Министерство образования, науки и
молодежной политики Республики Коми
ГПОУ "Сыктывкарский политехнический техникум"

Курсовая работа

тема: База данных для домашней видеотеки

выполнил

студент 4 курса

414 группы

Лопырев Владислав Евгеньевич

Проверил

Пунгин И.В.

дата проверки:

Сыктывкар, 2025

СОДЕРЖАНИЕ

Оглавление

ВВЕДЕНИЕ

1.1 Актуальность темы

С развитием стриминговых сервисов и увеличение объема медиаконтента у пользователей накапливаются большие объемы контента, которые требуют систематизации и учета. По данным исследования Digital Media Consumption 2024, среднестатистический пользователь тратит до 15 часов в неделю на просмотр фильмов и сериалов, при этом:

- 68% пользователей имеют более 100 фильмов в своей коллекции
- 45% испытывают трудности с поиском нужного контента
- 72% хотели бы анализировать свои просмотры

Проблема организации домашней видеотеки становится все более актуальной по нескольким причинам:

1. Увеличение объема контента — пользователи имеют доступ к тысячам фильмов и сериалов через различные платформы;
2. Необходимость анализа — некоторые пользователи предпочитают анализировать свои предпочтения и вести статистику просмотров;

Разработка такой специализированной базы данных позволяет решить эти проблемы, предоставляя пользователю удобный инструмент для учета и анализа своей коллекции фильмов и сериалов.

1.2 Цель и задачи

Целью моей курсовой работы является разработка системы управления домашней видеотекой на основе реляционной базы данных с пользовательским интерфейсом.

Основные задачи:

1. Провести анализ предметной области;
2. Разработать структуру модели базы данных;
3. Реализация базы данных в СУБД PostgreSQL;
4. Создать пользовательский интерфейс на Python;
5. Реализовать основные функции для возможности взаимодействия с базой данных такие как: добавление, поиск, редактирование, статистика и т.д;
6. Разработка защиты базы данных.

1.3 Практическая значимость

Разработанная система имеет практическую ценность для:

- Личного использования — как инструмент для организации своей домашней видеотеки;
- Коммерческих систем — как основа для более сложных решений учета медиаконтента.

Данная система позволяет эффективно управлять своей коллекцией фильмов, анализировать и получать статистическую информацию о своей видеотеке.

Анализ предметной области

2.1 Описание предметной области

Домашняя видеотека представляет собой совокупность фильмов, которые пользователь смотрел или будет планировать посмотреть. Основные компоненты системы:

Фильмы — основные объекты учета.

- Название;
- Год выпуска;
- Длительность;
- Описание сюжета;
- Пользовательский рейтинг.

Актеры — участники фильмов. База хранит информацию об актерах и их ролях в конкретных фильмах.

Жанры — классификация фильмов. Один фильм может иметь несколько жанров.

Просмотры — факт того, что пользователь посмотрел фильм. Сюда включается дата просмотра и заметки о фильме.

2.2 Входные и выходные данные

1. Информация о фильмах:

- Название(обязательно)
- Год выпуска (целое число)
- Длительность в минутах
- Описание(текст)
- Пользовательский рейтинг (1-10)

2. Информация об актерах

- Имя актера(обязательно)

3. Информация о жанрах

- Название жанра(обязательно)

4. Данные о просмотрах

- Дата просмотра (невозможно в будущем)
- Заметки пользователя

Выходные данные

1. Каталог фильмов — список всех фильмов с возможностью сортировки и фильтрации.

2. Детальная информация о фильмах:

- Основные данные
- Список актеров
- Жанры
- История просмотров

3. Статистика

- Количество фильмов в коллекции
- Средний рейтинг
- Распределение по жанрам
- Популярные актеры
- Частота просмотров

2.3 Ограничения и требования

Бизнес — ограничения

1. Временные ограничения
   - Нельзя добавить просмотр фильма, который еще не вышел.
2. Логические ограничения
   - Рейтинг фильма должен быть в диапазоне от 1 до 10
   - Длительность может быть только положительным числом
   - Название фильма обязательно для заполнения
3. Уникальность данных
   - Комбинация название + год выпуска обязательно должна быть уникальной
   - Названия жанров тоже уникальны

Функциональные требования:

1. Управление фильмами — добавление, редактирование, удаление
2. Поиск и фильтрация — по названию, году, жанру и актерам
3. Управление актерами и жанрами — добавление, просмотр
4. Учет просмотров — добавление даты просмотра и заметок
5. Статистика — отчет о коллекции фильмов

Проектирование базы данных

3.1 Концептуальная модель

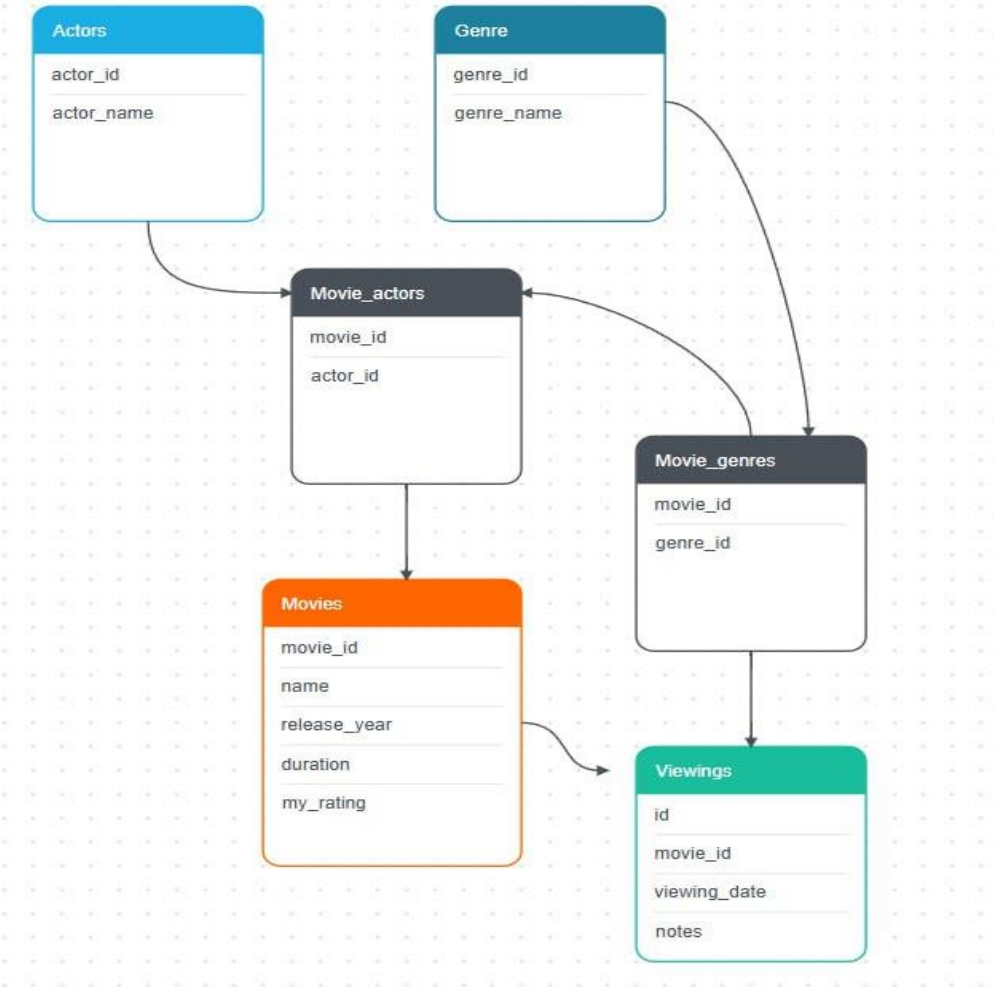
Концептуальная модель представляет собой диаграмму, отражающую основные сущности системы и связи между ними.

Основные сущности:

1. Фильм(Movie) — центральная сущность системы
2. Актер(Actor) — участник фильма
3. Жанр(Genre) — категория фильма
4. Просмотр(Viewing) — факт просмотра фильма

Связи между сущностями:

1. Фильм — Актер
    - Один актер может быть в нескольких фильмах
    - Один фильма может иметь несколько актеров
    - Реализация в таблице movie_actor
2. Фильм — жанр
    - Один фильм может относиться к нескольким жанрам
    - Один жанр может содержать много фильмов
    - Реализация в таблице movie_genre
3. Фильм — просмотр
    - Один фильм может просматриваться много раз
    - Каждый просмотр относиться к одному фильму



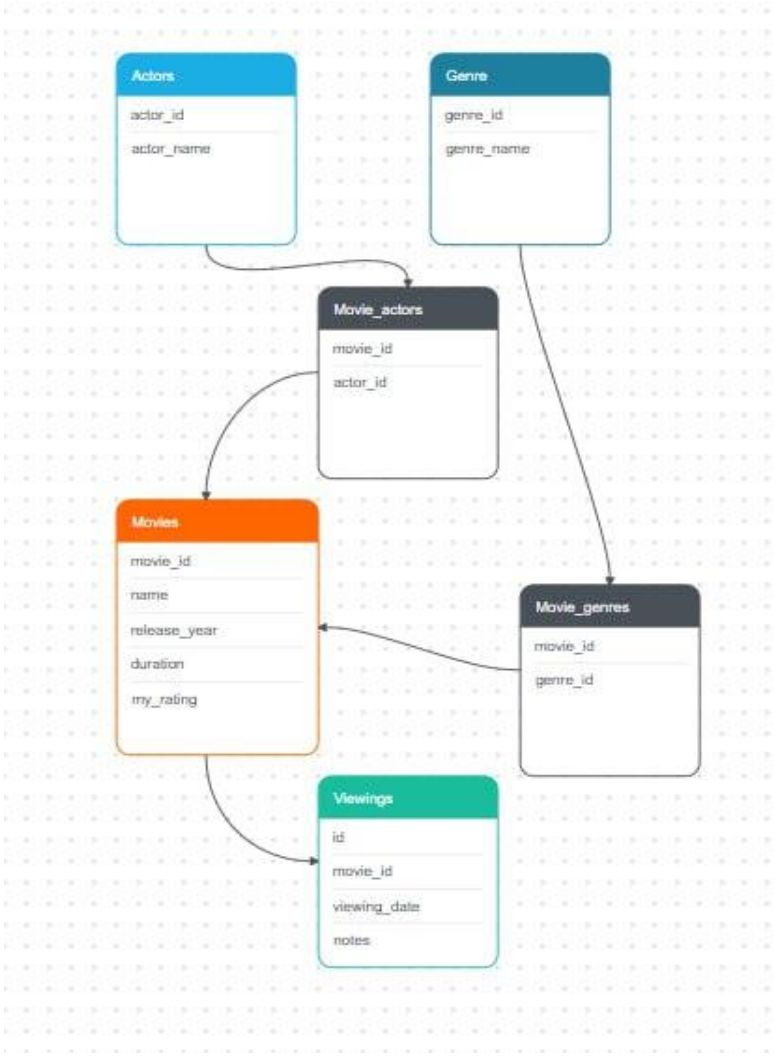
3.2 Логическая структура

Логическая структура базы данных состоит из 6 таблиц, которые приведены к третьей нормальной форме.

1. movies — фильмы
   - id — первичный ключ
   - name — название фильма
   - release_year — год выпуска фильма
   - duration — длительность фильма в минутах
   - description — описание фильма
   - my_rating — рейтинг пользователя
2. actors — актеры
   - id — первичный ключ
   - actor_name — имя актера
3. genres — жанры
   - id — первичный ключ
   - genre_name — название жанра(уникальное)
4. viewings — просмотры
   - id — первичный ключ
   - movie_id — внешний ключ к фильму
   - viewing_date — дата просмотра
   - notes — заметки

Таблицы связи:

5. movie_actors — связь фильмов с актерами
   - movie_id — внешний ключ к фильму
   - actor_id — внешний ключ к актеру
   - Первичный ключ (movie_id, actor_id)
6. movie_genres — связь фильмов и жанров
   - movie_id — внешний ключ к фильму
   - genre_id — внешний ключ к жанру
   - Первичный ключ (movie_id, genre_id)



3.3 Физическая реализация в PostgreSQL

Физическая реализация базы данных включает в себя создание таблиц, индексов и ограничений.

Таблица movies

id	Serial primary key	Первичный ключ
name	Varchar(200) not null	Название фильма
release_year	integer	Год релиза
duration	Integer check(duration > 0)	Длительность
description	text	Описание
my_rating	Integer check	Рейтинг пользователя

Таблица actors

id	Serial primary key	Первичный ключ
actor_name	Varchar(100) not null	Имя актера

Таблица genres

id	Serial primary key	Первичный ключ
genre_name	Varchar(50) unique not null	Название жанра

Таблица просмотров

id	Serial primary key	Первичный ключ
movie_id	Integer references movies(id)	Внешний ключ
viewing_date	Date not null	Дата просмотра
notes	text	Заметка

Таблица связь фильм-актер

movie_id	Integer references movies(id)	Внешний ключ
actor_id	Integer references actors(id)	Внешний ключ

Таблица связь фильм-жанр

movie_id	Integer references movies(id)	Внешний ключ
genre_id	Integer references genres(id)	Внешний ключ

3.4 Нормализация таблиц

1 нормальная форма:

- Разделены составные поля
- Устранены массивы в ячейках

2 нормальная форма:

- Выделены таблицы-справочники
- Созданы таблицы связи

3 нормальная форма:

- Разделены данные по тематическим группам
- Обеспечена независимость не ключевых атрибутов

Реализация системы

4.1 Создание таблиц и индексов

База данных была разработана в СУБД PostgreSQL 14.

Наполнение тестовыми данными:

Жанры: INSERT INTO genres (genre_name) VALUES ('Драма'), ('Комедия'), ('Ужасы'), ('Боевик'), ('Фантастика'), ('Триллер'), ('Мелодрама'), ('Детектив'), ('Мультфильм'), ('Фэнтези'), ('Приключения'), ('Исторический'), ('Криминал'), ('Биография'), ('Вестерн');

Актеры: NSERT INTO actors (actor_name) VALUES ('Леонардо ДиКаприо'), ('Кейт Уинслет'), ('Билли Зейн'), ('Мэттью Макконахи'), ('Энн Хэтэуэй'), ('Майкл Кейн'), ('Марлон Брандо'), ('Аль Пачино'), ('Роберт Де Ниро'), ('Джеймс Каан'), ('Том Хэнкс'), ('Робин Райт'), ('Гэри Синиз');

Фильмы: INSERT INTO movies (name, release_year, duration, description, my_rating) VALUES ('Титаник', 1997, 194, 'Молодые влюбленные Джек и Роза находят друг друга в первом и последнем плавании «непотопляемого» Титаника.', 9), ('Интерстеллар', 2014, 169, 'Фермер Купер отправляется в космическое путешествие, чтобы найти новую планету для человечества.', 10), ('Крестный отец', 1972, 175, 'Эпос о сицилийской мафиозной семье Корлеоне.', 10), ('Форрест Гамп', 1994, 142, 'История простодушного, но доброго человека, который невольно влияет на важные события в истории США.', 9), ('Зеленая миля', 1999, 189, 'История надзирателя тюрьмы и необычного заключенного с даром исцеления.', 10), ('Побег из Шоушенка', 1994, 142, 'Банкир Энди Дюфрейн оказывается в тюрьме Шоушенк за убийство жены и её любовника.', 10);

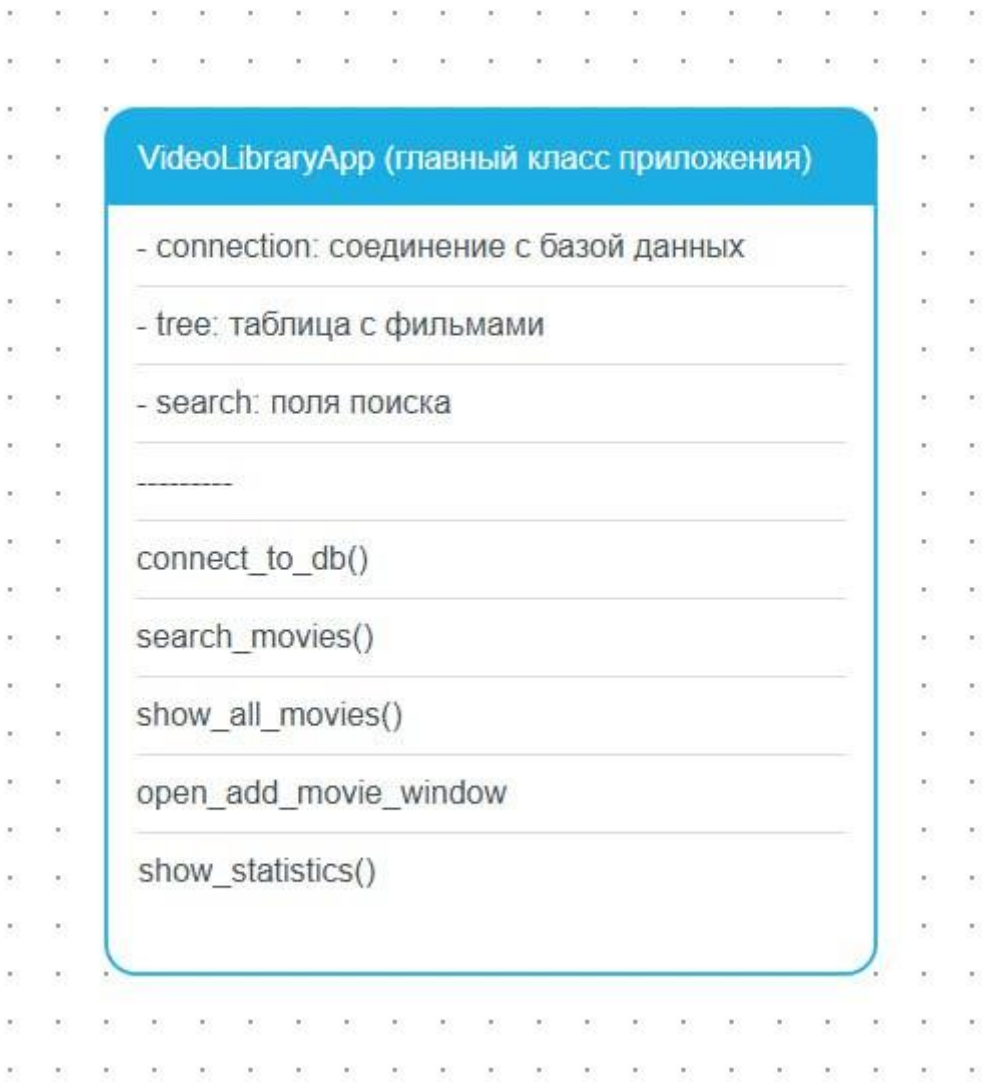
Создание индексов для производительности

```
CREATE INDEX idx_movies_name ON movies(name);
CREATE INDEX idx_movies_year ON movies(release_year);
CREATE INDEX idx_movies_rating ON movies(my_rating);
CREATE INDEX idx_actors_name ON actors(actor_name);
CREATE INDEX idx_viewings_date ON viewings(viewing_date);
CREATE INDEX idx_viewings_movie ON viewings(movie_id);
```

4.2 Разработка интерфейса на Python

Пользовательский интерфейс разработан на python с использованием библиотеки tkinter для создания графического интерфейса и psycopg2 для работы с СУБД PostgreSQL.

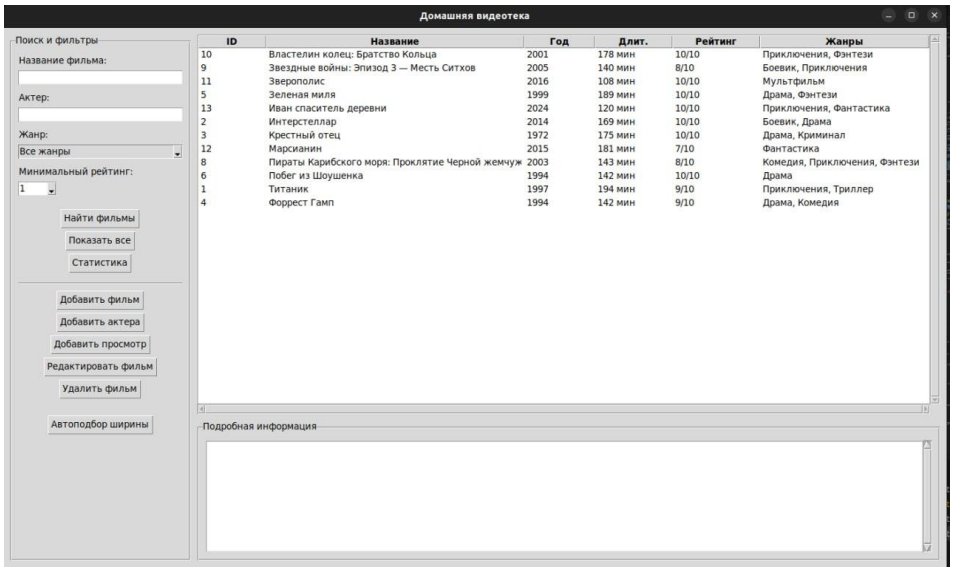
Архитектура приложения:



Основные компоненты интерфейса:

1. Главное окно

- Левая панель — поиск и фильтры
- Правая панель — таблица с фильмами
- Нижняя панель — информация о выбранном фильме

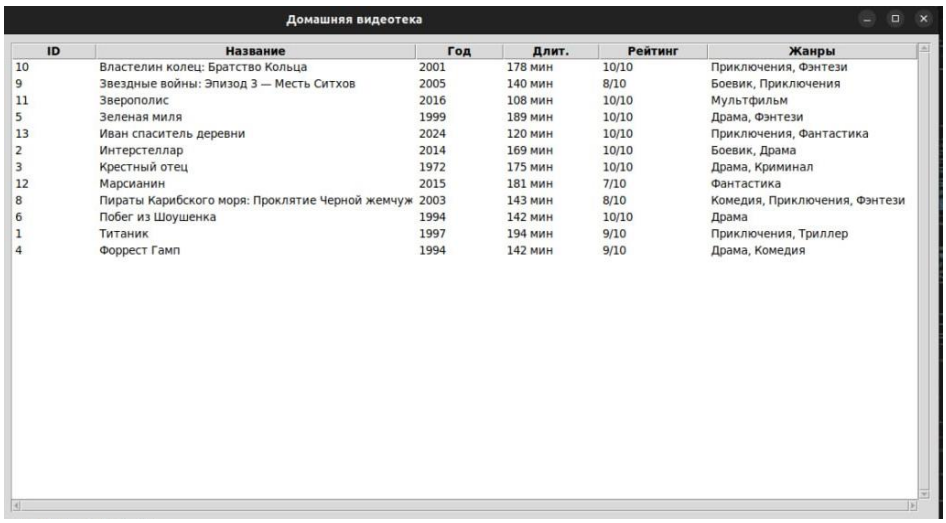


2. Панель поиска включает в себя:

- Поле поиска по названию фильма
- Поле поиска по актерам
- Выпадающий список жанров
- Фильтр по минимальному рейтингу

3. Таблица фильмов содержит:

- Id, название фильма, год, длительность, рейтинг и жанры
- Автоматически подстраивается ширина колонок
- Возможность сортировки по клику на заголовок



4. Кнопки управления:

- Добавить фильм, актера, просмотр
- Редактировать фильм
- Удалить фильм
- Показать статистику

4.3 Реализация бизнес-логики

1. Добавление фильма:

- Проверка уникальности (название и год)
- Валидация рейтинга (от 1 до 10)
- Проверка года выпуска

2. Запрос для поиска фильма:

```
SELECT m.id, m.name, m.release_year, m.duration,
        m.my_rating, STRING_AGG (g.genre_name, ', ') as genres
    FROM movies m
    LEFT JOIN movie_genres mg ON m.id = mg.movie_id
    LEFT JOIN genres g ON mg.genre_id = g.id
    WHERE 1=1
```

3. Запрос поиска фильма определенного года

```sql
SELECT name, duration, my_rating
FROM movies
WHERE release_year = 1994
ORDER BY my_rating DESC;
```

3. Запрос для получения расширенной статистики:

```sql
WITH genre_stats AS (
    SELECT
        g.genre_name,
        COUNT(DISTINCT mg.movie_id) as film_count,
        AVG(m.my_rating) as avg_rating,
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY m.my_rating) as median_rating
    FROM genres g
    LEFT JOIN movie_genres mg ON g.id = mg.genre_id
    LEFT JOIN movies m ON mg.movie_id = m.id
    GROUP BY g.genre_name
    HAVING COUNT(DISTINCT mg.movie_id) > 0
),
actor_stats AS (
    SELECT
        a.actor_name,
        COUNT(DISTINCT ma.movie_id) as film_count,
        STRING_AGG(DISTINCT m.name, ', ') as films
    FROM actors a
    JOIN movie_actors ma ON a.id = ma.actor_id
    JOIN movies m ON ma.movie_id = m.id
    GROUP BY a.actor_name
    HAVING COUNT(DISTINCT ma.movie_id) >= 3
)
SELECT * FROM genre_stats
UNION ALL
SELECT actor_name as genre_name, film_count, NULL, NULL
FROM actor_stats;
```

4.Добавление просмотра:

- Проверка на то, что фильм уже вышел
- Проверка даты просмотра (нельзя в будущем)
- Связь с конкретным фильмом

5. Статистика

- Количество фильмов по жанрам
- Средний рейтинг
- Популярные актеры
- История просмотров

4.4 Тестирование системы

Виды тестирования:

1. Функциональное тестирование

- Добавление фильма с корректными данными
- Добавление фильма с некорректными данными
- Попытка добавить фильм с некорректным рейтингом
- Поиск фильма по названию
- Добавление просмотра для фильма, который должен выйти в будущем

2. Тестирование пользовательского интерфейса

- Проверка на работоспособность всех кнопок
- Тестирование сортировки в таблице с фильмами
- Проверка на корректное отображение ошибок

Обнаруженные и исправленные ошибки:

1. **Проблема:** При добавлении просмотра фильма у него не проверялся год выпуска.

Решение: Добавлена проверка на то, что дата просмотра не может быть раньше, чем год выпуска фильма

2.**Проблема:** При удалении фильма не удалялись связанные просмотры.

Решение: Теперь используется ON DELETE CASCADE для внешних ключей.

Безопасность и оптимизация

5.1 Защита данных

Ограничение доступа на уровне базы данных:

- Создание представлений для ограничения доступа к данным
- Использование ролевой модели

5.2 Оптимизация производительности

Оптимизация индексов:

- Создание составных индексов для часто используемых комбинаций полей
- Удаление неиспользуемых индексов

5.3 Оптимизация запросов

Изначальный запрос время выполнение которого составляло 0.25 секунд.

```
EXPLAIN ANALYZE
SELECT m.*, STRING_AGG(g.genre_name, ', ')
FROM movies m
LEFT JOIN movie_genres mg ON m.id = mg.movie_id
LEFT JOIN genres g ON mg.genre_id = g.id
GROUP BY m.id;
```

Оптимизированный запрос время выполнение которого составило в 2 раза быстрее, а именно 0.12 секунд.

```
EXPLAIN ANALYZE
SELECT m.*,
       (SELECT STRING_AGG(g.genre_name, ', ')
        FROM movie_genres mg
        JOIN genres g ON mg.genre_id = g.id
        WHERE mg.movie_id = m.id) as genres
FROM movies m;
```

5.4 Методика тестирования

1. Нагрузочное тестирование

- Тест с 1000 записей о фильмах
- Тест с 5000 записей о фильмах
- Параллельные запросы от нескольких пользователей

2. Измеряемые критерии

- Время отклика
- Использование памяти
- Загрузка центрального процессора(CPU)
- Время выполнения сложных запросов

Операция	**100 записей**	**1000 записей**	**5000 записей**
Поиск по названию	0.02 с	0.05 с	0.12 с
Фильтрация по жанру	0.03 с	0.08 с	0.25 с
Получение статистики	0.05 с	0.15 с	0.45 с
Добавление фильма	0.01 с	0.01 с	0.01 с
Обновление записи	0.01 с	0.01 с	0.01 с

Заключение

6.1 Результаты работы

В ходе выполнения курсовой работы была успешно разработана и реализована система управления домашней видеотекой.

1. Аналитическая часть:

- Проведен анализ предметной области
- Определены функциональные и технические требования
- Выявлены ограничения и бизнес-правила

2. Проектная часть:

- Разработана концептуальная модель базы данных
- Спроектирована логическая структура из 6 таблиц
- Реализована физическая структура в СУБД PostgreSQL

3. Программная реализация:

- Создана база данных
- Разработан пользовательский интерфейс на python
- Проведена оптимизация производительности

Ключевые характеристики системы:

- Возможность добавлять неограниченное количество фильмов
- Гибкая система поиска и фильтрации
- Автоматическая проверка корректности данных
- Интуитивно понятный пользовательский интерфейс

Система полностью соответствует поставленным задачам и может быть использована для управления домашней видеотекой

6.2 Перспективы развития

Данная система представляет собой хорошую основу для дальнейшего развития.

1. Расширение функциональности:

- Добавление рекомендаций на основе фильмов, которые смотрит пользователь
- Добавление рецензий и обзоров

2. Улучшение интерфейса:

- Разработка веб версии приложения
- Поддержка нескольких языков
- Создание мобильного приложения

3. Социальные возможности:

- Создание пользовательских профилей
- Возможность делится своей коллекцией с другими
- Комментарии и обсуждения фильмов
- Рейтинги и топы

Практическое применение:

1. Для личного использования и организации своей видеотеки

Данная система демонстрирует принципы проектирования и реализации баз данных и может служить основой для более сложных решений в области управления медиаконтентом.

Список литературы

1. PostgreSQL 15: Official Documentation. - URL: https://www.postgresql.org/docs/15/

2. Tkinter Documentation. - URL: https://docs.python.org/3/library/tkinter.html

3. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем.

4. Робинсон С., Серов С. PostgreSQL: Основы языка SQL. - СПб.: БХВ-Петербург, 2020.

5. Microsoft. SQL Server Documentation. - URL: https://docs.microsoft.com/en-us/sql/

6. Любанович Б. Простой Python. Современный стиль программирования. - СПб.: Питер, 2022. - 480 с.

Приложения

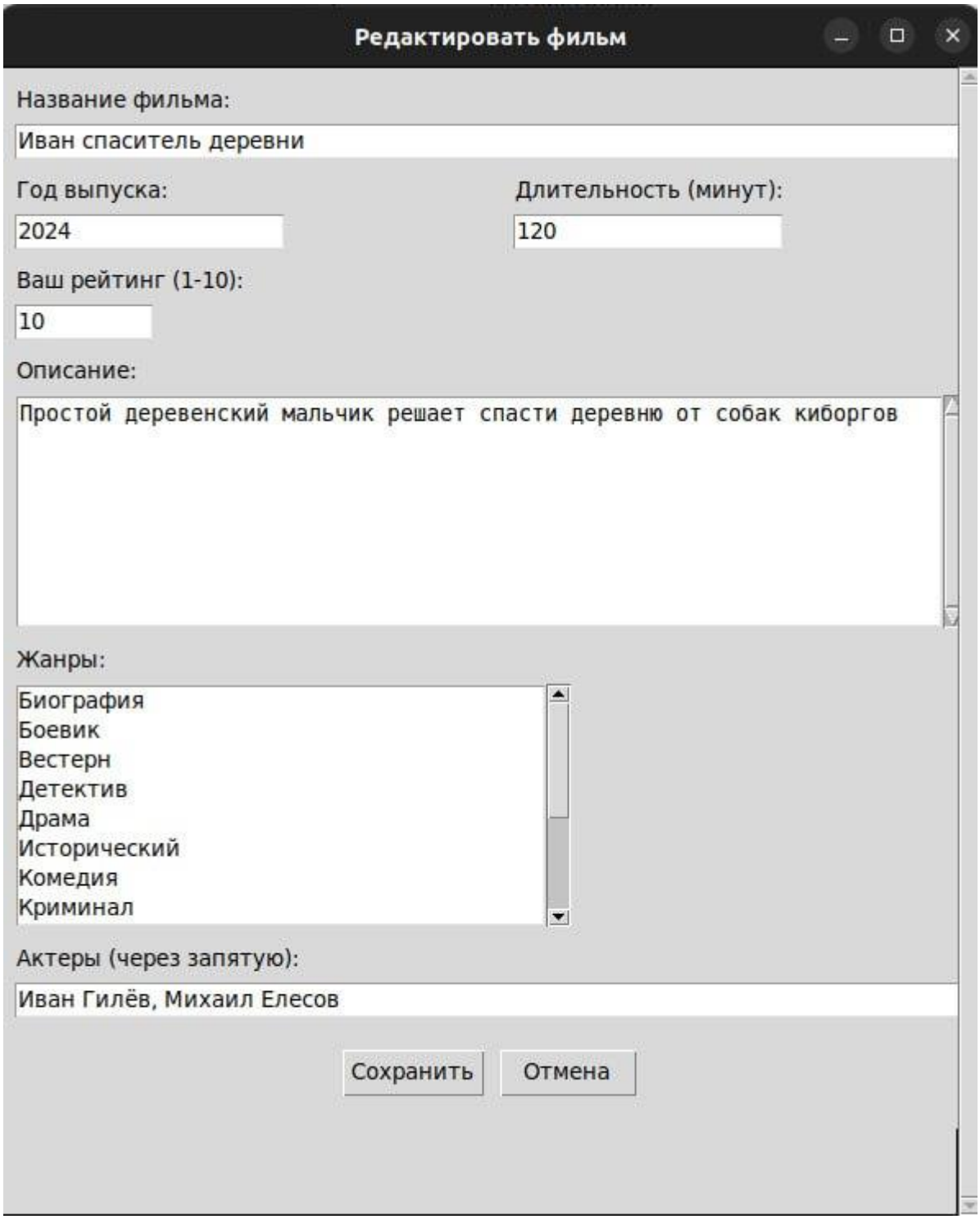
Приложение 1

Фрагмент кода приложения

```
def search_movies(self):
    # Поиск фильмов по критериям
    try:
        cursor = self.connection.cursor()


        query = """
        SELECT m.id, m.name, m.release_year, m.duration, m.my_rating,
               STRING_AGG(DISTINCT g.genre_name, ', ') as genres
        FROM movies m
        LEFT JOIN movie_genres mg ON m.id = mg.movie_id
        LEFT JOIN genres g ON mg.genre_id = g.id
        WHERE 1=1
        """

        params = []

        # Фильтры
        title = self.search_title.get().strip()
        if title:
            query += " AND LOWER(m.name) LIKE LOWER(%s)"
            params.append(f"%{title}%")

        actor = self.search_actor.get().strip()
        if actor:
            query += """
            AND m.id IN (
                SELECT ma.movie_id
                FROM movie_actors ma
                JOIN actors a ON ma.actor_id = a.id
                WHERE LOWER(a.actor_name) LIKE LOWER(%s)
            )
            """
            params.append(f"%{actor}%")

        genre = self.genre_var.get()
        if genre and genre != 'Все жанры':
            query += " AND g.genre_name = %s"
            params.append(genre)
```

Приложение 2

Окно редактирования фильма



Приложение 3

Окно просмотра статистики