Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Ижорский политехнический колледж»

КУРСОВАЯ РАБОТА

ТЕМА: «Форум»

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

Выполнил

обучающийся группы 231с

Гришин Роман Алексеевич

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](#_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): 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, PHP, JavaScript;
* средства проектирования базы данных – dbdiagram.io, Liquibase;
* инструменты для обеспечения безопасности данных – Vormetric, Splunk;
* средства тестирования и оптимизации приложений – Apache Bench, 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 – VS code;
* система контейнеризации – Docker;
* система управления версиями – git;
* фреймворк и библиотека – laravel;

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 (User)** – участник форума, который создает темы и сообщения.

**Тема (Topic/Thread)** – обсуждение, содержащее сообщения пользователей.

**Сообщение (Post/Comment)** – текст, написанный пользователем в теме.

**Категория (Category)** – раздел, к которому относятся темы.

**Тег (Tag)** – метка для группировки тем по тематике.

**Лайк/Реакция (Like/Reaction)** – оценка сообщения пользователем.

**Жалоба (Report)** – уведомление о нарушении правил форума.

**Уведомление (Notification)** – оповещение пользователя о событиях.

**Определение атрибутов**

#### **1. Пользователь (User)**

**UserID**, **Username**, **Password**, **Email**, **RegistrationDate**, **Role**, **Avatar**, **Status**

#### **2. Тема (Topic/Thread)**

**TopicID**, **Title**, **CreationDate**, **AuthorID**, **CategoryID**, **Status**, **ViewCount**

#### **3. Сообщение (Post/Comment)**

**PostID**, **Content**, **PostDate**, **AuthorID**, **TopicID**, **ParentPostID**, **Status**

#### **4. Категория (Category)**

**CategoryID**, **Name**, **Description**, **ParentCategoryID**, **AccessLevel**

#### **5. Тег (Tag)**

**TagID**, **TagName**, **Description**

#### **6. Лайк/Реакция (Like/Reaction)**

**ReactionID**, **UserID**, **PostID**, **ReactionType**

#### **7. Жалоба (Report)**

**ReportID**, **ReporterID**, **ReportedPostID/ReportedTopicID**, **Reason**, **Status**

#### **8. Уведомление (Notification)**

**NotificationID**, **RecipientID**, **Type**, **LinkedEntityID**, **IsRead**

**Установление связей**

**Пользователь** создает **Темы** и **Сообщения**.

**Темы** содержат **Сообщения** и могут относиться к **Категориям**.

**Сообщения** могут получать **Реакции** от пользователей.

**Темы** могут быть помечены **Тегами**.

Пользователи могут отправлять **Жалобы** на контент.

Система отправляет **Уведомления** пользователям.

**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**

**User** - **UserID** (автоинкремент)

**Topic** - **TopicID** (автоинкремент)

**Post** - **PostID** (автоинкремент)

**Category** - **CategoryID** (автоинкремент)

**Tag** - **TagID** (автоинкремент)

**Reaction** - **ReactionID** (автоинкремент)

**Report** - **ReportID** (автоинкремент)

**Notification** - **NotificationID** (автоинкремент)

**TopicTag** - составной ключ (**TopicID** + **TagID**)

**Формализация связей**

**Один-ко-многим (1:N):**

**User** → **Topic** (**AuthorID** в Topic)

**User** → **Post** (**AuthorID** в Post)

**Topic** → **Post** (**TopicID** в Post)

**Category** → **Topic** (**CategoryID** в Topic)

**Многие-ко-многим (M:N):**

**Topic** ↔ **Tag** → таблица **TopicTag** (**TopicID** + **TagID**)

**Рекурсивные:**

**Post** → **Post** (**ParentPostID** для ответов)

**Category** → **Category** (**ParentCategoryID** для вложенности)

**Прочие:**

**User** → **Reaction** (**UserID** в Reaction)

**Post** → **Reaction** (**PostID** в Reaction)

**User** → **Report** (**ReporterID** в Report)

**User** → **Notification** (**RecipientID** в Notification)

**Определение типов данных**

**1. User**

**UserID** - INT (PK)

**Username** - VARCHAR(50)

**Password** - VARCHAR(255)

**Email** - VARCHAR(100)

**RegistrationDate** - DATETIME

**Role** - ENUM('user','moderator','admin')

**Avatar** - VARCHAR(255)

**Status** - ENUM('active','banned')

**2. Topic**

**TopicID** - INT (PK)

**Title** - VARCHAR(200)

**CreationDate** - DATETIME

**AuthorID** - INT (FK)

**CategoryID** - INT (FK)

**Status** - ENUM('open','closed','deleted')

**ViewCount** - INT

**3. Post**

**PostID** - INT (PK)

**Content** - TEXT

**PostDate** - DATETIME

**AuthorID** - INT (FK)

**TopicID** - INT (FK)

**ParentPostID** - INT (FK, NULL)

**Status** - ENUM('published','deleted','edited')

**4. Category**

**CategoryID** - INT (PK)

**Name** - VARCHAR(100)

**Description** - TEXT

**ParentCategoryID** - INT (FK, NULL)

**AccessLevel** - ENUM('public','moderators')

**5. Tag**

**TagID** - INT (PK)

**TagName** - VARCHAR(50)

**Description** - TEXT

**6. Reaction**

**ReactionID** - INT (PK)

**UserID** - INT (FK)

**PostID** - INT (FK)

**ReactionType** - ENUM('like','dislike','laugh')

**7. Report**

**ReportID** - INT (PK)

**ReporterID** - INT (FK)

**ReportedPostID** - INT (FK, NULL)

**ReportedTopicID** - INT (FK, NULL)

**Reason** - TEXT

**Status** - ENUM('pending','resolved')

**8. Notification**

**NotificationID** - INT (PK)

**RecipientID** - INT (FK)

**Type** - ENUM('reply','report','system')

**LinkedEntityID** - INT

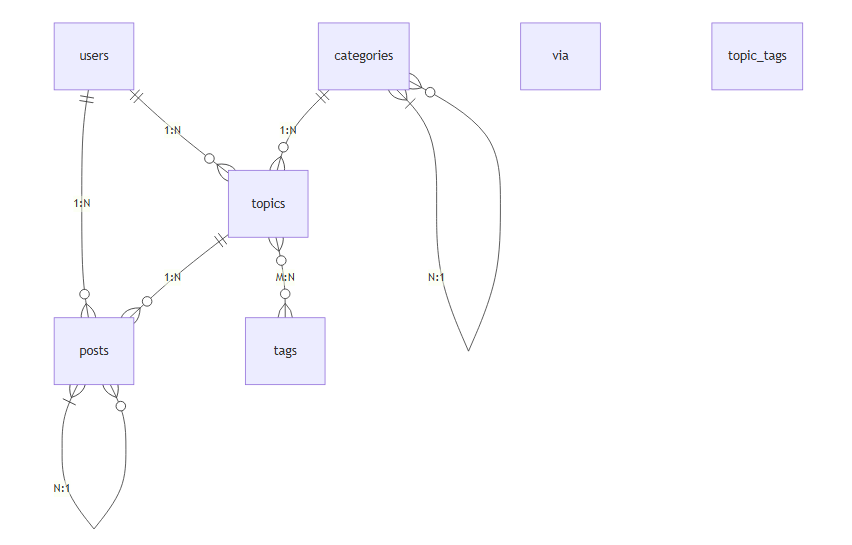
**IsRead** - BOOLEAN

**9. TopicTag**

**TopicID** - INT (PK, FK)

**TagID** - INT (PK, FK)

**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 |
| **User** | user\_id, username, email, password\_hash, role, created\_at | 1→N с Topic, Post, Reaction, Report, Notification | email UNIQUE, username NOT NULL | INT (PK), VARCHAR(50-255), ENUM('user','moder','admin'), DATETIME |
| **Topic** | topic\_id, title, user\_id, category\_id, created\_at, status | 1→N Post, N→1 User, N→1 Category, M→N Tag | title NOT NULL, status DEFAULT 'open' | INT (PK), VARCHAR(200), INT (FK), DATETIME, ENUM('open','closed') |
| **Post** | post\_id, content, user\_id, topic\_id, parent\_post\_id, created\_at | 1→N Reaction, N→1 User, N→1 Topic, N→1 Post (рекурсивная) | content NOT NULL | INT (PK), TEXT, INT (FK), DATETIME |
| **Category** | category\_id, name, parent\_id | 1→N Topic, N→1 Category (рекурсивная) | name NOT NULL | INT (PK), VARCHAR(100), INT (FK NULL) |
| **Tag** | tag\_id, name | M→N Topic (через TopicTag) | name UNIQUE | INT (PK), VARCHAR(50) |
| **TopicTag** | topic\_id, tag\_id | N→1 Topic, N→1 Tag | Составной PK | INT (PK,FK) |
| **Reaction** | reaction\_id, user\_id, post\_id, type | N→1 User, N→1 Post | Одна реакция пользователя на пост | INT (PK), INT (FK), ENUM('like','dislike' |

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

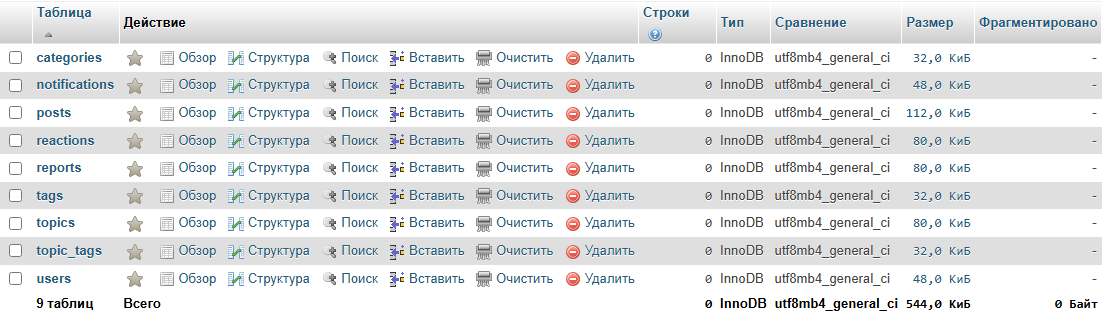


Рисунок 2 – созданная база данных

## 2.3.2. Заполнение базы данных

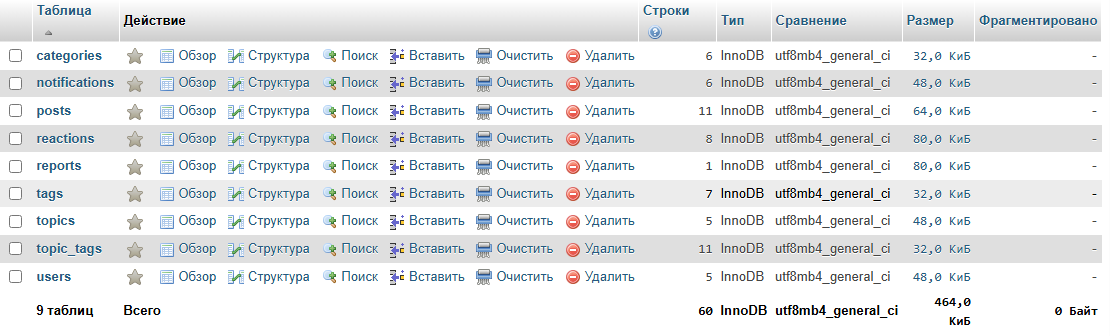


Рисунок 3 – добавление объекта в базу данных

## 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.

|  |
| --- |
| -- 1. Активные пользователи  CREATE VIEW active\_users AS  SELECT user\_id, username, email FROM users WHERE status = 'active';  -- 2. Открытые темы с кол-вом сообщений  CREATE VIEW open\_topics AS  SELECT t.topic\_id, t.title, u.username, COUNT(p.post\_id) as posts  FROM topics t  JOIN users u ON t.user\_id = u.user\_id  LEFT JOIN posts p ON t.topic\_id = p.topic\_id  WHERE t.status = 'open'  GROUP BY t.topic\_id;  -- 3. Популярные темы (20+ просмотров)  CREATE VIEW popular\_topics AS  SELECT topic\_id, title, view\_count FROM topics WHERE view\_count > 20;  -- 4. Сообщения с реакциями  CREATE VIEW post\_reactions AS  SELECT p.post\_id, COUNT(r.reaction\_id) as reactions  FROM posts p LEFT JOIN reactions r ON p.post\_id = r.post\_id  GROUP BY p.post\_id;  -- 5. Темы с тегами  CREATE VIEW topic\_tags\_view AS  SELECT t.topic\_id, t.title, GROUP\_CONCAT(tg.name) as tags  FROM topics t  JOIN topic\_tags tt ON t.topic\_id = tt.topic\_id  JOIN tags tg ON tt.tag\_id = tg.tag\_id  GROUP BY t.topic\_id; |

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

|  |
| --- |
| -- 1. Получить всех активных пользователей  SELECT \* FROM active\_users;  -- 2. Найти открытые темы с более чем 5 сообщениями  SELECT \* FROM open\_topics WHERE posts > 5;  -- 3. Показать топ-5 популярных тем  SELECT \* FROM popular\_topics ORDER BY view\_count DESC LIMIT 5;  -- 4. Найти сообщения с 3+ реакциями  SELECT \* FROM post\_reactions WHERE reactions >= 3;  -- 5. Поиск тем по тегу (например, 'MySQL')  SELECT \* FROM topic\_tags\_view WHERE tags LIKE '%MySQL%'; |

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

|  |
| --- |
| -- 1. Обновить статус пользователя (если представление updatable)  UPDATE active\_users  SET status = 'banned'  WHERE user\_id = 3;  -- 2. Переименовать тему через представление  UPDATE open\_topics  SET title = 'Новое название'  WHERE topic\_id = 5;  -- 3. Изменить категорию популярной темы  UPDATE popular\_topics  SET category = 'Базы данных'  WHERE topic\_id = 2; |

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

|  |
| --- |
| -- 1. Добавить дату создания в представление активных пользователей  CREATE OR REPLACE VIEW active\_users AS  SELECT user\_id, username, email, created\_at  FROM users  WHERE status = 'active';  -- 2. Расширить представление тем с количеством просмотров  CREATE OR REPLACE VIEW open\_topics AS  SELECT t.topic\_id, t.title, u.username, t.view\_count, COUNT(p.post\_id) as posts  FROM topics t  JOIN users u ON t.user\_id = u.user\_id  LEFT JOIN posts p ON t.topic\_id = p.topic\_id  WHERE t.status = 'open'  GROUP BY t.topic\_id;  -- 3. Обновить представление для популярных тем (порог 50+ просмотров)  CREATE OR REPLACE VIEW popular\_topics AS  SELECT topic\_id, title, view\_count  FROM topics  WHERE view\_count > 50; |

|  |
| --- |
| -- 1. Удалить представление активных пользователей  DROP VIEW IF EXISTS active\_users;  -- 2. Удалить представление открытых тем  DROP VIEW open\_topics;  -- 3. Удалить несколько представлений сразу  DROP VIEW popular\_topics, post\_reactions; |

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

### **1. Создание темы**

**Вызов:** **CALL create\_topic(1, 'Новая тема', 3);**

### **2. Добавление поста**

**Вызов:** **CALL add\_post(1, 5, 'Мой комментарий');**

### **3. Бан пользователя**

**Вызов:** **CALL ban\_user(3);**

### **4. Подсчёт постов в теме**

**Вызов:** **CALL count\_posts(5, @result); SELECT @result;**

### **5. Удаление темы**

### **Вызов:** **CALL delete\_topic(7);**

## 2.6. Создание ролей в базе данных

|  |
| --- |
| -- 1. Создание ролей  CREATE ROLE forum\_admin;  CREATE ROLE forum\_moderator;  CREATE ROLE forum\_user;  -- 2. Назначение прав администратору  GRANT ALL PRIVILEGES ON forum\_db.\* TO forum\_admin;  -- 3. Назначение прав модератору  GRANT SELECT, INSERT, UPDATE, DELETE ON forum\_db.\* TO forum\_moderator;  REVOKE DELETE ON forum\_db.users FROM forum\_moderator; -- Не может удалять пользователей  -- 4. Назначение прав обычному пользователю  GRANT SELECT ON forum\_db.topics, forum\_db.posts TO forum\_user;  GRANT INSERT ON forum\_db.posts TO forum\_user;  -- 5. Назначение ролей пользователям  GRANT forum\_admin TO 'admin\_user'@'localhost';  GRANT forum\_moderator TO 'moder\_user'@'%';  GRANT forum\_user TO 'regular\_user'@'%';  -- 6. Активация ролей (для MySQL 8.0+)  SET DEFAULT ROLE forum\_user TO 'regular\_user'@'%'; |

## 2.7. Аутентификация пользователей в базе данных

|  |
| --- |
| -- Регистрация  CALL register\_user('new\_user', 'secure\_password123', 'user@example.com');  -- Аутентификация  CALL authenticate\_user('new\_user', 'secure\_password123', @user\_id);  SELECT IF(@user\_id IS NULL, 'Ошибка входа', CONCAT('Добро пожаловать, ID: ', @user\_id)); |

## 2.8. Импорт и экспорт базы данных

Экспорт

|  |
| --- |
| mysqldump -u пользователь -p forum\_db > forum\_backup.sql |

Импорт

|  |
| --- |
| mysql -u пользователь -p forum\_db < forum\_backup.sql |

## 2.9. Разработка стратегии резервного копирования базы данных

Типы резервных копий

|  |
| --- |
| -- Полный бэкап (ежедневно)  mysqldump --all-databases > full\_backup\_$(date +%F).sql  -- Инкрементный (ежечасно, только изменения)  mysqlbinlog /var/log/mysql/mysql-bin.12345 > incr\_backup.sql |

Автоматизация

|  |
| --- |
| # Ежедневно в 2:00  0 2 \* \* \* mysqldump -u backup\_user -p пароль forum\_db > /backups/daily/forum\_$(date +\%F).sql  # Еженедельно полный бэкап  0 3 \* \* 0 mysqldump --all-databases > /backups/weekly/full\_$(date +\%F).sql |

Проверка бэкапов

|  |
| --- |
| # Тестовое восстановление  mysql -u test\_user -p test\_db < backup\_file.sql  # Проверка целостности  mysqlcheck -u admin -p --all-databases |

Мониторинг

|  |
| --- |
| -- Проверка последнего бэкапа  SELECT MAX(backup\_time) FROM backup\_logs;  -- Логирование ошибок  INSERT INTO backup\_logs (backup\_type, status, size\_mb)  VALUES ('daily', 'success', 250); |

## 2.10.1. Тестирование базы данных

**Ключевые аспекты:**

1. Проверка **граничных значений** (пустые/некорректные данные)
2. Контроль **времени ответа** (запросы >1s - повод для оптимизации)
3. **Восстановление** после сбоев (удаление тестовых данных)
4. **Логирование** результатов тестов

**Инструменты:**

**mysqlslap** - нагрузочное тестирование

**pt-query-digest** - анализ медленных запросов

**DBUnit** - автоматизация тестов (Java)

## 2.10.2. Оптимизация базы данных

**Ключевые методы:**

1. **EXPLAIN** - анализ выполнения запросов
2. **Оптимизация JOIN** - правильный порядок таблиц
3. **Денормализация** - для часто читаемых данных
4. **Настройка СУБД** - буферы, кэши, пулы соединений

**Инструменты:**

**pt-index-usage** - анализ использования индексов

**mysqltuner.pl** - настройка параметров MySQL

**Percona Toolkit** - набор инструментов для оптимизации

# 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).