания, Преподаватели | Один курс связан со многими пользователями | Текст обязателен | INTEGER, VARCHAR, DATETIME |
| Диплом(diplom) | ID, Номер диплома, Дата выдачи, Пользователь | Один диплом связан только с одним человеком | Текст обязателен | INTEGER, VARCHAR, DATETIME |
| Отзывы(comments) | ID, Пользователь, Курс, Оценка, Текст отзыва, Дата создания | Многие отзывы связаны с одним пользователем и с одной записью | Текст обязателен | INTEGER, VARCHAR, DATETIME |

Описание информационных объектов базы данных позволяет структурировать и организовать информацию, обеспечивая эффективное хранение и доступ к данным. Правильное определение сущностей, их атрибутов и связей является ключом к успешному проектированию базы данных и ее функциональной полноте.

2.3.1. Создание базы данных

Для того чтобы создать базу данных можно использовать консольные SQL-запросы или взаимодействие с интерфейсом MySQL, здесь будет использован вариант с отправлением SQL-запросов. Создаем базу данных

с нестандартным именем.

|  |
| --- |
| sql CREATE DATABASE sait; |

Рисунок 2 – созданная база данных

2.3.2. Заполнение базы данных

Следующим шагом является создание таблиц для хранения данных. Каждая таблица создаётся с учётом структуры, описанной на этапе концептуального моделирования. Для начала будет создана таблица «Пользователи».

|  |
| --- |
| sql  CREATE TABLE users (  id INT PRIMARY KEY,  name VARCHAR(255) NOT NULL,  email VARCHAR(255) NOT NULL ,  phone VARCHAR(20) DEFAULT NULL,  password VARCHAR(255) NOT NULL,  ); |

После успешного ответа от SQL-запроса мы видим таблицу в нашей базе данных.

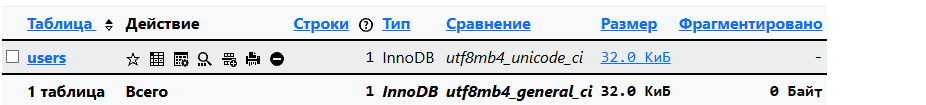


Рисунок 3 – добавление объекта в базу данных

Точно так же создаем еще три таблицы: «Курс», «Диплом», «Отзывы» - которые представляют из себя объекты.

|  |
| --- |
| Sql  CREATE TABLE courses (  ID INT AUTO\_INCREMENT PRIMARY KEY,  Title VARCHAR(255) NOT NULL,  Description TEXT NOT NULL,  Start\_Date DATETIME NOT NULL,  End\_Date DATETIME NOT NULL,  Instructors VARCHAR(255) NOT NULL  ); |

|  |
| --- |
| Sql  CREATE TABLE diplomas (  ID INT AUTO\_INCREMENT PRIMARY KEY,  Diploma\_Number VARCHAR(50) NOT NULL,  Issuance\_Date DATETIME NOT NULL,  User\_id integer  ); |

|  |
| --- |
| Sql  CREATE TABLE reviews (  ID INT AUTO\_INCREMENT PRIMARY KEY,  User\_id integer NOT NULL,  Course VARCHAR(255) NOT NULL,  Grade INT NOT NULL,  Feedback\_Text TEXT NOT NULL,  Creation\_Date DATETIME NOT NULL  ); |

После успешных SQL-запросов мы можем перейти к установке связей между объектами в базе данных

|  |
| --- |
| Sql  ALTER TABLE diplomas  ADD CONSTRAINT user  FOREIGN KEY (user\_id) REFERENCES users(id); |

Выше показан пример как связать пользователей

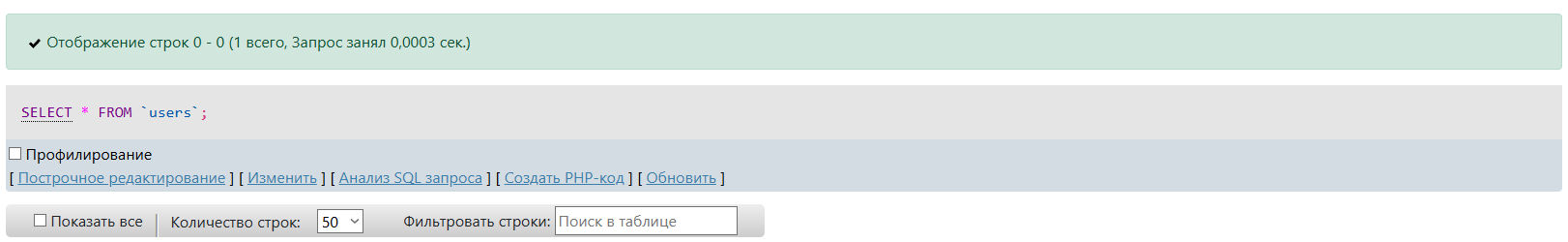


Рисунок 4 – добавление атрибутов объекту в базе данных



Рисунок 5 – установление связей между объектами в базе данных

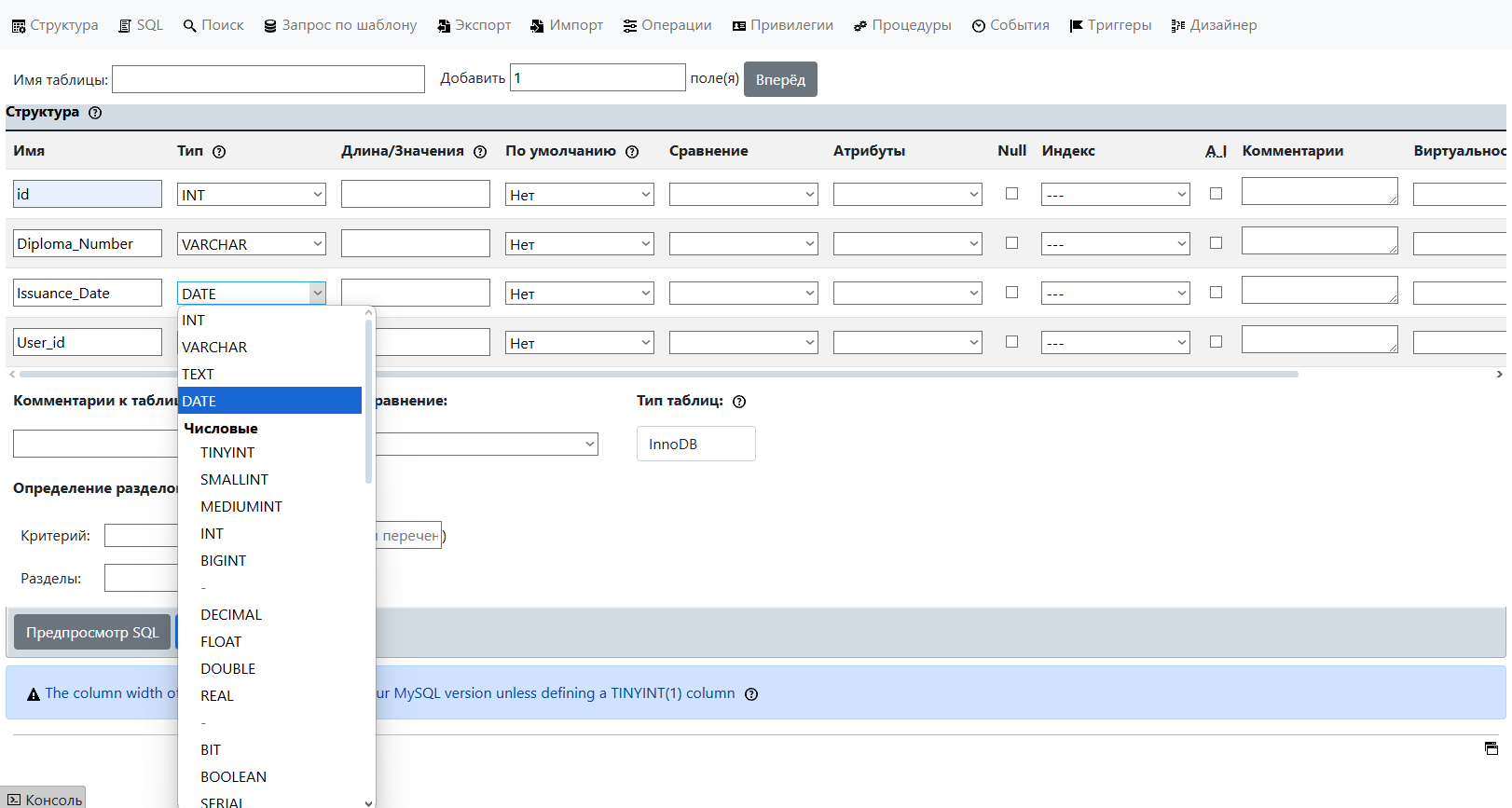


Рисунок 6 – выбор типа данных для объекта в базе данных

2.4. Представления в базе данных

Представление в базе данных – это виртуальная таблица, которая формируется на основе результата выполнения SQL-запроса. Она не содержит данных, а лишь определяет, как данные будут отображаться пользователям.

Функции представлений:

* упрощение сложных запросов – позволяют объединять данные из нескольких таблиц и показывать только нужные поля, что делает работу с данными более удобной;
* безопасность – можно ограничить доступ пользователей к определённым столбцам или строкам баз данных, предоставляя доступ только к представлениям;
* логическая независимость – изменения в базах данных не влияют на внешний интерфейс, так как представления могут скрывать изменения структуры таблиц;
* кодовая реиспользуемость – позволяют сократить код, избавляя от необходимости повторно писать одни и те же запросы.

Представление создается с помощью команды CREATE VIEW. К примеру, вот таким запросом можно вывести пользователей, которые были зарегистрированы после 01.01.2024.

|  |
| --- |
| Sql CREATE VIEW recent\_diplomas AS  SELECT User\_id, Issuance\_Date  FROM diplomas  WHERE Issuance\_Date > '2024-01-01'; |

После создания представления его можно использовать так же, как и таблицу: Данный запрос выведет все полученные дипломы после 01.01.2024, и от пользователя с id= 055011

|  |
| --- |
| Sql SELECT \* FROM diplomas WHERE user\_id = 055011; |

Представления могут быть обновляемыми или не обновляемыми. Обновляемые представления позволяют выполнять операции INSERT, UPDATE и DELETE, при этом изменения будут применяться к базовой таблице. В качестве примеря в базе данных я поменял имя человека с id=055011.

|  |
| --- |
| Sql UPDATE users  SET name = 'Владимир'  WHERE id = 055011; |

Представления можно изменять с помощью команды CREATE OR REPLACE VIEW или удалять с помощью DROP VIEW. С помощью данного запроса мы поменяли представление recent\_diplomas, добавили в него id и Diploma\_Number.

|  |
| --- |
| Sql CREATE VIEW recent\_diplomas  ASSELECT id, Diploma\_Number, User\_id, Issuance\_Date  FROM diplomas  WHERE Issuance\_Date > '2024-01-01'; |

Если представление больше не нужно его можно удалить.

|  |
| --- |
| Sql  DROP VIEW recent\_diplomas; |

Представления в базах данных являются мощным инструментом для работы с данными. Они обеспечивают большую гибкость, безопасность и легкость в использовании, способствуя более эффективной организации и обработке информации. Понимание работы с представлениями – важный аспект для разработчиков и администраторов баз данных.

2.5. Процедуры в базе данных

## Процедуры (хранимые процедуры) представляют собой программные блоки, сохраняемые непосредственно на сервере СУБД, которые инкапсулируют бизнес-логику и позволяют выполнять сложные операции с данными.

В рамках курсовой работы, посвящённой разработке онлайн курсов, процедуры могут быть использованы для реализации ряда функциональных возможностей.Рассмотрим пример процедуры для добавления нового пользователя в таблицу «AddUser»:

|  |
| --- |
| Sql CREATE PROCEDURE AddUser(  IN p\_email VARCHAR(100),  IN p\_password VARCHAR(255)  )  Sql  BEGIN  DECLARE v\_now TIMESTAMP;  SET v\_now = CURRENT\_TIMESTAMP;  INSERT INTO users (email, password, created\_at)  VALUES (p\_email, p\_password, v\_now);  END |

## Аналогичным образом можно создать процедуры для работы с комментариями, сообщениями и другими объектами базы данных. Важным аспектом является обработка ошибок и обеспечение целостности данных, что может быть реализовано через механизмы обработки исключений, поддерживаемые выбранной СУБД.

2.6. Создание ролей в базе данных

Роли в базе данных представляют собой удобный способ централизованного управления доступом, позволяющий группировать права пользователей и назначать их в виде набора привилегий. Это упрощает администрирование, поскольку вместо того, чтобы назначать каждое разрешение отдельно, можно создать роль с необходимыми привилегиями и затем присваивать эту роль нужным пользователям. Такой подход не только повышает безопасность системы, но и обеспечивает удобное масштабирование управления правами при добавлении новых пользователей или изменении требований к доступу.

Например, в MySQL, начиная с версии 8.0, можно создать роль с ограниченными правами, предназначенную только для чтения данных. Для этого выполняются следующие SQL-запросы:

|  |
| --- |
| Sql CREATE ROLE 'read\_only';  GRANT SELECT ON sait.\* TO ' read\_user'; |

После создания роли read\_user её можно назначить конкретным пользователям, чтобы они могли просматривать данные без возможности их изменения:

|  |
| --- |
| Sql GRANT 'read\_only' TO 'имя\_пользователя'; |

Если возникает необходимость изменить набор привилегий роли, это можно сделать с помощью команды REVOKE для удаления существующих разрешений и последующего добавления новых. Таким образом, управление правами становится централизованным и более гибким, что особенно важно в условиях многопользовательской среды и высоких требований к безопасности базы данных.

2.7. Аутентификация пользователей в базе данных

Аутентификация пользователей – это процесс проверки подлинности данных, предоставленных пользователем, с целью подтверждения его личности и предоставления доступа к системе. В рамках данной курсовой работы реализована система аутентификации с использованием PHP и базы данных MySQL. Рассмотрим основные этапы и механизмы реализации:

**Этапы регистрации пользователя**

1. Получение данных из формы (номер и пароль).
2. Проверка совпадения пароля и его подтверждение.
3. Проверка уникальности номера в базе данных с использованием SQL запроса.
4. Хеширование пароля с использованием функции password\_hash() и алгоритма PASSWORD\_DEFAULT.
5. Сохранение данных в базе и создание сессии.

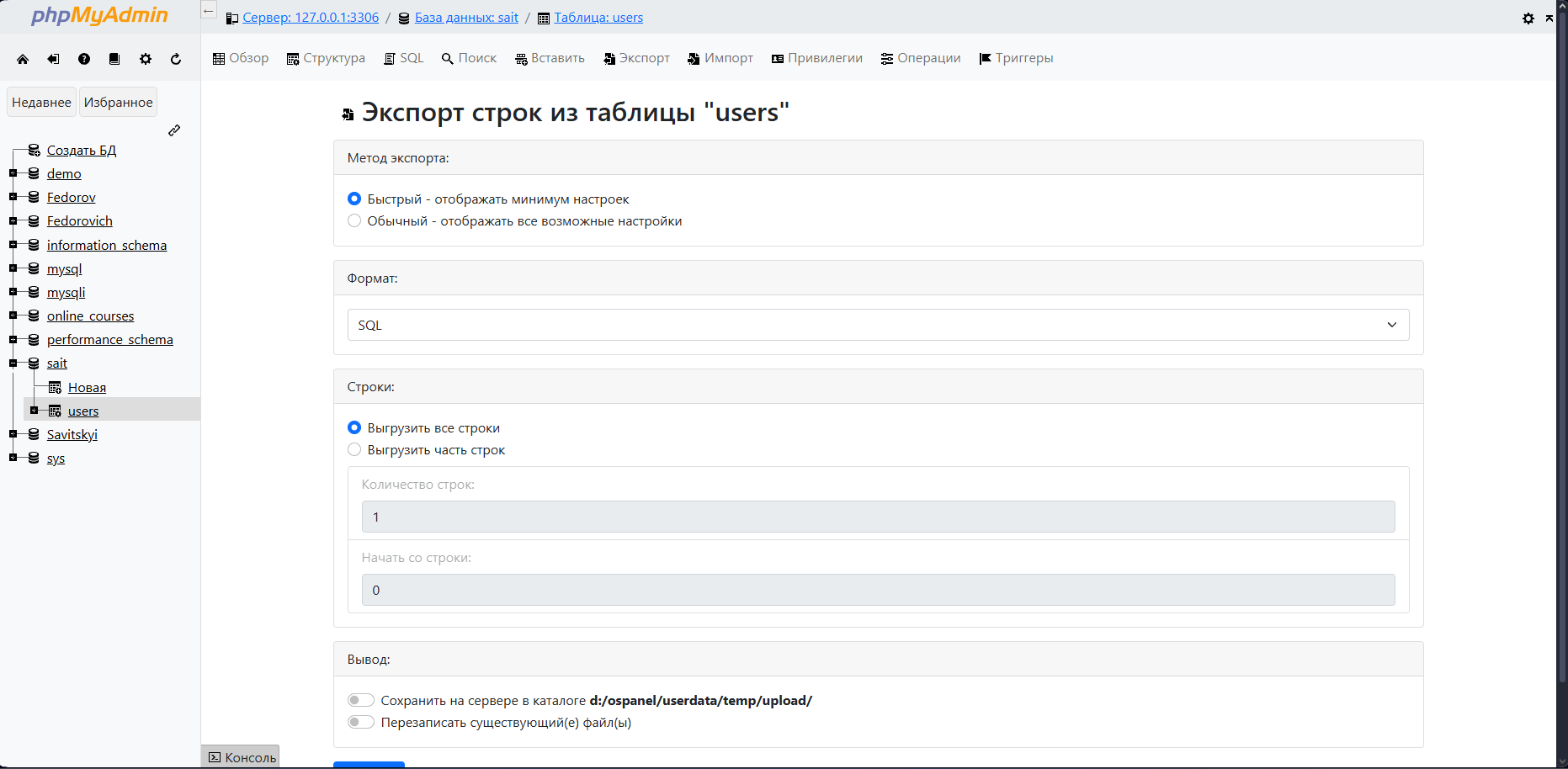
|  |
| --- |
| $hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);  try {  $stmt = $db->prepare("INSERT INTO users (name, email, phone, password, created\_at) VALUES (?, ?, ?, ?, NOW())");  $result = $stmt->execute([$name, $email, $phone, $hashed\_password]);    if ($result) {  echo json\_encode([  'success' => true,  'message' => 'Регистрация прошла успешно! Вы будете перенаправлены на страницу входа.'  ]);  } else {  throw new Exception("Failed to insert user data");  }  } catch (PDOException $e) {  error\_log("Registration error: " . $e->getMessage());  echo json\_encode([  'success' => false,  'message' => 'Ошибка при сохранении данных: ' . $e->getMessage()  ]);  } |

**Этапы авторизации пользователей**

1. Получение данных из формы (номер и пароль).
2. Поиск пользователя по номеру с использованием подготовленного запроса.
3. Сравнение введённого пароля с хранимым хешем с помощью password\_verify().
4. В случае успешного совпадения – создание сессии и перенаправление на главную страницу.
5. В случае ошибки – возврат на страницу авторизации с кодом ошибки.

|  |
| --- |
| if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  $email = trim($\_POST['email'] ?? '');  $password = $\_POST['password'] ?? '';  error\_log("Attempting login for email: " . $email);  if (empty($email) || empty($password)) {  $response['message'] = 'Все поля обязательны для заполнения';  error\_log("Empty email or password");  } else {  try {  // Проверяем существование пользователя  $stmt = $db->prepare("SELECT \* FROM users WHERE email = ?");  $stmt->execute([$email]);  $user = $stmt->fetch();  if ($user) {    if (password\_verify($password, $user['password'])) {  $\_SESSION['user'] = [  'id' => $user['id'],  'name' => $user['name'],  'email' => $user['email']  ];  $response = [  'success' => true,  'message' => 'Авторизация успешна'  ];  } else {  $response['message'] = 'Неверный пароль';  }  } else {  $response['message'] = 'Пользователь не найден';  }  } catch (PDOException $e) {  error\_log("Database error: " . $e->getMessage());  $response['message'] = 'Ошибка базы данных';  }  }  } |

2.8. Импорт и экспорт базы данных

Для экспорта базы данных в phpMyAdmin необходимо перейти на вкладку «Экспорт». В данном разделе можно выбрать тип экспорта (например, быстрый или настроенный), указать дополнительные параметры и подтвердить операцию. После завершения процесса будет сформирован файл с расширением .sql, который можно использовать для дальнейшего восстановления или переноса данных.

Для импорта базы данных требуется файл в формате .sql, содержащий структуру и содержимое таблиц. Чтобы выполнить импорт, необходимо открыть вкладку «Импорт» в phpMyAdmin, выбрать нужный файл и запустить процесс, нажав кнопку подтверждения. Если импорт прошёл успешно, все данные из файла будут загружены в базу данных.

2.9. Разработка стратегии резервного копирования базы данных

Резервное копирование базы данных – это ключевой элемент обеспечения надежности и устойчивости информационной системы социальной сети. При разработке социальной платформы, где осуществляется постоянный обмен данными между пользователями, необходимо обеспечить сохранность информации в случае сбоев, аппаратных или программных ошибок, а также действий злоумышленников.

### **Стратегия резервного копирования**

1. Полное резервное копирование: Выполняется раз в сутки в ночное время, когда нагрузка на систему минимальна. Это позволяет создать полную копию базы данных, включая все таблицы, индексы, связи и данные пользователей.
2. Место хранения резервных копий: локальный сервер в защищенной внутренней сети; облачное хранилище с двухфакторной аутентификацией.
3. Контроль целостности: После каждого копирования автоматически проверяется целостность резервной копии с использованием контрольных сумм (hash).
4. План восстановления: Регламентировано пошаговое восстановление данных из резервной копии, включая развертывание полной копии и последовательное применение данных до нужного состояния.

2.10.1. Тестирование базы данных

Для безопасности и стабильности своей базы данных я попытался учесть ошибки и недостатки системы авторизации. На данной таблице указаны все предусмотренные случаи ошибок для регистрации:

|  |  |
| --- | --- |
| Сценарий | Результат |
| Пользователь вводит неправильный формат почты |  |
| Пользователь ввел разные пароли |  |
| Пользователь вводит недостаточно цифр |  |
| Пользователь ввел слишком короткий пароль |  |

А также для входа в учетную запись:

|  |  |
| --- | --- |
| Сценарий | Результат |
| Пользователь ввел неправильную почту или пароль |  |
| Пользователь вводит русскими буквами поту |  |

2.10.2. Оптимизация базы данных

Оптимизация моей базы данных направлена на повышение производительности, уменьшение времени отклика системы, снижение нагрузки на сервер и обеспечение масштабируемости при росте числа пользователей и объема данных. Для оптимизации базы данных были применены следующие подходы:

1. На этапе проектирования структура базы данных была приведена к нормальной форме, что позволило устранить избыточность и логические аномалии при обновлении данных. Например, информация о пользователях, сообщениях и друзьях хранится в отдельных таблицах, связанных между собой с помощью внешних ключей.
2. Были созданы индексы по полям, которые часто участвуют в операциях поиска и фильтрации (например, user\_id, email, post\_id). Это ускоряет выполнение SQL-запросов, особенно при больших объемах данных.
3. Настроено регулярное резервное копирование базы данных, а также автоматическое удаление устаревших или временных данных, что позволяет сохранять оптимальный объем базы.

ЗАКЛЮЧЕНИЕ

В ходе изучения технологии разработки и защиты баз данных было выявлено, что успешная работа с данными в современных условиях требует комплексного подхода, включающего как аспекты проектирования, так и обеспечения безопасности. Принимая во внимание стремительное развитие информационных технологий, применение современных методов и инструментов становится неизбежным. Это позволяет не только оптимизировать процессы хранения и обработки данных, но и защитить их от возможных угроз.

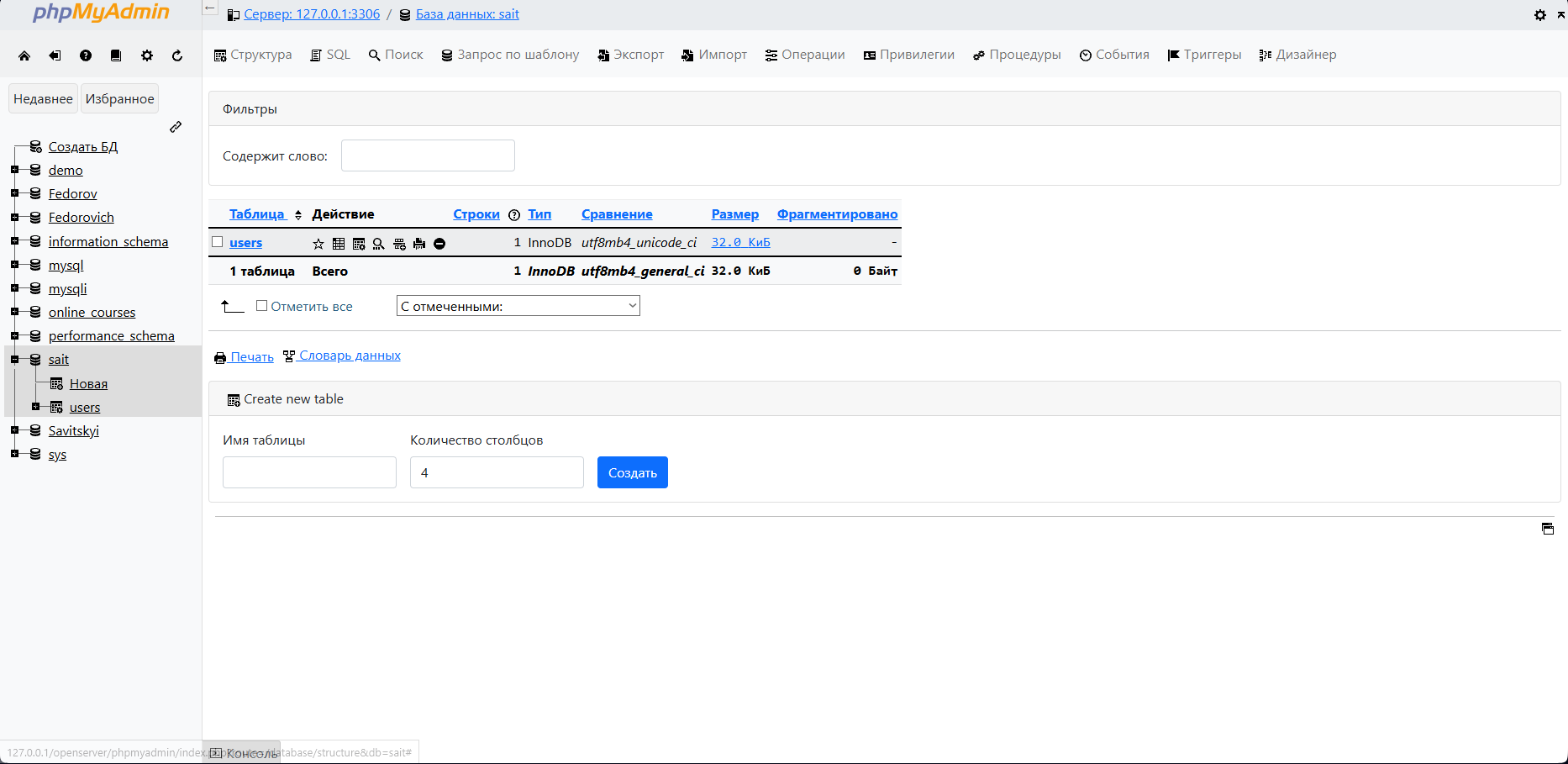
Особое внимание следует уделить вопросам шифрования данных, аутентификации пользователей и резервного копирования. Эффективная защита баз данных требует активного мониторинга и регулярного обновления систем безопасности.

Таким образом, подходы к разработке и защите баз данных должны быть гибкими и адаптивными, чтобы соответствовать быстро меняющимся требованиям и угрозам. В будущем важно продолжать исследовать и внедрять новые технологии, что позволит обеспечить надежность и безопасность информации на всех уровнях.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* Волик, М.В. Разработка базы данных в Access / М.В. Волик. – Издательство «Прометей», 2021. – 88 с.
* Даккет, Дж. PHP и MySQL. Серверная веб-разработка / Дж. Даккет. – Издательство «Эксмо», 2023. – 688 с.
* Дьяков, И.А. Базы данных. Язык SQL : учебное пособие / И.А. Дьяков. – Тамбов : Тамбовский государственный технический университет (ТГТУ), 2022. – 82 с.
* Ла Рокка, М. Продвинутые и структуры данных. / М. ла Рокка. – Издательство «Питер», 2024. – 848 с.
* Никсон, Р. Создаём динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 6-е изд. / Р. Никсон. – Издательство «Питер»: Санкт-Петербург, 2023. – 832 с.
* <https://www.figma.com/> (дата обращения: 15.03.2025).
* <https://www.github.com/> (дата обращения: 16.03.2025).

ПРИЛОЖЕНИЯ

Рисунок 8 - скриншот базы данных

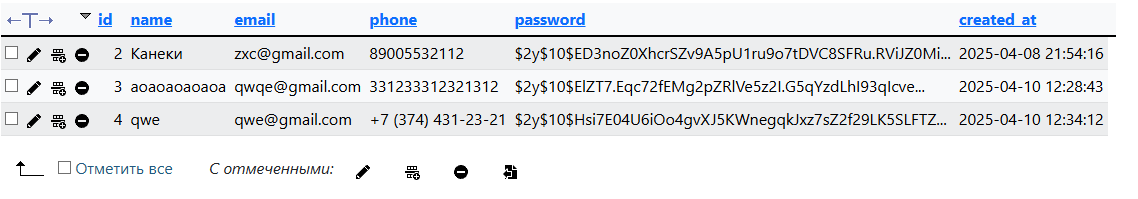


Рисунок 9 - скриншот таблицы