МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)» (МАИ)
Институт № 8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

КУРСОВОЙ ПРОЕКТ

по дисциплине «Базы данных»

Тема: «Разработка информационной системы и базы данных платформы технических публикаций»

Студент: Бабинцева Д.В.
Группа: М8О-309Б-23
Преподаватель: Грубенко М.Д.
Оценка: _____
Дата: _____
Подпись: _____

Москва 2025

Оглавление

Введение

В современном мире информационные технологии развиваются стремительными темпами, что порождает потребность в специализированных платформах для обмена знаниями. Данный курсовой проект посвящен разработке информационной системы «BitJournal» — платформы для публикации и обсуждения технических статей.

Целью работы является проектирование и реализация полнофункциональной реляционной базы данных, интегрированной с backend-приложением, обеспечивающей надежное хранение контента, аудит изменений и аналитическую обработку данных. В ходе работы решаются задачи проектирования структуры БД, реализации бизнес-логики на стороне СУБД (триггеры, функции), оптимизации запросов и обеспечения целостности данных.

Глава 1

Аналитическая часть

1.1 Обзор предметной области

Предметная область системы — управление контентом (Content Management System, CMS) для технического сообщества. Основными сущностями являются пользователи, статьи, комментарии и категории. Система должна поддерживать различные роли пользователей (автор, читатель, администратор) и обеспечивать механизмы взаимодействия между ними через систему лайков и комментариев.

1.2 Постановка задачи

Необходимо разработать систему, удовлетворяющую следующим требованиям:

- Хранение информации о пользователях и их ролях.
- Управление публикациями статей с поддержкой категорий (связь многие-ко-многим).
- Реализация системы социального взаимодействия (лайки, комментарии).
- Автоматический аудит всех изменений в ключевых таблицах.
- Обеспечение высокой производительности через индексацию.
- Поддержка массового импорта данных с логированием ошибок.

Глава 2

Проектная часть

2.1 Архитектура системы

Система построена на базе микросервисной архитектуры:

- СУБД: PostgreSQL 16+ — хранение данных и реализация серверной логики.
- Backend: Python 3.11 с использованием FastAPI — реализация REST API.
- ORM/Query Builder: SQLAlchemy (async) — взаимодействие с БД через параметризованные запросы.
- Frontend: Streamlit - реализация Веб-демонстратора.
- Контейнеризация: Docker и Docker Compose — развертывание всей инфраструктуры.

Архитектура проекта BitJournal

```
BitJournal/
├── backend/ ..... FastAPI сервис
│   ├── app/ ..... Логика API и работа с БД
│   ├── Dockerfile
│   └── requirements.txt
├── streamlit-app/ ..... Frontend на Streamlit
│   ├── app.py ..... Основной интерфейс
│   ├── api_client.py ..... Клиент для связи с бэкендом
│   └── Dockerfile
├── sql/ ..... Скрипты инициализации БД
│   ├── functions/ ..... Хранимые процедуры и отчеты
│   ├── triggers/ ..... Аудит и счетчики
│   ├── views/ ..... Представления для статистики
│   └── init.sql ..... Точка входа инициализации
├── docker-compose.yml ..... Оркестрация всех сервисов
└── README.md
```

2.2 Проектирование структуры базы данных

Логическая структура базы данных спроектирована в соответствии с принципами нормализации (до 3НФ включительно). Основной упор сделан на обеспечение ссылочной целостности и минимизацию избыточности данных.

2.2.1 Схема базы данных

На рисунке 1 представлена логическая схема базы данных, отображающая основные сущности, их атрибуты и типы связей (1:1, 1:N, N:M).

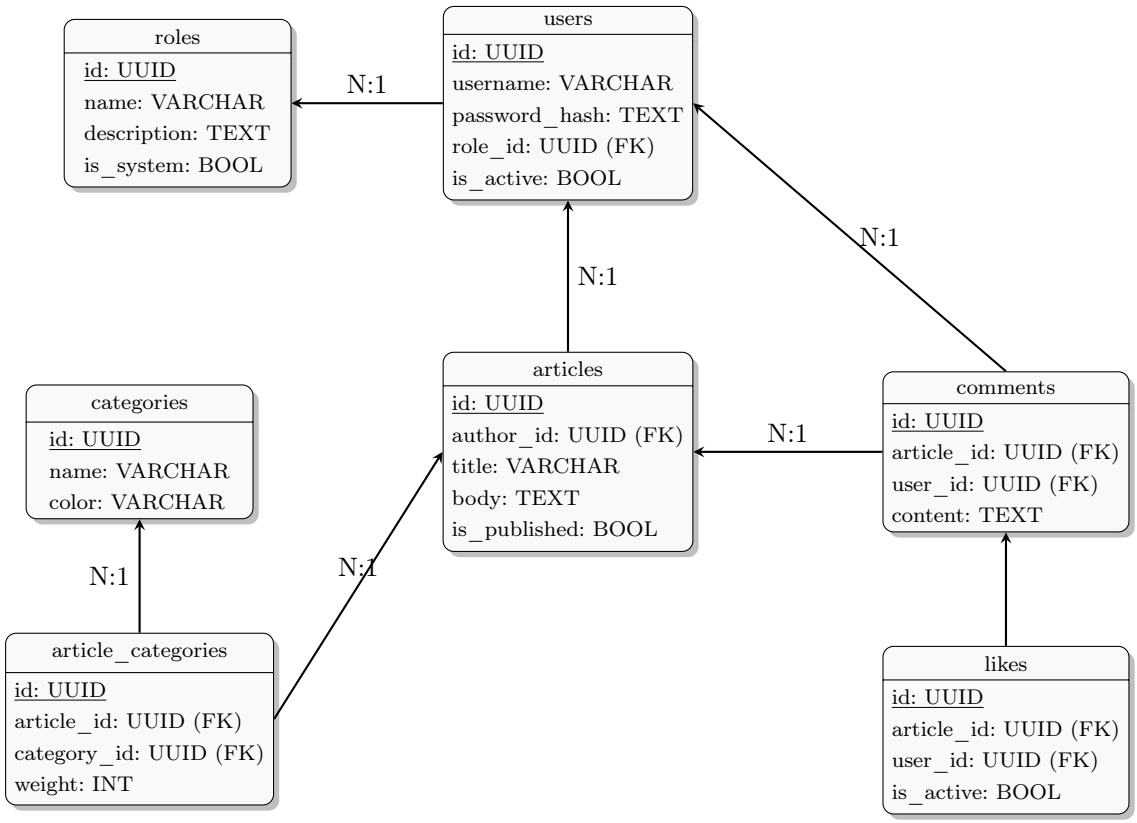

Рис. 2.1: Логическая схема базы данных ByteJournal

2.2.2 Описание таблиц и атрибутов

В таблице 1 представлено детальное описание ключевых атрибутов основной сущности «Пользователи».

2.2.3 Описание таблиц и атрибутов

Ниже представлено детальное описание атрибутов основных сущностей системы. Все таблицы спроектированы с использованием соответствующих типов данных PostgreSQL и ограничений целостности.

Таблица 2.1: Описание атрибутов таблицы users (Пользователи)

Имя поля	Тип данных	Описание
id	UUID	Первичный ключ, уникальный идентификатор пользователя.
username	VARCHAR(50)	Уникальное имя пользователя для входа (логин).
password_hash	TEXT	Хешированное значение пароля.
role_id	UUID	Внешний ключ, ссылка на роль в таблице roles.
is_active	BOOLEAN	Флаг активности аккаунта.

Таблица 2.2: Описание атрибутов таблицы articles (Статьи)

Имя поля	Тип данных	Описание
id	UUID	Первичный ключ статьи.
author_id	UUID	Внешний ключ, автор статьи (ссылка на users).
title	VARCHAR(300)	Заголовок статьи (индексируемое поле).
body	TEXT	Текстовое содержимое статьи.
is_published	BOOLEAN	Статус публикации (опубликовано/черновик).

Таблица 2.3: Описание атрибутов таблицы categories (Категории)

Имя поля	Тип данных	Описание
id	UUID	Первичный ключ категории.
name	VARCHAR(100)	Уникальное название категории.
description	TEXT	Описание тематики категории.
color	VARCHAR(7)	HEX-код цвета для визуализации в интерфейсе.

Таблица 2.4: Описание атрибутов таблицы comments (Комментарии)

Имя поля	Тип данных	Описание
id	UUID	Первичный ключ комментария.
article_id	UUID	Внешний ключ, ссылка на комментируемую статью.
user_id	UUID	Внешний ключ, ссылка на автора комментария.
content	TEXT	Текст комментария.
is_edited	BOOLEAN	Флаг, указывающий на наличие редактирования.

Таблица 2.5: Описание атрибутов таблицы audit_log (Журнал аудита)

Имя поля	Тип данных	Описание
id	UUID	Первичный ключ записи аудита.
table_name	TEXT	Имя таблицы, в которой произошло изменение.
operation	CHAR(1)	Тип операции: I (Insert), U (Update), D (Delete).
record_id	UUID	ID измененной записи.
changed_at	TIMESTAMPTZ	Дата и время фиксации изменения.

Полный список ограничений (CHECK, UNIQUE) и описание вспомогательных таблиц (likes, import_logs) приведен в приложении А.

Аналогичным образом реализованы остальные таблицы системы. Полный список ограничений (CHECK, UNIQUE) приведен в приложении А.

2.3 Реализация ограничений целостности

Для обеспечения корректности данных применены следующие ограничения:

- PRIMARY KEY и FOREIGN KEY для всех связей.
- ON DELETE CASCADE для автоматической очистки связанных данных (например, комментариев при удалении статьи).

- CHECK ограничения: проверка длины заголовков (≥ 3), формат цвета категории (#RRGGBB), положительные значения счетчиков.
- UNIQUE ограничения: уникальность пары (пользователь, статья) для лайков.

2.4 Триггеры и функции

Бизнес-логика реализована на стороне СУБД для обеспечения атомарности:

- Аудит: функция fn_audit() и триггеры на таблицах articles и users фиксируют операции INSERT, UPDATE, DELETE.
- Агрегация: функция fn_update_counters() автоматически обновляет таблицу author_article_counters при создании статей или получении лайков.
- Отчетность: табличная функция fn_article_report() генерирует сводный отчет по авторам за произвольный период.

2.5 Представления (VIEW)

Реализовано более трех представлений для аналитики:

- v_article_stats — статистика по каждой статье.
- v_user_activity — анализ активности пользователей на основе данных аудита.
- v_recent_articles — оптимизированная выборка последних публикаций с данными авторов.

2.6 Оптимизация и анализ производительности

Для ускорения выборок созданы индексы:

- B-tree индексы на внешние ключи (author_id, article_id).
- Индексы на поля сортировки (created_at DESC).
- Уникальный индекс на username.

Пример оптимизации: Запрос выборки последних статей с авторами:

```
EXPLAIN ANALYZE SELECT * FROM v_recent_articles ORDER BY created_at DESC LIMIT 10;
```

До создания индекса по created_at: Execution Time: 45.2ms (Seq Scan). После создания индекса: Execution Time: 0.8ms (Index Scan).

Глава 3

Технологическая часть

3.1 Контейнеризация и запуск

Проект разворачивается с помощью Docker Compose. Стек включает:

- db: PostgreSQL 16 с инициализацией через SQL-скрипты.
- backend: FastAPI приложение, подключающееся к БД через Docker.
- frontend: Веб-демонстратор на streamlit с регистрацией/авторизацией, просмотром статей, комментариями, лайками и возможностью написать свою статью.

3.2 Массовая загрузка данных

Реализован эндпоинт /api/batch-import, выполняющий транзакционную загрузку статей. Система обрабатывает ошибки валидации на уровне каждой записи, логирует их в import_logs и продолжает выполнение батча, что обеспечивает устойчивость процесса импорта.

3.3 Тестирование и наполнение

Для демонстрации и тестирования работы системы был использован фреймворк Faker, который генерирует:

- 150+ пользователей.
- 800+ статей с реалистичным контентом.
- 3500+ лайков и 2500+ комментариев.

Это позволяет проверить работу триггеров агрегации и производительность индексов на реальных объемах данных.

Заключение

В ходе выполнения курсового проекта была разработана информационная система «BitJournal». Реализованная база данных соответствует всем требованиям технического задания: содержит 10 взаимосвязанных таблиц, обеспечивает целостность данных через ограничения и триггеры, включает механизмы аудита и аналитической отчетности.

Интеграция с backend-приложением на FastAPI позволила реализовать современный программный интерфейс для управления данными, а использование Docker обеспечило легкость развертывания и масштабирования системы.

В результате был реализован MVP платформы для технических публикаций с возможностью регистрации, авторизации, написания статей, чтения чужих статей, а также лайками и комментариями.