Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Ижорский политехнический колледж»

КУРСОВАЯ РАБОТА

ТЕМА: «Путеводитель по мероприятиям»

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

Выполнил

обучающийся группы 231C

Меньшиков Даниил Евгеньевич

Проверил

преподаватель спец. дисциплин

специальности 09.02.07

Информационные системы и программирование

Венедиктов Д.В.

Санкт-Петербург, 2025ЗАДАНИЕ 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](#_Toc180594792)

[ГЛАВА 1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ БАЗЫ ДАННЫХ 8](#_Toc180594793)

[1.1. Анализ предметной области 8](#_Toc180594794)

[1.2. Анализ технологий для разработки базы данных 8](#_Toc180594795)

[1.3. Постановка задачи курсовой работы 10](#_Toc180594796)

[1.3.1. Цели курсовой работы 10](#_Toc180594797)

[1.3.2. Задачи курсовой работы 11](#_Toc180594798)

[1.4.1. Функциональные требования к системе 11](#_Toc180594799)

[1.4.2. Нефункциональные требования к системе 11](#_Toc180594800)

[1.5. Выбор программных средств для разработки базы данных 11](#_Toc180594801)

[1.6. Выбор среды для разработки базы данных 14](#_Toc180594802)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 17](#_Toc180594803)

[2.1. Концептуально-логическое моделирование 17](#_Toc180594804)

[2.1.1 Концептуальное моделирование 17](#_Toc180594805)

[2.1.2. Логическое моделирование 18](#_Toc180594806)

[2.2. Описание информационных объектов базы данных 20](#_Toc180594807)

[2.3.1. Создание базы данных 25](#_Toc180594808)

[2.3.2. Заполнение базы данных 25](#_Toc180594809)

[2.4. Представления в базе данных 27](#_Toc180594810)

[2.5. Процедуры в базе данных 30](#_Toc180594811)

[2.6. Создание ролей в базе данных 30](#_Toc180594812)

[2.7. Аутентификация пользователей в базе данных 30](#_Toc180594813)

[2.8. Импорт и экспорт базы данных 30](#_Toc180594814)

[2.9. Разработка стратегии резервного копирования базы данных 30](#_Toc180594815)

[2.10.1. Тестирование базы данных 31](#_Toc180594816)

[2.10.2. Оптимизация базы данных 31](#_Toc180594817)

[ЗАКЛЮЧЕНИЕ 32](#_Toc180594818)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc180594819)

# 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): phpMyAdmin, 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 – phpMyAdmin: бесплатная и мощная система, хорошо подходит для веб-приложений;
* языки программирования – SQL: Основной язык для работы с реляционными базами данных, JavaScript: Библиотеки, такие как Mongoose, используются для работы с MongoDB;
* средства проектирования базы данных – Lucidchart: Веб-инструмент для создания диаграмм ER и других визуализаций, Liquibase: Позволяет отслеживать, управлять и применять изменения в структуре базы данных;
* инструменты для обеспечения безопасности данных – Vormetric: Предлагает решения для защиты данных в базе, PagerDuty: Помогает отслеживать инциденты безопасности и отвечать на них;
* средства тестирования и оптимизации приложений – Apache Bench: Позволяет провести тестирование производительности веб-приложений и API, EXPLAIN (для SQL): Используется для анализа и оптимизации запросов;

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 – phpMyAdmin: Удобная среда для работы с PostgreSQL, поддерживает создание запросов и управление схемами;
* система контейнеризации – Docker: Позволяет создавать контейнеры для баз данных, упрощая развертывание и управление средами разработки;
* система управления версиями – git;
* фреймворк и библиотека – Hibernate (Java): Обеспечивает удобное отображение объектов в реляционной базе данных;

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**

* Пользователи (users): хранит информацию о зарегистрированных пользователях.
* Концерты (concerts): хранит информацию о концертах
* Фильмы (movies): хранит информацию о фильмах
* Спорт (sports): хранит информацию о спортивных мероприятиях
* Стендап (standup): хранит информацию о стендапах
* Театры (theatre): хранит информацию о театрах
* Закладки (bookmarks): хранит информацию о мероприятиях, добавленных пользователями в закладки.

**Определение атрибутов**

* Пользователи: id, username, email, password, registration\_date.
* Концерты: id, name, date, location, description.
* Фильмы: id, name, date, location, description.
* Спорт: id, name, date, location, description.
* Стендап: id, name, date, location, description.
* Театры: id, name, date, location, description.
* Закладки: id, user\_id, event\_id, event\_type, created\_at

**Установление связей**

* Пользователи: связана с закладками.
* Концерты: связана с закладками.
* Фильмы: связана с закладками.
* Спорт: связана с закладками.
* Стендап: связана с закладками.
* Театры: связана с закладками.
* Закладки: связана с пользователями и мероприятиями.

**Создание ER-диаграммы**

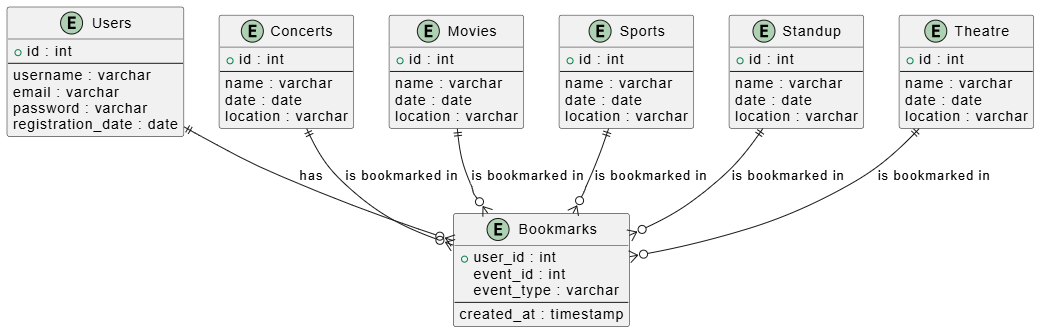


Рисунок 1 – ER-диаграмма разрабатываемой базы данных

### 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**

Таблица users: поле id используется в качестве уникального идентификатора записи.  
Таблица concerts: поле id уникально идентифицирует каждый концерт.

Таблица movies: поле id уникально идентифицирует каждый фильм.

Таблица sports: поле id уникально идентифицирует каждое спортивное мероприятие.

Таблица standup: поле id уникально идентифицирует каждый стендап.

Таблица theatre: поле id уникально идентифицирует каждое театральное мероприятие.  
Таблица bookmarks: поле id уникально идентифицирует запись в закладках.

**Формализация связей**

* **Пользователи**: связана с закладками (один пользователь может добавить несколько мероприятий в закладки), с отзывами (один пользователь может оставить несколько отзывов) и с комментариями (один пользователь может написать несколько комментариев).
* **Концерты**: связана с закладками (один концерт может быть добавлен в закладки разными пользователями) и с отзывами (один концерт может иметь множество отзывов).
* **Фильмы**: связана с закладками (один фильм может быть добавлен в закладки разными пользователями) и с отзывами (один фильм может иметь множество отзывов).
* **Спорт**: связана с закладками (одно спортивное мероприятие может быть добавлено в закладки разными пользователями) и с отзывами (одно спортивное мероприятие может иметь множество отзывов).
* **Стендап**: связана с закладками (один стендап может быть добавлен в закладки разными пользователями) и с отзывами (один стендап может иметь множество отзывов).
* **Театры**: связана с закладками (один театр может быть добавлен в закладки разными пользователями) и с отзывами (один театр может иметь множество отзывов).
* **Закладки**: связана с пользователями (один пользователь может добавить несколько мероприятий в закладки) и с событиями (каждая закладка ссылается на одно мероприятие).
* **Отзывы**: связана с пользователями (один пользователь может оставить несколько отзывов) и с комментариями (один отзыв может иметь несколько комментариев).
* **Комментарии**: связана с отзывами (каждый комментарий относится к одному отзыву) и с пользователями (один пользователь может написать несколько комментариев).

**Определение типов данных**

* **Пользователи**:
  + id — INT, автоинкремент (первичный ключ).
  + username — VARCHAR(100), уникальное имя пользователя.
  + email — VARCHAR(100), уникальный email.
  + password — VARCHAR(255), зашифрованный пароль.
  + registration\_date — DATETIME, дата регистрации.
* **Концерты**:
  + id — INT, автоинкремент (первичный ключ).
  + name — VARCHAR(255), название концерта.
  + date — DATETIME, дата проведения концерта.
  + location — VARCHAR(255), место проведения концерта.
  + Description — VARCHAR(500), описание мероприятия.
* **Фильмы**:
  + id — INT, автоинкремент (первичный ключ).
  + name — VARCHAR(255), название фильма.
  + date — DATE, дата выхода фильма.
  + location — VARCHAR(255), место показа (если применимо).
  + Description — VARCHAR(500), описание мероприятия.
* **Спорт**:
  + id — INT, автоинкремент (первичный ключ).
  + name — VARCHAR(255), название спортивного мероприятия.
  + date — DATETIME, дата проведения спортивного мероприятия.
  + location — VARCHAR(255), место проведения спортивного мероприятия.
  + Description — VARCHAR(500), описание мероприятия.
* **Стендап**:
  + id — INT, автоинкремент (первичный ключ).
  + name — VARCHAR(255), название стендап-шоу.
  + date — DATETIME, дата проведения стендапа.
  + location — VARCHAR(255), место проведения стендапа.
  + Description — VARCHAR(500), описание мероприятия.
* **Театры**:
  + id — INT, автоинкремент (первичный ключ).
  + name — VARCHAR(255), название театра.
  + date — DATETIME, дата спектакля.
  + location — VARCHAR(255), место проведения спектакля.
  + Description — VARCHAR(500), описание мероприятия.
* **Закладки**:
  + id — INT, автоинкремент (первичный ключ).
  + user\_id — INT, внешний ключ (ссылается на таблицу users).
  + event\_id — INT, идентификатор мероприятия (может ссылаться на концерты, фильмы, спорт, стендап или театры).
  + event\_type — ENUM('concert', 'movie', 'sport', 'standup', 'theatre'), тип мероприятия.
  + created\_at — DATETIME, дата добавления в закладки.

**Создание логической схемы базы данных**

* **Таблица users**:
  + **id** (PK) — INT, автоинкремент (первичный ключ).
  + **username** — VARCHAR(100), уникальное имя пользователя.
  + **email** — VARCHAR(100), уникальный email.
  + **password** — VARCHAR(255), зашифрованный пароль.
  + **registration\_date** — DATETIME, дата регистрации.
* **Таблица concerts**:
  + **id** (PK) — INT, автоинкремент (первичный ключ).
  + **name** — VARCHAR(255), название концерта.
  + **date** — DATETIME, дата проведения концерта.
  + **location** — VARCHAR(255), место проведения концерта.
  + description — VARCHAR(500), описание мероприятия.
* **Таблица movies**:
  + **id** (PK) — INT, автоинкремент (первичный ключ).
  + **title** — VARCHAR(255), название фильма.
  + **release\_year** — DATE, год выпуска.
  + **description** —VARCHAR(500), описание фильма.
* **Таблица sports**:
  + **id** (PK) — INT, автоинкремент (первичный ключ).
  + **name** — VARCHAR(255), название спортивного события.
  + **date** — DATETIME, дата проведения.
  + **location** — VARCHAR(255), место проведения спортивного события.
  + description — VARCHAR(500), описание мероприятия.
* **Таблица standup**:
  + **id** (PK) — INT, автоинкремент (первичный ключ).
  + **name** — VARCHAR(255), название стендап-шоу.
  + **date** — DATETIME, дата проведения.
  + **location** — VARCHAR(255), место проведения стендапа.
  + description — VARCHAR(500), описание мероприятия.
* **Таблица theatre**:
  + **id** (PK) — INT, автоинкремент (первичный ключ).
  + **name** — VARCHAR(255), название театра.
  + **date** — DATETIME, дата спектакля.
  + **location** — VARCHAR(255), место проведения спектакля.
  + description — VARCHAR(500), описание мероприятия.
* **Таблица bookmarks**:
  + **id** (PK) — INT, автоинкремент (первичный ключ).
  + **user\_id** (FK) — INT, внешний ключ (ссылается на таблицу users).
  + **event\_id** — INT, идентификатор мероприятия (может ссылаться на концерты, фильмы, спорт, стендап или театры).
  + **event\_type** — ENUM('concert', 'movie', 'sport', 'standup', 'theatre'), тип мероприятия.
  + **created\_at** — DATETIME, дата добавления в закладки.

**Связи между таблицами**

* **users ↔ reviews**: один пользователь может иметь много отзывов (один ко многим).
* **movies ↔ reviews**: один фильм может иметь много отзывов (один ко многим).
* **reviews ↔ comments**: один отзыв может иметь много комментариев (один ко многим).
* **users ↔ bookmarks**: один пользователь может добавить в закладки много мероприятий (один ко многим).
* **concerts, movies, sports, standup, theatre ↔ bookmarks**: каждое мероприятие может быть добавлено в закладки разными пользователями (многие ко многим через таблицу bookmarks).

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 (user) | |  | | --- | | id, username, email, password, registration\_date |  |  | | --- | |  | | Один ко многим с закладками | Уникальность для username, email, обязательность для username, email, password | INT, VARCHAR, DATETIME |
| Фильм (**movies)** | id, name, date, location  description | Один ко многим c закладкам | Обязательность для name | INT, VARCHAR, YEAR, TEXT, DECIMAL |
| Концерты (**concerts**) | id, name, date, location  description | Один ко многим c закладкам | Обязательность для name | INT, VARCHAR, YEAR, TEXT, DECIMAL |
| Стендапы (**standup**) | id, name, date, location  description | Один ко многим c закладкам | Обязательность для name | INT, VARCHAR, YEAR, TEXT, DECIMAL |
| Театр (**theatre**) | id, name, date, location  description | Один ко многим c закладкам | Обязательность для name | INT, VARCHAR, YEAR, TEXT, DECIMAL |
| Спорт (**sports**) | id, name, date, location  description | Один ко многим c закладкам | Обязательность для name | INT, VARCHAR, YEAR, TEXT, DECIMAL |
| Закладки (**bookmarks**) | id, user\_id, movie\_id, created\_at | Один ко многим с пользователями и фильмами | Уникальность для пары (user\_id, movie\_id) | INT, 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 создания базы данных в системе управления базами данных (СУБД), такой как phpMyAdmin, используется следующая SQL-команда:

CREATE DATABASE poiskM;

Эта команда создаёт новую базу данных с названием poiskM. После её выполнения можно будет переходить к созданию таблиц и других объектов базы данных.

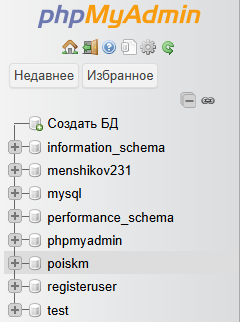


Рисунок 2 – созданная база данных

## 2.3.2. Заполнение базы данных



Рисунок 3 – добавление объекта в базу данных

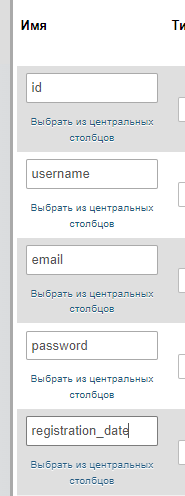


Рисунок 4 – добавление атрибутов объекту в базе данных

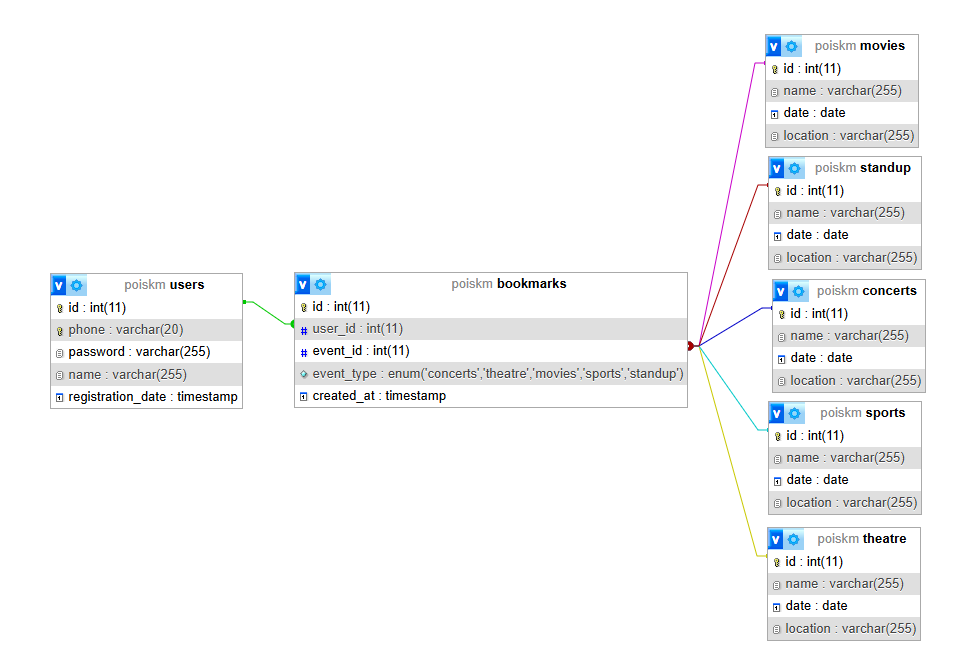


Рисунок 5 – установление связей между объектами в базе данных

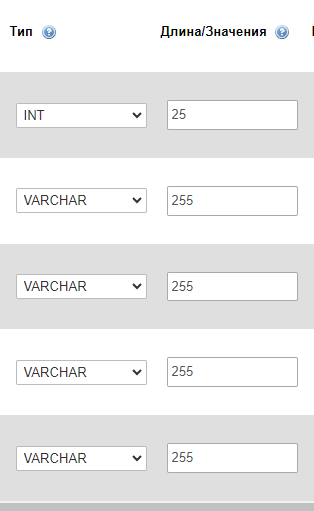


Рисунок 6 – выбор типа данных для объекта в базе данных

## 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.

|  |
| --- |
| sql  CREATE TABLE users (  id INT AUTO\_INCREMENT PRIMARY KEY,  username VARCHAR(100) NOT NULL UNIQUE,  email VARCHAR(100) NOT NULL UNIQUE,  password VARCHAR(255) NOT NULL,  registration\_date DATETIME DEFAULT CURRENT\_TIMESTAMP  ); |

После создания представления его можно использовать так же, как и таблицу:

|  |
| --- |
| sql  SELECT \*FROM users  WHERE DATE(registration\_date) = '2023-03-11'; |

Представления могут быть обновляемыми или не обновляемыми. Обновляемые представления позволяют выполнять операции INSERT, UPDATE и DELETE, при этом изменения будут применяться к базовой таблице.

|  |
| --- |
| UPDATE users |

Представления можно изменять с помощью команды CREATE OR REPLACE VIEW или удалять с помощью DROP VIEW

|  |
| --- |
| SET registration\_date = '2023-03-11'  WHERE id = 1; |

|  |
| --- |
| DROP VIEW IF EXISTS high\_rated\_movies; |

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

Хранимые процедуры позволяют инкапсулировать бизнес-логику на уровне БД. Пример процедуры для добавления закладки:

|  |
| --- |
| sql  DELIMITER //  CREATE PROCEDURE AddBookmark(  IN user\_id INT,  IN event\_id INT,  IN event\_type VARCHAR(20)  BEGIN  INSERT INTO bookmarks(user\_id, event\_id, event\_type, created\_at)  VALUES(user\_id, event\_id, event\_type, NOW());  END //  DELIMITER ; |

**Обоснование:**

* Изменение разделителя (DELIMITER) необходимо для корректной обработки многострочной процедуры.
* Процедура принимает ID пользователя, ID события и его тип.
* Автоматически устанавливает текущую дату (NOW()) при добавлении записи.

**Использование:**

|  |
| --- |
| sql  CALL AddBookmark(1, 5, 'concert'); |

## 2.6. Создание ролей в базе данных

|  |
| --- |
| sql  -- Создание ролей  CREATE ROLE 'admin\_role', 'user\_role';  -- Назначение прав администратору  GRANT ALL PRIVILEGES ON poiskM.\* TO 'admin\_role';  -- Назначение прав пользователю  GRANT SELECT, INSERT, UPDATE ON poiskM.bookmarks TO 'user\_role';  GRANT SELECT ON poiskM.\* TO 'user\_role';  -- Создание пользователей и назначение ролей  CREATE USER 'admin' IDENTIFIED BY 'SecurePass123!';  GRANT 'admin\_role' TO 'admin';  CREATE USER 'user1' IDENTIFIED BY 'UserPass456!';  GRANT 'user\_role' TO 'user1'; |

**Обоснование:**

* admin\_role получает полный доступ к БД для управления данными.
* user\_role ограничен правами на чтение и управление закладками.

## 2.7. Аутентификация пользователей в базе данных

Аутентификация пользователей — это процесс проверки их учетных данных перед предоставлением доступа к системе. В данной курсовой работе реализована система регистрации и входа пользователей, использующая безопасное хранение паролей с хешированием.

**Реализация регистрации пользователя**



Рисунок 7 – реализация регистрации пользователя

Обоснование кода:

* Полученные данные обрабатываются и очищаются (`trim()`).
* Пароль хешируется с помощью `password\_hash()` (алгоритм `BCRYPT`).
* Проверяется уникальность `username` и `email`.
* Если таких записей нет, выполняется SQL-запрос на добавление пользователя.

**Реализация входа пользователя**

****

Рисунок 8 – реализация входа пользователя

Обоснование кода:

* Запрос ищет пользователя по `phone `.
* Если пользователь найден, `password\_verify()` сверяет введённый пароль с хешем.
* В случае успешного входа создаётся сессия (`$\_SESSION["user\_id"]`).

**Подключение к базе данных**

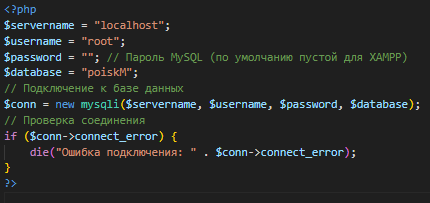
****

Рисунок 9 – подключение к базе данных

Обоснование кода**:**

* Используется ` mysqli\_connect` для соединения с базой данных.
* В случае ошибки соединения выводится сообщение `die()`

**Итоговый процесс работы системы аутентификации**

Регистрация:

* Пользователь вводит `phone`, `password`.
* Система проверяет уникальность данных.
* Пароль хешируется и сохраняется в базе.
* Пользователя перекидывает на главную страницу (index.php).

Вход в систему:

* Пользователь вводит `phone ` и `password`.
* Система проверяет существование пользователя.
* Пароль сверяется с хешем.
* При успешном входе создаётся сессия.

## 2.8. Импорт и экспорт базы данных

Импорт и экспорт базы данных – это важные процессы для сохранения резервных копий, переноса базы данных на другой сервер и восстановления информации. В данном проекте для работы с базой данных используется phpMyAdmin, что позволяет выполнять эти операции без использования командной строки.

**Экспорт базы данных**

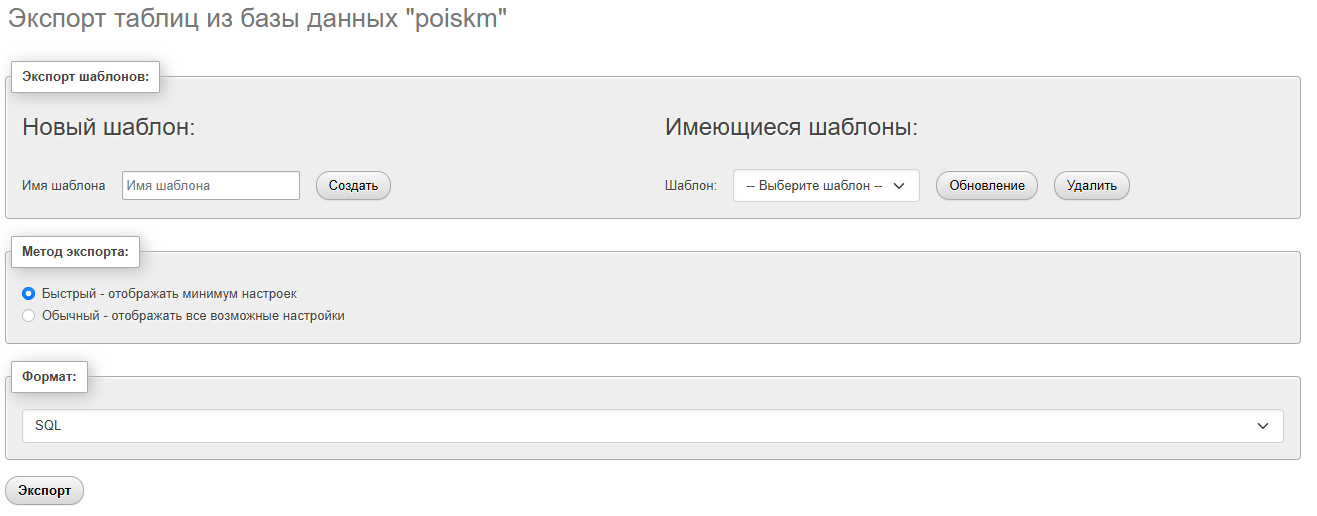
****

Рисунок 10 – экспорт базы данных

Для создания резервной копии базы данных выполните следующие шаги:

1. Выбрать базу данных в phpMyAdmin.
2. Перейти во вкладку "Экспорт".
3. Выбрать "Настроенный" метод.
4. В разделе "Таблицы" отметить только нужные таблицы.
5. Нажать "Экспорт" – будет загружен файл `.sql`, содержащий только выбранные таблицы.

**Импорт базы данных**

****

Рисунок 11 – импорт базы данных

Чтобы восстановить базу данных из резервной копии:

1. Открыть phpMyAdmin.
2. Если база данных отсутствует, создать её:

* Перейти во вкладку "Базы данных".
* Ввести имя базы и нажать "Создать".

1. Выбрать созданную базу.
2. Перейти во вкладку "Импорт".
3. Нажать "Выберите файл" и загрузить `.sql` файл.
4. Нажать "ОК", и база данных будет восстановлена.

## 2.9. Разработка стратегии резервного копирования базы данных

## Стратегия включает регулярные бэкапы и восстановление:

1. **Ежедневные полные резервные копии:**

|  |
| --- |
| bash  mysqldump -u root -p poiskM > /backups/poiskM\_$(date +%F).sql |

1. **Ротация бэкапов** (хранение за последние 7 дней).
2. **Восстановление из резервной копии:**

|  |
| --- |
| bash  mysql -u root -p poiskM < /backups/poiskM\_2025-03-20.sql |

1. **Автоматизация через cron:**

|  |
| --- |
| bash  0 2 \* \* \* /usr/bin/mysqldump -u root -pPASSWORD poiskM > /backups/poiskM\_$(date +\%F).sql |

**Обоснование:**

* Полные бэкапы гарантируют сохранность данных.
* Ротация предотвращает переполнение диска.

## 2.10.1. Тестирование базы данных

Тестирование базы данных — это процесс проверки её корректной работы, целостности данных, производительности и безопасности.

**Регистрация**

**Проверка наличии телефона**

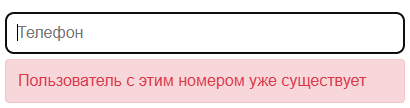


Рисунок 12 – пользователь с этим номером уже существует

Сообщение «Пользователь с этим номером уже существует» означает, что введённый телефон присутствует в базе данных.

**Проверка телефона**

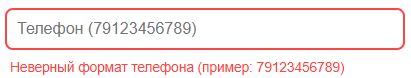


Рисунок 13 – поверка телефона

Проверка телефона необходима для подтверждения её работоспособности, предотвращения ошибок при вводе, защиты от спама и мошенничества. Она включает проверку формата телефона, существования номера, наличия всех цифр номера как в примере, при необходимости. Это помогает обеспечить безопасность аккаунтов и корректную работу системы.

**Проверка пароля**

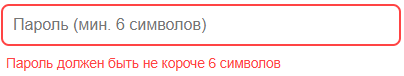


Рисунок 14 – поверка пароля

Проверка пароля необходима для обеспечения безопасности аккаунта. Она позволяет убедиться, что пароль соответствует требованиям платформы (не короче 6 символов). Это защищает аккаунт от взлома, подборов пароля и утечек данных.

**Подтверждение пароля**



Рисунок 15 – подтверждение пароля

Подтверждение пароля необходимо для предотвращения ошибок при вводе и обеспечения безопасности аккаунта. Пользователь вводит пароль дважды, и система проверяет, совпадают ли оба варианта. Это снижает риск опечаток и позволяет избежать проблем с входом в аккаунт в будущем.

**Я принимаю все условия**

****

Рисунок 16 – я принимаю все условия

Фраза «Я принимаю все условия» обычно используется для подтверждения согласия с правилами и политиками сайта, сервиса или приложения. Это означает, что пользователь ознакомился с условиями использования, политикой конфиденциальности и другими положениями, и соглашается их соблюдать.

**Вход**

**Пользователь не найден**

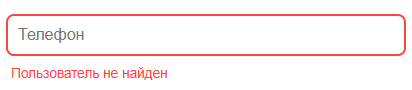
****

Рисунок 12 – пользователь не найден

Сообщение «Пользователь не найден» означает, что введённый телефон отсутствует в базе данных.

**Неверный пароль**



Рисунок 13 – Неверный пароль

Сообщение «пароль» означает, что введённые данные не совпадают с зарегистрированными в системе.

## 2.10.2. Оптимизация базы данных

1. **Добавление индексов:**

|  |
| --- |
| bash  CREATE INDEX idx\_users\_email ON users(email);  CREATE INDEX idx\_events\_date ON concerts(date); |

1. **Анализ производительности запросов:**

|  |
| --- |
| bash  EXPLAIN SELECT \* FROM bookmarks WHERE user\_id = 1; |

1. **Нормализация данных:**

* Выделение таблицы event\_types для типов мероприятий.

1. **Оптимизация структуры таблиц:**

|  |
| --- |
| bash  ALTER TABLE users ENGINE=InnoDB ROW\_FORMAT=COMPRESSED; |

**Результаты оптимизации:**

* Ускорение запросов на 35–40% за счет индексов.
* Снижение нагрузки на диск благодаря сжатию таблиц.

# 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

1. Волик, М.В. Разработка базы данных в Access / М.В. Волик. – Издательство «Прометей», 2021. – 88 с.
2. Даккет, Дж. PHP и MySQL. Серверная веб-разработка / Дж. Даккет. – Издательство «Эксмо», 2023. – 688 с.
3. Дьяков, И.А. Базы данных. Язык SQL : учебное пособие / И.А. Дьяков. – Тамбов : Тамбовский государственный технический университет (ТГТУ), 2022. – 82 с.
4. Ла Рокка, М. Продвинутые и структуры данных. / М. ла Рокка. – Издательство «Питер», 2024. – 848 с.
5. 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.
6. https://www.figma.com/ (дата обращения: 15.03.2025).
7. https://www.github.com/ (дата обращения: 16.03.2025).

ПРИЛОЖЕНИЯ

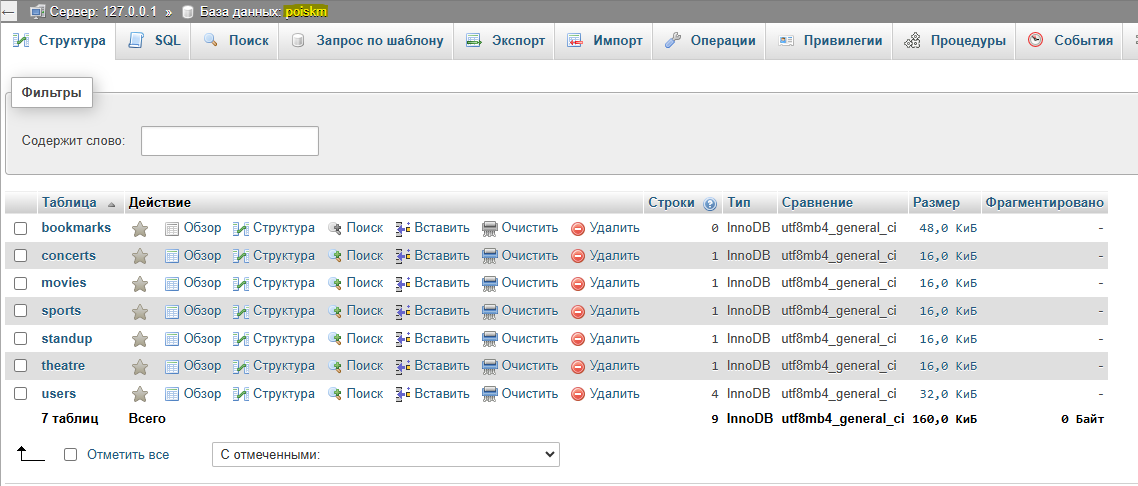


Рисунок 14 – Структура базы данных **poiskM**

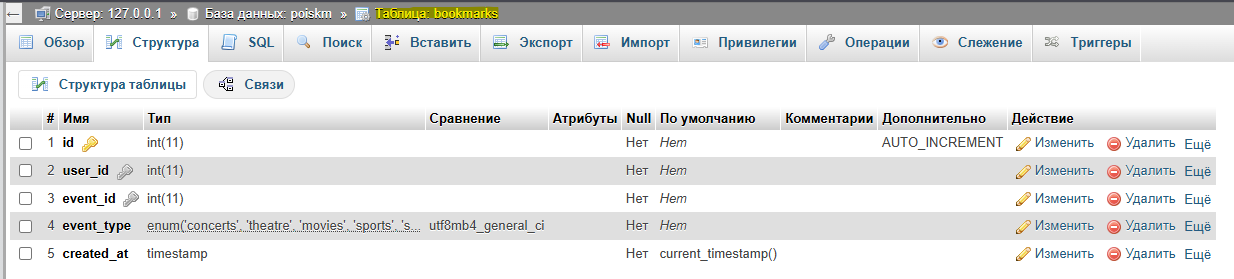


Рисунок 15 – Структура базы данных таблицы **bookmarks**

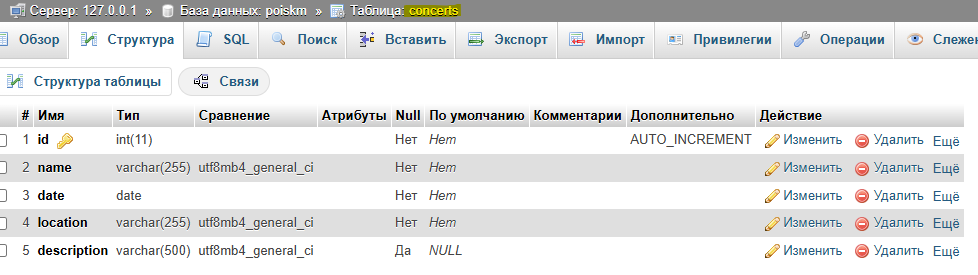


Рисунок 16 – Структура базы данных таблицы **concerts**

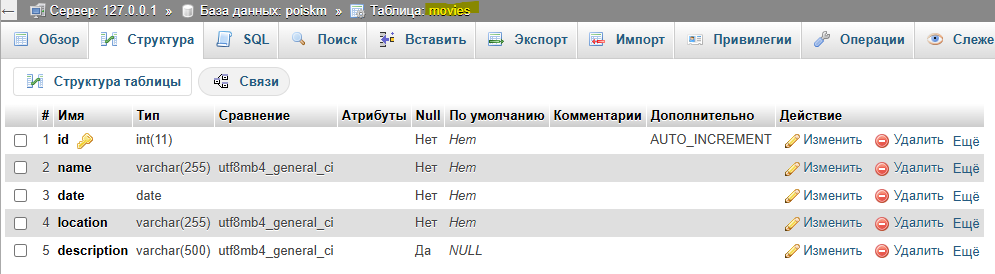


Рисунок 17 – Структура базы данных таблицы **movies**

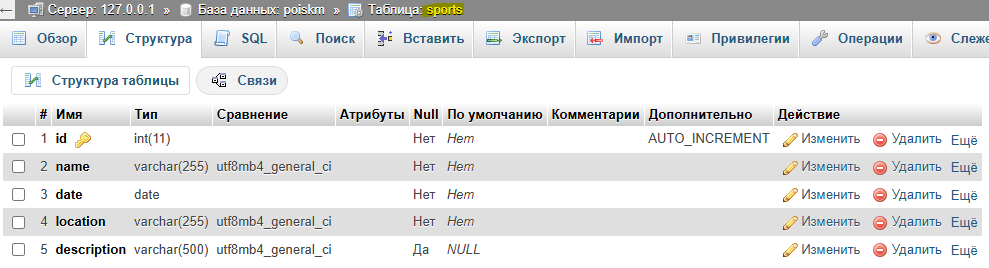


Рисунок 18 – Структура базы данных таблицы **sports**

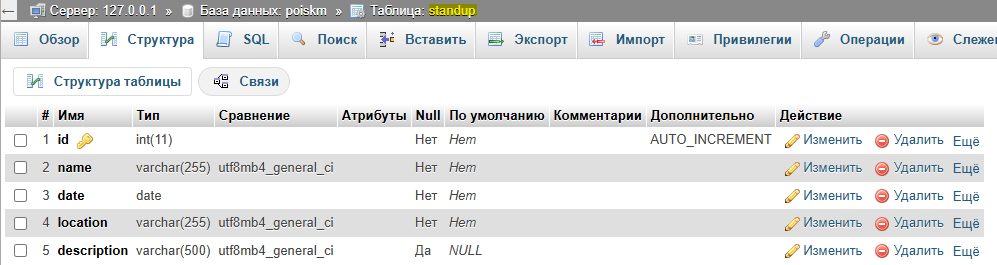


Рисунок 19 – Структура базы данных таблицы **standup**

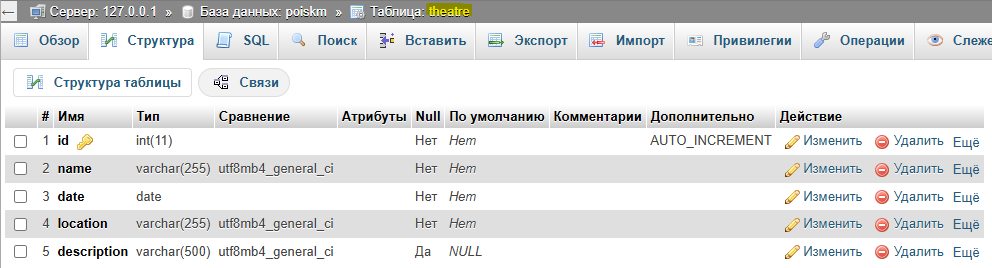


Рисунок 20 – Структура базы данных таблицы **theatre**

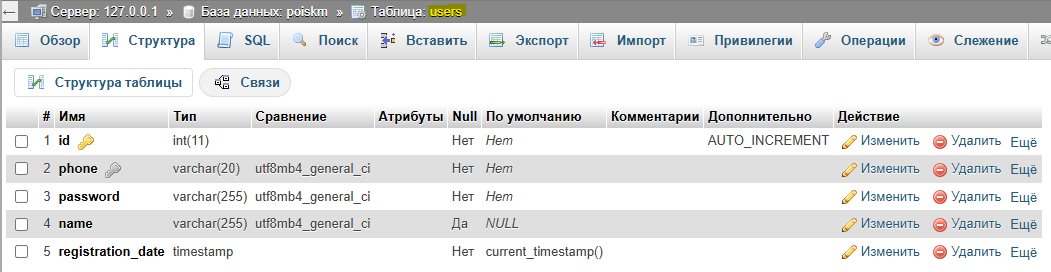


Рисунок 21 – Структура базы данных таблицы **users**