

Задание

Задача – информационная поддержка сотрудников справочной службы кинотеатров города. БД должна содержать актуальную информацию по работающим кинотеатрам, репертуаре, расписание сеансов, расценки, наличие свободных мест.

Необходимо предусмотреть:

- хранение сведений о кинотеатрах (название, расположение, вместимость, сколько кинозалов, наличие дополнительных услуг, репертуар на месяц вперед);
- информацию о фильме (название, режиссер, основной актерский состав, год выхода, длительность, жанр, кадры/трейлер к фильму, кинотеатрах, где этот фильм демонстрируется или будет демонстрироваться);
- корректировку информации о прокатных фильмах и работающих кинотеатрах.

Примечание: на разных сеансах в одном кинотеатре могут демонстрироваться разные фильмы, а если в кинотеатре несколько залов, то и на одном; цена билета зависит от сеанса, количества дополнительных услуг кинозала, стоимости проката данной копии фильма.

Содержание

Задание	2
Содержание.....	3
Аннотация	4
Введение.....	5
Обоснование выбора стека технологий для курсовой работы.....	7
Логическое проектирование базы данных	9
Описание предметной области	9
Основные требования к базе данных	10
Основные таблицы.....	10
Нормализация базы данных	14
ER-диаграмма	16
Создание таблиц.....	17
Создание триггеров.....	25
Заполнение таблиц данными	32
Создание индексов.....	42
Создание ролей.....	45
Запросы	49
Создание представлений	59
Заключение	61
Список используемых источников.....	63

Аннотация

В данной курсовой работе рассматривается разработка информационной базы данных (БД) для поддержки сотрудников справочной службы кинотеатров города. Основной целью работы является создание системы, которая будет обеспечивать актуальную и структурированную информацию о работающих кинотеатрах, их репертуаре, расписании сеансов, расценках и наличии свободных мест.

База данных включает в себя несколько ключевых компонентов: сведения о кинотеатрах (таких как название, расположение, категория, вместимость, количество кинозалов и наличие дополнительных услуг), а также детальную информацию о фильмах (название, режиссер, актерский состав, год выхода, длительность, жанр и мультимедийные материалы). Особое внимание уделяется возможности корректировки информации о прокатных фильмах и работающих кинотеатрах, что позволяет поддерживать базу в актуальном состоянии.

Система учитывает сложные взаимосвязи между кинотеатрами и фильмами, позволяя отображать различные сеансы в одном кинотеатре и учитывать особенности нескольких залов. Также предусмотрены механизмы для расчета цен билетов в зависимости от сеанса, количества дополнительных услуг кинозала и стоимости проката фильма.

Работа включает в себя проектирование структуры базы данных, разработку интерфейсов для ввода и редактирования данных, а также реализацию функций для поиска и фильтрации информации. Результатом станет эффективный инструмент для сотрудников справочной службы, способствующий улучшению качества обслуживания клиентов и оптимизации процессов управления репертуаром кинотеатров.

Введение

Современная индустрия кино и развлечений активно развивается, предлагая зрителям широкий выбор фильмов и удобные условия для их просмотра. В условиях высокой конкуренции между кинотеатрами и стремления к повышению качества обслуживания клиентов, становится необходимым создание эффективной информационной системы, которая обеспечивала бы сотрудников справочной службы актуальными данными о работе кинотеатров, их репертуаре и расписании сеансов. Разработка базы данных "Кинотеатр" призвана решить эти задачи, предоставляя структурированную и доступную информацию о всех аспектах работы кинотеатров.

Объект исследования

Объектом исследования данной курсовой работы является система управления базами данных (СУБД), предназначенная для хранения и обработки информации о кинотеатрах, фильмах, сеансах и ценах на билеты.

Предмет исследования

Предметом исследования является проектирование и реализация базы данных "Кинотеатр", включая структуру хранения данных, механизмы их обработки и интерфейсы взаимодействия с пользователями.

Актуальность

Актуальность работы обусловлена необходимостью создания эффективных инструментов для поддержки сотрудников справочной службы кинотеатров, что позволит оптимизировать процессы управления репертуаром и повысить качество обслуживания клиентов. В условиях быстроменяющегося рынка развлечений, наличие актуальной информации о кинотеатрах и фильмах становится ключевым фактором для привлечения зрителей и повышения их удовлетворенности.

Цель работы

Целью данной курсовой работы является разработка информационной базы данных "Кинотеатр", которая будет обеспечивать сотрудников справочной

службы необходимыми данными о кинотеатрах, репертуаре, расписании сеансов и ценах на билеты.

Задачи работы

Для достижения поставленной цели необходимо решить следующие задачи:

- Проектирование структуры базы данных, включая таблицы для хранения информации о кинотеатрах, фильмах, сеансах и ценах на билеты.
- Разработка интерфейсов для ввода, редактирования и удаления данных о кинотеатрах и фильмах.
- Реализация функций для поиска и фильтрации информации по различным критериям (например, по названию фильма, дате сеанса, категории кинотеатра).
- Обеспечение возможности корректировки информации о прокатных фильмах и работающих кинотеатрах.
- Разработка механизмов для расчета цен на билеты с учетом различных факторов (сеанс, категория кинотеатра, стоимость проката фильма).
- Тестирование разработанной системы на предмет ее функциональности и удобства использования для сотрудников справочной службы.

Обоснование выбора стека технологий для курсовой работы

При выборе стека технологий для разработки базы данных "Кинотеатр" было принято решение использовать систему управления базами данных (СУБД) PostgreSQL. Данный выбор обоснован рядом факторов, касающихся функциональности, производительности и удобства работы с этой СУБД.

1. Функциональные возможности

PostgreSQL — это мощная объектно-реляционная СУБД, которая поддерживает множество современных функций, таких как:

- **Поддержка сложных запросов:** PostgreSQL позволяет выполнять сложные SQL-запросы, включая подзапросы, объединения и оконные функции, что делает его подходящим для обработки больших объемов данных и сложной логики бизнес-процессов.
- **Расширяемость:** PostgreSQL предоставляет возможность создавать собственные типы данных, функции и операторы, что позволяет адаптировать базу данных под специфические требования проекта.
- **Поддержка транзакций:** СУБД обеспечивает надежность данных через поддержку ACID (атомарность, согласованность, изолированность, долговечность), что критично для обеспечения целостности информации о кинотеатрах и сеансах.

2. Производительность и масштабируемость

PostgreSQL обладает высокой производительностью при работе с большими объемами данных и может эффективно масштабироваться в зависимости от потребностей проекта. Это особенно важно для системы, которая будет обрабатывать информацию о множестве кинотеатров, фильмов и сеансов.

3. Сообщество и поддержка

PostgreSQL имеет активное сообщество разработчиков и пользователей, что обеспечивает доступ к большому количеству ресурсов, документации и готовых решений. Это упрощает процесс разработки и устранения возможных проблем.

4. Кроссплатформенность

PostgreSQL является кроссплатформенным решением, что позволяет разрабатывать и развертывать базу данных на различных операционных системах, таких как Windows, Linux и macOS. Это дает гибкость в выборе среды разработки и развертывания.

5. Интеграция с другими технологиями

PostgreSQL легко интегрируется с различными языками программирования и фреймворками, такими как Python, Java, Ruby on Rails и другими. Это позволяет использовать современные инструменты разработки для создания пользовательских интерфейсов и бизнес-логики приложения.

6. Безопасность

PostgreSQL предлагает множество механизмов безопасности, таких как аутентификация пользователей, шифрование данных и управление правами доступа. Это важно для защиты конфиденциальной информации о клиентах и фильмах.

Выбор PostgreSQL в качестве основной СУБД для создания базы данных "Кинотеатр" обоснован его функциональными возможностями, производительностью, активным сообществом и поддержкой современных стандартов безопасности. Данная СУБД обеспечит надежное хранение и обработку данных, а также гибкость в разработке интерфейсов для взаимодействия с пользователями.

Логическое проектирование базы данных

Описание предметной области

Предметная область базы данных "Кинотеатр" охватывает информационную поддержку сотрудников справочной службы кинотеатров. Основной задачей является хранение и управление актуальной информацией о кинотеатрах, фильмах, расписании сеансов, ценах на билеты и наличии свободных мест. База данных должна обеспечивать доступ к информации как для сотрудников кинотеатров, так и для клиентов, интересующихся репертуаром и услугами.

Функции базы данных

База данных "Кинотеатр" должна выполнять следующие функции:

1. Хранение информации о кинотеатрах:

- Регистрация новых кинотеатров.
- Обновление информации о существующих кинотеатрах (название, расположение, категория, вместимость, количество залов, дополнительные услуги).

2. Управление репертуаром:

- Добавление и редактирование информации о фильмах.
- Связывание фильмов с кинотеатрами, где они демонстрируются.

3. Расписание сеансов:

- Хранение информации о сеансах (время начала, зал, фильм).
- Управление свободными местами на сеансах.

4. Цены на билеты:

- Установка цен на билеты в зависимости от сеанса, количества дополнительных услуг кинозала и проката фильма.
- Изменение цен при необходимости.

5. Отчеты и аналитика:

Генерация отчетов по текущему репертуару, заполняемости залов и выручке от продаж билетов.

Основные требования к базе данных

1. Целостность данных: База данных должна обеспечивать целостность и согласованность хранимой информации.
2. Безопасность: Доступ к базе данных должен быть ограничен для неавторизованных пользователей.
3. Масштабируемость: База данных должна быть способна обрабатывать увеличивающиеся объемы данных по мере роста количества кинотеатров и фильмов.
4. Удобство использования: Интерфейс для сотрудников должен быть интуитивно понятным и удобным для работы
5. Актуальность информации: База данных должна позволять быстро обновлять информацию о фильмах и сеансах.

Основные таблицы

1. Кинотеатр - таблица, содержащая информацию о кинотеатрах, включая их название, расположение и дополнительные услуги.
2. Кинозал – таблица, содержащая информацию о кинозалах, включая название, идентификатор кинотеатра, к которому принадлежит, дополнительные услуги.
3. Фильм - таблица, содержащая данные о фильмах, включая название, режиссера, актерский состав и жанр.
4. Сеанс - таблица, содержащая расписание сеансов, связывающая фильмы и кинотеатры с указанием времени начала и зала.
5. Дополнительные услуги - таблица, содержащая информацию о дополнительных услугах, предлагаемых в кинотеатрах.

Эти таблицы образуют основу для построения системы управления данными о кинотеатрах. Чтобы лучше понять взаимосвязи между этими сущностями и их роли в системе, мы можем представить диаграмму объектов.

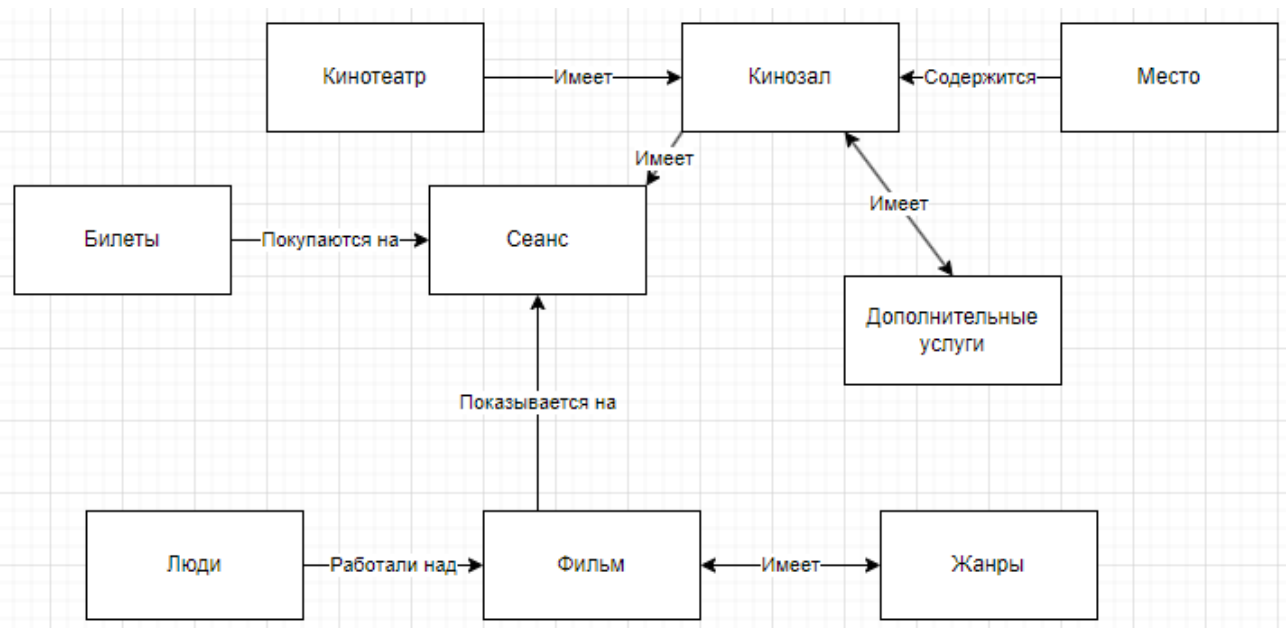


Рис. 1. Диаграмма объектов

На диаграмме объектов представлена более детальная структура базы данных. Каждый Кинотеатр содержит несколько Кинозалов, а в каждом кинозале располагаются Места, которые могут быть забронированы для сеансов.

Кроме того, Кинозал может предлагать различные Дополнительные услуги, что создает сложную взаимосвязь между этими сущностями (М:М).

Сеансы показываются в определённых кинозалах и связывают фильмы с конкретным временем и местом. Каждый Сеанс также связан с Билетами, которые покупаются зрителями.

Что касается Фильмов, они показываются на сеансах и могут принадлежать нескольким Жанрам (М:М), что позволяет разнообразить репертуар. Также стоит отметить, что над каждым фильмом работают различные Люди (М:М), включая режиссеров и актеров, что добавляет еще один уровень сложности к нашей модели данных.

Эта структура позволит эффективно управлять информацией о кинотеатрах, фильмах и сеансах, а также обеспечит возможность гибкого изменения данных в случае необходимости.

Атрибуты основных сущностей:

1. Кинотеатр

- ID (первичный ключ)

- Название
- Состояние кинотеатра
- Рейтинг
- Расположение
- Номер телефона
- Вместимость
- Количество залов
- Дополнительные услуги
- Время начала работы
- Время окончания работы

2. Кинозал

- ID (первичный ключ)
- ID кинотеатра (внешний ключ на таблицу Кинотеатр)
- Название
- Состояние кинозала
- Тип проектора
- Тип экрана
- Наличие видеокамеры
- Дата последнего ремонта
- Количество мест
- Количество рядов

3. Фильм

- ID (первичный ключ)
- Название
- Описание
- Год выпуска
- Длительность
- Страна создания
- Язык фильма

- Рейтинг
- Формат фильма
- Цена проката фильма

4. Сеанс

- ID (первичный ключ)
- ID кинозала (внешний ключ на таблицу Кинозал)
- ID фильма (внешний ключ на таблицу Фильм)
- Время начала
- Время окончания
- Стоимость
- Состояние сеанса (Доступен или продан)

5. Дополнительные услуги (AdditionalService)

- ID (первичный ключ)
- Название услуги
- Описание

Связи между сущностями:

- Кинотеатр может иметь множество кинозалов (1:M)
- Кинозал может иметь множество сеансов (1:M)
- Кинозал может иметь множество мест (1:M)
- Один кинозал может иметь несколько дополнительных услуг, и одна дополнительная услуга может принадлежать многим кинозалам (M:M)
- Кинозал может показывать 1 сеанс (1:1)
- На сеанс может покупаться множество билетов (1:M)
- На сеансе может показываться 1 фильм (1:1)
- Над 1 фильмом может работать множество человек, и 1 человек может работать над многими фильмами (M:M)
- Фильм может иметь множество жанров, и жанр может принадлежать множеству фильмов (M:M)

Нормализация базы данных

Нормализация базы данных — это процесс организации данных в реляционной базе данных для уменьшения избыточности и повышения целостности данных. Основные цели нормализации включают:

1. Устранение избыточности: нормализация помогает избежать дублирования данных, что экономит место и упрощает управление данными.
2. Повышение целостности данных: при изменении данных (например, обновлении или удалении) нормализованные структуры помогают избежать аномалий, которые могут возникнуть при работе с избыточными данными.
3. Упрощение управления: нормализованные таблицы легче поддерживать и модифицировать, так как изменения в одной таблице не требуют изменений в других.
4. Оптимизация запросов: нормализация может улучшить производительность запросов, так как данные структурированы логически и могут быть легко связаны.

Пример нормализации для таблиц "Фильмы" и "Жанры"

Таблица 1. Фильмы

ID	Название	Жанр	Год выпуска
1	Фильм А	Драма	2021
2	Фильм Б	Комедия	2020
3	Фильм В	Драма, комедия	2022

В данной таблице наблюдается избыточность, так как жанры для фильма В записаны в одной ячейке, что затрудняет поиск и фильтрацию по жанрам.

Для достижения 1 НФ необходимо устранить повторяющиеся группы и привести все значения к атомарному виду. Мы можем создать отдельную запись для каждого жанра фильма.

Таблиц 2. Фильмы

ID	Название	Год выпуска
1	Фильм А	2021
2	Фильм Б	2020
3	Фильм В	2022

Таблица 3. Жанры

Фильм_ID	Жанр
1	Драма
2	Комедия
3	Драма
3	Комедия

Теперь каждая запись в таблице "Жанры" соответствует одному жанру для конкретного фильма.

Для достижения 2 НФ нужно устранить частичные зависимости. В нашем случае у нас нет частичных зависимостей, так как все атрибуты в таблице "Фильмы" зависят от первичного ключа (ID фильма). Однако мы можем выделить жанры в отдельную таблицу.

Таблица 4. Фильмы

ID	Название	Год выпуска
1	Фильм А	2021
2	Фильм Б	2020
3	Фильм В	2022

Таблица 5. Жанры

ID	Жанр
1	Драма
2	Комедия

Таблица 6: Фильм_Жанр

Фильм_ID	Жанр_ID
1	1
2	2
3	1
3	2

Теперь у нас есть отдельная таблица "Жанры", где каждый жанр имеет уникальный идентификатор, а таблица "Фильм_Жанр" связывает фильмы с их жанрами.

Для достижения 3 НФ необходимо устранить транзитивные зависимости. В нашем случае все зависимости уже находятся в нормальной форме, так как ни один из атрибутов не зависит от других атрибутов, кроме первичных ключей.

ER-диаграмма

Теперь, чтобы лучше понять взаимосвязи между этими сущностями и их атрибутами, представим их в виде ER-диаграммы. Эта диаграмма визуализирует структуру нашей базы данных.

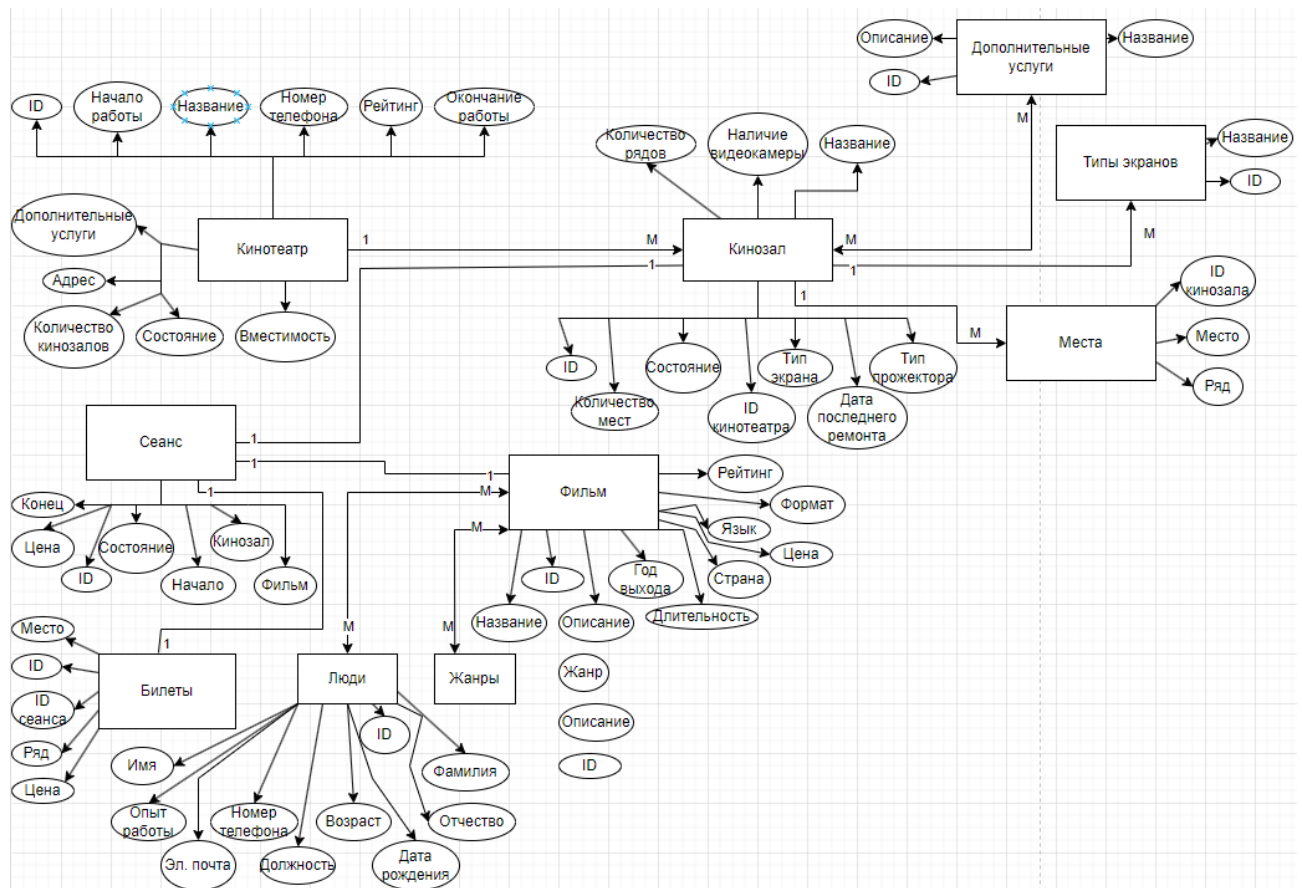


Рис. 2. ER-диаграмма

Создание таблиц

Таблица кинотеатр

```
CREATE TABLE cinema_houses (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(128) NOT NULL,  
    house_state STATES DEFAULT 'Работает',  
    rating NUMERIC(4, 2) check (rating > 0 AND rating < 10),  
    address VARCHAR(256) NOT NULL,  
    phone_number VARCHAR(64) NOT NULL,  
    number_halls INTEGER DEFAULT 0,  
    capacity INTEGER DEFAULT 0,  
    additional_services BOOLEAN DEFAULT false,  
    start_work TIME NOT NULL,  
    end_work TIME NOT NULL  
);
```

Для атрибута `house_state` был создан отдельный тип, потому что данное решение обладает рядом преимуществ:

1. Упрощение управления состояниями: использование перечисляемого типа (ENUM) позволяет четко определить допустимые значения для состояния дома. Это уменьшает вероятность ошибок, связанных с вводом данных, так как база данных будет принимать только заранее определенные состояния.

2. Повышение читаемости кода: ENUM типы делают структуру базы данных более понятной. При чтении схемы базы данных сразу видно, какие состояния могут быть у дома, что улучшает понимание для разработчиков и администраторов.

3. Улучшение целостности данных: ENUM гарантирует, что в поле `house_state` не будут записаны некорректные или неожиданные значения, что помогает поддерживать целостность данных.

4. Легкость в изменении: если в будущем потребуется добавить новое состояние или изменить существующее, это можно сделать централизованно в определении ENUM, что упрощает процесс обновления.

5. Оптимизация хранения: ENUM типы могут занимать меньше места по сравнению с текстовыми строками, так как они хранятся как целые числа, что может привести к экономии памяти в больших таблицах.

```
CREATE TYPE STATES AS ENUM ('Работает', 'Ремонт', 'Закрит');
```

Таблица кинозал

```
CREATE TABLE cinema_halls (  
    id SERIAL PRIMARY KEY,  
    cinema_house_id INTEGER REFERENCES cinema_houses(id),  
    name VARCHAR(64),  
    hall_state STATES DEFAULT 'Работает',  
    projector_type PROJECTOR DEFAULT 'Цифровой',  
    screen_type INTEGER REFERENCES screen_types(id),  
    videocamera BOOLEAN DEFAULT true,  
    last_renovation_date DATE NOT NULL,  
    seats_number INTEGER DEFAULT 0,  
    rows_number INTEGER DEFAULT 0  
);
```

Для атрибута `projector_type` был создан свой тип `PROJECTOR`. Создание собственного типа `PROJECTOR` в виде перечисления (ENUM) позволяет четко определить допустимые значения для типов проекторов, что улучшает читаемость и поддержку базы данных. Это решение обеспечивает:

1. Ясность: перечисление явно указывает на возможные типы проекторов, что упрощает понимание структуры данных.

2. Безопасность данных: использование ENUM предотвращает ввод некорректных значений, что снижает вероятность ошибок.

3. Удобство рефакторинга: добавление новых типов проекторов становится централизованным, что упрощает обновление кода.

4. Оптимизация запросов: ENUM может быть более эффективным с точки зрения хранения и производительности по сравнению с текстовыми строками.

```
CREATE TYPE PROJECTOR AS ENUM ('Цифровой', 'Лазерный', 'UHD', '4K', '3D');
```

Для атрибута screen_type была создана отдельная таблица Типы экранов.

Таблица типы экранов

```
CREATE TABLE screen_types (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(64) NOT NULL  
);
```

Создание отдельной таблицы для перечисления типов экрана имеет несколько преимуществ:

1. Нормализация данных: это позволяет избежать дублирования информации. Вместо того чтобы хранить тип экрана в каждой записи, мы можем ссылаться на его уникальный идентификатор.

2. Упрощение изменений: если необходимо изменить название или добавить новый тип экрана, это можно сделать в одном месте (в таблице screen_types), что упрощает управление данными.

3. Улучшение целостности данных: использование внешних ключей для связи с таблицей типов экранов помогает поддерживать целостность данных и предотвращает ошибки, такие как использование несуществующих типов экранов.

4. Гибкость и расширяемость: в будущем, если потребуется добавить дополнительные атрибуты для типов экранов (например, описание или категории), это можно сделать, не изменяя структуру других таблиц.

5. Упрощение запросов: с отдельной таблицей можно легко выполнять запросы для получения информации о типах экранов, что улучшает читаемость и поддержку кода.

Таблица дополнительные услуги

```
CREATE TABLE additional_services (
```

```
id SERIAL PRIMARY KEY,  
name VARCHAR(256) NOT NULL,  
description TEXT  
);
```

Таблица связи кинозала и дополнительных услуг

```
CREATE TABLE addit_hall (  
    hall_id INTEGER REFERENCES cinema_halls(id),  
    service_id INTEGER REFERENCES additional_services(id),  
    PRIMARY KEY (hall_id, service_id)  
);
```

Так как между данными сущностями имеется связь многие ко многим, была создана отдельная таблица. Создание отдельной таблицы для данной связи обусловлено несколькими причинами:

1. Упрощение структуры данных: когда две сущности могут иметь множество взаимосвязей, создание отдельной таблицы позволяет четко организовать эти связи, избегая дублирования данных и упрощая структуру базы данных.

2. Гибкость: отдельная таблица связки позволяет добавлять дополнительные атрибуты к отношениям между сущностями. Например, можно хранить информацию о дате создания связи или статусе, что невозможно сделать при прямой связи.

3. Поддержка нормализации: создание отдельной таблицы способствует нормализации базы данных, что помогает избежать избыточности и аномалий обновления. Это улучшает целостность данных.

4. Упрощение запросов: с помощью таблицы связки можно легко выполнять запросы для получения связанных данных, что делает работу с базой данных более эффективной и понятной.

Таблица места

```
CREATE TABLE seats (  
    hall_id INTEGER REFERENCES cinema_halls(id),
```

```
row INTEGER CHECK (row > 0),
seats INTEGER CHECK (seats > 0),
PRIMARY KEY (hall_id, row)
);
```

Таблица фильмы

```
CREATE TABLE films (
    id SERIAL PRIMARY KEY,
    title TEXT NOT NULL,
    description TEXT,
    release_year DATE NOT NULL,
    duration INTEGER CHECK (duration > 0),
    country VARCHAR(64) DEFAULT 'Россия',
    film_language VARCHAR(64) DEFAULT 'Русский',
    rating NUMERIC(4, 2) CHECK (rating > 0 AND rating < 10),
    format VARCHAR(16) CHECK (format in ('2D', '3D', 'IMAX')),
    price MONEY
);
```

Таблица люди

```
CREATE TABLE persons (
    id SERIAL PRIMARY KEY,
    firstname TEXT NOT NULL,
    secondname TEXT NOT NULL,
    thirdname TEXT,
    birth_date DATE NOT NULL,
    age INTEGER,
    job VARCHAR(32) CHECK (job in ('Актёр', 'Продюсер')) DEFAULT 'Актёр',
    phone TEXT NOT NULL,
    email TEXT NOT NULL,
    experience INTEGER NOT NULL CHECK (experience > 0)
);
```

Таблица связи фильмов и людей

Так как связь между сущностями многие ко многим, мы поступили аналогичным образом, как и с сущностями дополнительные услуги и кинозал, создав отдельную таблицу связи:

```
CREATE TABLE film_person (  
    film_id INTEGER REFERENCES films(id),  
    person_id INTEGER REFERENCES persons(id),  
    PRIMARY KEY (film_id, person_id)  
);
```

Таблица жанров

```
CREATE TABLE genres (  
    id SERIAL PRIMARY KEY,  
    description TEXT,  
    genre TEXT NOT NULL  
);
```

Таблица связи фильмов и жанров

```
CREATE TABLE film_genre (  
    film_id INTEGER REFERENCES films(id),  
    genre_id INTEGER REFERENCES genres(id),  
    PRIMARY KEY (film_id, genre_id)  
);
```

Таблица сеансов

```
CREATE TABLE film_screenings (  
    id SERIAL PRIMARY KEY,  
    film INTEGER REFERENCES films(id),  
    hall INTEGER REFERENCES cinema_halls(id),  
    start_time TIMESTAMP NOT NULL,  
    end_time TIMESTAMP NOT NULL,  
    price MONEY,  
    screening_state VARCHAR(32) CHECK (screening_state in
```

```

        ('Доступен', 'Продан')) DEFAULT 'Доступен'
    );

Таблица билетов

CREATE TABLE tickets (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    screening_id INTEGER REFERENCES film_screenings(id),
    row SMALLINT,
    seat SMALLINT,
    price MONEY
);

```

Выбор типа данных UUID для хранения уникальных идентификаторов в нашей базе данных обусловлен несколькими важными факторами. UUID (Universally Unique Identifier) представляют собой 128-битные значения, которые обеспечивают уникальность идентификаторов на глобальном уровне. В контексте нашей системы, где номера билетов могут быть напечатаны на бумажных носителях или представлены в электронных форматах, использование UUID позволяет избежать раскрытия внутренней информации о бизнесе.

Традиционные последовательные идентификаторы, такие как тип данных `serial`, могут подвергать систему риску, так как их предсказуемая природа позволяет злоумышленникам легко итерировать по идентификаторам и потенциально получать доступ к данным других пользователей. Например, если злоумышленник знает, что номера билетов начинаются с определенного значения и идут последовательно, он может попытаться получить информацию о продажах, анализируя данные по времени покупки.

С другой стороны, использование UUID значительно снижает вероятность таких атак. Благодаря своей случайной природе и большому объему возможных значений, UUID практически исключает возможность предсказания следующих идентификаторов. Таким образом, применение этого типа данных не только повышает уровень безопасности, но и защищает конфиденциальность бизнес-

данных, что является критически важным аспектом в современных распределенных системах.

Чтобы использовать тип данных UUID, нужно активировать расширение, которое предоставляет функции для генерации UUID. Производится это командой:

```
CREATE EXTENSION "uuid-oss";
```

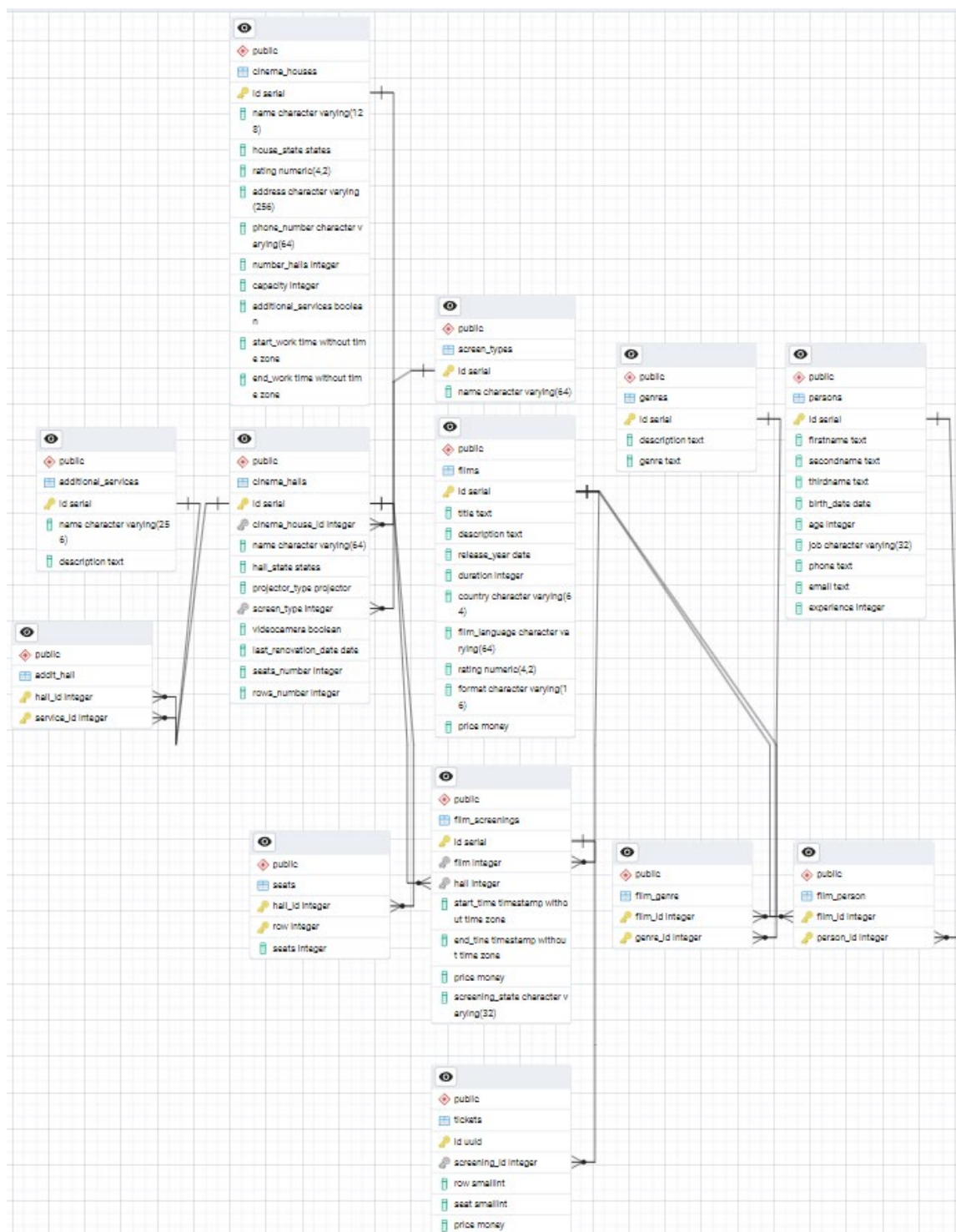


Рис. 3. Дополненная ER-диаграмма

Создание триггеров

Триггер для обновления поля number_halls в таблице cinema_houses

Этот триггер будет срабатывать после добавления нового кинозала и обновлять количество залов в соответствующем кинотеатре.

```
CREATE OR REPLACE FUNCTION update_number_halls()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE cinema_houses
    SET number_halls = (SELECT COUNT(*) FROM cinema_halls WHERE
cinema_house_id = NEW.cinema_house_id)
    WHERE id = NEW.cinema_house_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_update_number_halls
AFTER INSERT ON cinema_halls
FOR EACH ROW
EXECUTE FUNCTION update_number_halls();
```

Триггер для обновления поля capacity в таблице cinema_houses

Этот триггер будет срабатывать при добавлении нового кинозала и обновлять вместимость кинотеатра на основе всех залов.

```
CREATE OR REPLACE FUNCTION update_capacity()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE cinema_houses
    SET capacity = (SELECT SUM(seats_number) FROM cinema_halls WHERE
cinema_house_id = NEW.cinema_house_id)
    WHERE id = NEW.cinema_house_id;
    RETURN NEW;
END;
```



```

$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_update_capacity
AFTER INSERT OR UPDATE ON cinema_halls
FOR EACH ROW
EXECUTE FUNCTION update_capacity();

```

Триггер для обновления поля additional_services в таблице cinema_houses

Этот триггер будет срабатывать после добавления дополнительных услуг в кинозале, и обновлять поле additional_services в таблице cinema_houses.

```

CREATE OR REPLACE FUNCTION update_additional_services()
RETURNS TRIGGER AS $$
BEGIN

```

```

    UPDATE cinema_houses
    SET additional_services = TRUE
    WHERE id = (
        SELECT cinema_house_id
        FROM cinema_halls
        WHERE id = NEW.hall_id
        LIMIT 1

```

```

    )

```

```

    AND EXISTS (
        SELECT 1
        FROM cinema_halls
        JOIN addit_hall ON cinema_halls.id = addit_hall.hall_id
        WHERE cinema_halls.cinema_house_id = (
            SELECT cinema_house_id
            FROM cinema_halls
            WHERE id = NEW.hall_id
            LIMIT 1

```

```

        )

```

```

    );

```

```

        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_update_additional_services
AFTER INSERT ON addit_hall
FOR EACH ROW
EXECUTE FUNCTION update_additional_services();

```

Триггер для обновления полей seats_number и rows_number в таблице cinema_halls

Для создания триггера, который будет обновлять поля seats_number и rows_number в таблице cinema_halls при добавлении новой строки в таблицу seats, нужно написать триггер, который будет отслеживать вставку данных в таблицу seats и изменять соответствующие значения в таблице cinema_halls.

```

CREATE OR REPLACE FUNCTION update_hall_seat_count()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE cinema_halls
    SET rows_number = (SELECT COUNT(DISTINCT row) FROM seats
    WHERE hall_id = NEW.hall_id) WHERE id = NEW.hall_id;
    UPDATE cinema_halls
    SET seats_number = (SELECT SUM(seats) FROM seats WHERE hall_id =
    NEW.hall_id) WHERE id = NEW.hall_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER update_seat_count_after_insert
AFTER INSERT ON seats
FOR EACH ROW
EXECUTE FUNCTION update_hall_seat_count();

```

При вставке новой строки в таблицу `seats`, триггер вызывает функцию `update_hall_seat_count`. В первой части функции обновляется количество рядов (используется `COUNT(DISTINCT row)` для подсчета уникальных рядов. Во второй части обновляется общее количество мест (суммируется значение столбца `seats`).

Триггер `update_seat_count_after_insert` срабатывает после добавления новой строки в таблицу `seats`. Для каждой новой строки в таблице `seats` вызывается функция, которая обновляет соответствующие значения в таблице `cinema_halls`. С этим триггером, когда добавляется новая строка в таблицу `seats`, количество мест и рядов в соответствующем кинозале будет автоматически обновляться в таблице `cinema_halls`.

Триггер для вычисления поля `age` в таблице `persons`

Чтобы поле `age` в таблице `persons` автоматически вычислялось на основе даты рождения (`birth_date`), можно создать триггер, который будет обновлять это поле при вставке или обновлении строки. В PostgreSQL можно использовать функцию, которая будет рассчитывать возраст на основе текущей даты.

```
CREATE OR REPLACE FUNCTION calculate_age()
RETURNS TRIGGER AS $$
BEGIN
    NEW.age := EXTRACT(YEAR FROM age(NEW.birth_date));
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER update_age_before_insert_or_update
BEFORE INSERT OR UPDATE ON persons
FOR EACH ROW
EXECUTE FUNCTION calculate_age();
```

Функция `calculate_age` вычисляет возраст для каждого нового или обновленного значения даты рождения. В функции используется функция PostgreSQL `age()`, которая возвращает интервал между датой рождения и

текущей датой. Затем с помощью EXTRACT(YEAR FROM age(...)) извлекается количество полных лет (возраст).

Триггер update_age_before_insert_or_update срабатывает перед вставкой или обновлением строки в таблице persons (срабатывает до того, как данные будут сохранены в таблице). Для каждой строки, которую добавляют или обновляют в таблице, триггер вызывает функцию calculate_age, которая обновляет значение поля age. Теперь, когда происходит добавление или обновление строки в таблице persons, поле age будет автоматически вычисляться на основе значения поля birth_date.

Триггер для таблицы билетов

Следующее, что я хочу сделать – это защитить таблицу tickets от ввода ошибочных данных. Я хочу предотвратить “овербукинг” – то есть продажу большего количества билетов чем есть мест в зале. Так как это ограничение накладывается внешней таблицей то я не смогу использовать индексы или ограничения таблицы

```
CREATE FUNCTION check_overbooking() RETURNS TRIGGER AS $func$
DECLARE
    seat_possible BOOLEAN;
BEGIN
    SELECT true INTO seat_possible
    FROM seats
    JOIN film_screenings ON film_screenings.hall = seats.hall_id
    WHERE film_screenings.id = new.screening_id
    AND seats.row = new.row
    AND new.seat BETWEEN 1 AND seats.seats;
    IF (seat_possible IS NULL OR NOT seat_possible) THEN
        RAISE EXCEPTION 'The seat % in row % does not exist for screening %',
        new.seat, new.row, new.screening_id;
        RETURN NULL;
    END IF;
```

```

IF EXISTS (
    SELECT 1
    FROM tickets
    WHERE screening_id = new.screening_id
        AND row = new.row
        AND seat = new.seat
) THEN
    RAISE EXCEPTION 'The seat % in row % is already booked for screening %',
new.seat, new.row, new.screening_id;
    RETURN NULL;
END IF;
RETURN NEW;
END;
$func$ LANGUAGE plpgsql;
CREATE TRIGGER check_overbooking
BEFORE INSERT ON tickets
FOR EACH ROW
EXECUTE FUNCTION check_overbooking();

```

Запрос в этой функции вернет true только в случае если в зале, соответствующем сеансу, указанному на билете, существует выбранное место в указанном ряду.

Триггер для таблицы сеансов

В таблице Фильмы указанная цена – цена проката фильма, но цена сеанса имеет другую смысловую нагрузку. Формула вычисления цены сеанса – цена проката фильма складывается с количеством дополнительных услуг в кинозале, умноженным на 10. Данный триггер будет вычислять стоимость билета на сеанс.

```

CREATE OR REPLACE FUNCTION calculate_screening_price()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE film_screenings

```

```

        SET price = (SELECT price FROM films WHERE id = NEW.film) +
        ((SELECT COUNT(*) * 10 FROM addit_hall WHERE hall_id =
NEW.hall)::MONEY)
        WHERE id = NEW.id;
        RETURN NEW;
END;

```

```

$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_calculate_screening_price
AFTER INSERT ON film_screenings
FOR EACH ROW
EXECUTE FUNCTION calculate_screening_price();

```

Триггер для таблицы Билеты

Данный триггер будет автоматически устанавливать стоимость билета на сеанс, взяв стоимость из таблицы Сеансы

```

CREATE OR REPLACE FUNCTION set_ticket_price() RETURNS TRIGGER AS
$$
BEGIN

```

```

    SELECT price INTO NEW.price
    FROM film_screenings
    WHERE film_screenings.id = NEW.screening_id;
    RETURN NEW;

```

```

END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER set_ticket_price_trigger
BEFORE INSERT ON tickets
FOR EACH ROW
EXECUTE FUNCTION set_ticket_price();

```

Заполнение таблиц данными

```
INSERT INTO cinema_houses (name, rating, address, phone_number, start_work,
end_work) VALUES
('Кинотеатр Люкс', 8.5, 'Казань, ул. Центральная, 12', '+7 123 456 7890',
'10:00:00', '22:00:00'),
('СинемаПарк', 7.2, 'Казань, пр. Победы, 45', '+7 987 654 3210', '09:30:00',
'23:00:00'),
('Кинотеатр Галактика', 9.1, 'Казань, ул. Мира, 101', '+7 555 123 4567', '11:00:00',
'00:00:00'),
('Синемаград', 6.8, 'Казань, ул. Ленина, 34', '+7 800 123 9876', '10:30:00',
'21:30:00'),
('Вестерн Кино', 8.2, 'Казань, пр. Советский, 56', '+7 123 987 6543', '12:00:00',
'23:30:00');
```

	id [PK] integer	name character varying (128)	house_state states	rating numeric (4,2)	address character varying (256)	phone_number character varying (16)	number_halls integer	capacity integer	additional_services boolean	start_work time without time zone	end_work time without time zone
1	1	Кинотеатр Люкс	Работает	8.50	Казань, ул. Центра...	+7 123 456 7890	0	0	false	10:00:00	22:00:00
2	2	СинемаПарк	Работает	7.20	Казань, пр. Победы...	+7 987 654 3210	0	0	false	09:30:00	23:00:00
3	3	Кинотеатр Галактика	Работает	9.10	Казань, ул. Мира, 1...	+7 555 123 4567	0	0	false	11:00:00	00:00:00
4	4	Синемаград	Работает	6.80	Казань, ул. Ленина,...	+7 800 123 9876	0	0	false	10:30:00	21:30:00
5	5	Вестерн Кино	Работает	8.20	Казань, пр. Советс...	+7 123 987 6543	0	0	false	12:00:00	23:30:00

Рис. 4. Таблица cinema_houses

```
INSERT INTO cinema_halls (cinema_house_id, name, hall_state, projector_type,
screen_type, videocamera, last_renovation_date) VALUES
(1, 'Зал 1', 'Работает', 'Цифровой', 1, true, '2023-05-10'),
(2, 'Зал 1', 'Работает', 'Лазерный', 2, true, '2023-06-15'),
(2, 'Зал 2', 'Работает', 'UHD', 1, true, '2023-06-15'),
(3, 'Зал 1', 'Работает', '4K', 3, true, '2023-04-20'),
(3, 'Зал 2', 'Работает', 'Цифровой', 2, true, '2023-04-20'),
(3, 'Зал 3', 'Работает', 'Лазерный', 1, true, '2023-04-20'),
(4, 'Зал 1', 'Работает', 'UHD', 1, true, '2023-07-01'),
(4, 'Зал 2', 'Работает', 'Цифровой', 2, true, '2023-07-01'),
(4, 'Зал 3', 'Работает', '4K', 3, true, '2023-07-01'),
(4, 'Зал 4', 'Работает', '3D', 1, true, '2023-07-01'),
(5, 'Зал 1', 'Работает', 'Цифровой', 2, true, '2023-08-10'),
```

(5, 'Зал 2', 'Работает', 'Лазерный', 3, true, '2023-08-10');

	id [PK] integer	cinema_house_id integer	name character varying (64)	hall_state states	projector_type projector	screen_type integer	videocamera boolean	last_renovation_date date	seats_number integer	rows_number integer
1	1	1	Зал 1	Работает	Цифровой	1	true	2023-05-10	0	0
2	2	2	Зал 1	Работает	Лазерный	2	true	2023-06-15	0	0
3	3	2	Зал 2	Работает	УHD	1	true	2023-06-15	0	0
4	4	3	Зал 1	Работает	4K	3	true	2023-04-20	0	0
5	5	3	Зал 2	Работает	Цифровой	2	true	2023-04-20	0	0
6	6	3	Зал 3	Работает	Лазерный	1	true	2023-04-20	0	0
7	7	4	Зал 1	Работает	УHD	1	true	2023-07-01	0	0
8	8	4	Зал 2	Работает	Цифровой	2	true	2023-07-01	0	0
9	9	4	Зал 3	Работает	4K	3	true	2023-07-01	0	0
10	10	4	Зал 4	Работает	3D	1	true	2023-07-01	0	0
11	11	5	Зал 1	Работает	Цифровой	2	true	2023-08-10	0	0
12	12	5	Зал 2	Работает	Лазерный	3	true	2023-08-10	0	0

Рис. 5. Таблица cinema_halls;

INSERT INTO screen_types (name) VALUES

('IMAX'),

('2D'),

('3D'),

('4DX'),

('ScreenX');

	id [PK] integer	name character varying (64)
1	1	IMAX
2	2	2D
3	3	3D
4	4	4DX
5	5	ScreenX

Рис. 6. Таблица screen_types

	id [PK] integer	name character varying (256)	description text
1	1	Билеты с доставкой	Удобная доставка билетов на указанный адрес.
2	2	VIP-зал	Специальный зал с улучшенными условиями просмотра.
3	3	Попкорн и напитки	Заказ попкорна и напитков прямо в зал.
4	4	Парковка	Обеспеченная парковка для зрителей.
5	5	Детская комната	Комната для детей с присмотром во время сеанса.

Рис. 7. Таблица additional_services

	hall_id [PK] integer	row [PK] integer	seats integer
1	1	1	15
2	1	2	15
3	1	3	15
4	2	1	20
5	2	2	20
6	2	3	20
7	3	1	18
8	3	2	18
9	3	3	18
10	3	4	18
11	4	1	22
12	4	2	22
13	4	3	22
14	4	4	22
15	5	1	16
16	5	2	16
17	6	1	18
18	6	2	18
19	7	1	20
20	7	2	20
21	8	1	24
22	8	2	24
23	8	3	24
24	9	1	12
25	9	2	12
26	10	1	14
27	10	2	14
28	11	1	16
29	11	2	16
30	12	1	20
31	12	2	20

Рис. 7. Таблица seats

После внесения данных в таблицу Места сработал триггер, который обновил значения рядов и мест в таблице Кинозалы:

	id [PK] integer	seats_number integer	rows_number integer
1	1	45	3
2	2	60	3
3	3	72	4
4	4	88	4
5	5	32	2
6	6	36	2
7	7	40	2
8	8	72	3
9	9	24	2
10	10	28	2
11	11	32	2
12	12	40	2

Рис. 8. Действие триггера на таблицу cinema_halls

Также изменилось значения полей capacity в таблицу Кинотеатры, потому что сработал соответствующий триггер:

	id [PK] integer	name character varying (128)	capacity integer
1	2	СинемаПарк	132
2	3	Кинотеатр Галактика	156
3	1	Кинотеатр Люкс	45
4	4	Синемаград	164
5	5	Вестерн Кино	72

Рис. 9. Действие триггера на таблицу cinema_houses

	hall_id [PK] integer	service_id [PK] integer
1	1	1
2	1	2
3	1	3
4	2	1
5	2	3
6	2	4
7	3	1
8	3	3
9	4	2
10	4	4
11	5	1
12	5	5
13	6	2
14	6	3
15	7	1
16	7	4
17	8	3
18	8	5
19	9	1
20	9	2
21	10	4
22	10	3
23	11	1
24	11	5
25	12	2
26	12	4

Рис. 10. Таблица addit_hall

После внесения данных в таблицу `addit_hall` изменились поля `additional_services` в таблице Кинотеатры, так как сработал соответствующий триггер:

	name character varying (128)	additional_services boolean
1	Кинотеатр Люкс	true
2	СинемаПарк	true
3	Кинотеатр Галактика	true
4	Синемаград	true
5	Вестерн Кино	true

Рис. 11. Действие триггера на таблицу `cinema_houses`

	id [PK] integer	description text	genre text
1	1	Фильмы, которые исследуют человеческие эмоции и конфликты.	Драма
2	2	Фильмы, основанные на научных и фантастических концепциях.	Фантастика
3	3	Фильмы, которые вызывают смех и развлекают зрителей.	Комедия
4	4	Фильмы, которые рассказывают истории о приключениях и действиях.	Экшен
5	5	Фильмы, основанные на романтических отношениях между персонажами.	Романтика
6	6	Фильмы, которые содержат элементы магии и волшебства.	Фэнтези
7	7	Фильмы, основанные на реальных событиях или документальных фактах.	Документальный
8	8	Анимационные фильмы, созданные с использованием различных техник анимац...	Анимация
9	9	Фильмы ужасов, которые призваны напугать зрителей.	Ужасы
10	10	Фильмы, которые исследуют социальные или политические темы.	Социальная драма
11	11	Фильмы, в которых главными героями являются животные или природа.	Приключенческий
12	12	Фильмы, которые используют элементы триллера для создания напряжения.	Триллер
13	13	Фильмы, основанные на мифах и легендах.	Мифология
14	14	Музыкальные фильмы, в которых музыка играет центральную роль.	Музыкальный
15	15	Фильмы о детях и подростках, их проблемах и приключениях.	Подростковый
16	16	Фильмы, которые исследуют темы самопознания и внутренней борьбы.	Психологический
17	17	Исторические фильмы, которые рассказывают о событиях прошлого.	Исторический
18	18	Фильмы о супергероях и их борьбе со злом.	Супергеройский
19	19	Фильмы, которые фокусируются на спортивных событиях и соревнованиях.	Спортивный
20	20	Фильмы, основанные на комиксах или графических новеллах.	Комикс

Рис. 12. Таблица `genres`

	id [PK] integer	firstname text	secondname text	thirdname text	birth_date date	age integer	job character varying (32)	phone text	email text	experience integer
1	3	Владимир	Сафонов	Владимирович	1985-06-15	39	Актёр	+79001234567	ivan.ivanov@example.com	10
2	4	Мария	Петрова	Сергеевна	1990-03-22	34	Продюсер	+79007654321	maria.petrova@example.com	5
3	5	Сергей	Сидоров	[null]	1978-11-30	46	Актёр	+79009876543	sergey.sidorov@example.com	15
4	6	Анна	Кузнецова	Алексеевна	1995-01-10	29	Актёр	+79004561234	anna.kuznetsova@example.com	3
5	7	Дмитрий	Смирнов	[null]	1982-08-05	42	Продюсер	+79005432123	dmitry.smirnov@example.com	8
6	8	Елена	Фёдорова	Викторовна	1989-02-14	35	Актёр	+79006789012	elena.fedorova@example.com	6
7	9	Александр	Николаев	[null]	1975-04-20	49	Продюсер	+79007890123	alexander.nikolaev@example.com	12
8	10	Ольга	Морозова	Петровна	1992-09-05	32	Актёр	+79008901234	olga.morozova@example.com	4
9	11	Владимир	Соловьёв	[null]	1980-12-25	43	Актёр	+79009123456	vladimir.soloviev@example.com	7
10	12	Татьяна	Лебедева	Игоревна	1988-07-30	36	Продюсер	+79002345678	tatiana.lebedeva@example.com	9
11	13	Максим	Григорьев	[null]	1993-11-11	31	Актёр	+79003456789	maxim.grigorev@example.com	2
12	14	Ксения	Семенова	Анатольевна	1991-05-18	33	Продюсер	+79004567890	ksenia.semenova@example.com	11

Рис. 13. Таблица persons

После внесения данных сработал триггер, который вычислил значения полей age по дате рождения:

	firstname text	birth_date date	age integer
1	Владимир	1985-06-15	39
2	Мария	1990-03-22	34
3	Сергей	1978-11-30	46
4	Анна	1995-01-10	29
5	Дмитрий	1982-08-05	42
6	Елена	1989-02-14	35
7	Александр	1975-04-20	49
8	Ольга	1992-09-05	32
9	Владимир	1980-12-25	43
10	Татьяна	1988-07-30	36
11	Максим	1993-11-11	31
12	Ксения	1991-05-18	33

Рис. 14. Действие триггера на таблицу persons

	id [PK] integer	title text	description text	release_year date	duration integer	country character varying (64)	film_language character varying (64)	rating numeric (4,2)	format character varying (16)	price money
1	1	В поисках счастья	Драма о поисках смысла жизни.	2006-01-01	117	США	Английский	8.00	2D	500,00 ?
2	2	Зеленая миля	Фильм о тюремной жизни и чудесах.	1999-12-10	189	США	Английский	9.20	2D	700,00 ?
3	3	Интерстеллар	Научно-фантастический фильм о космосе.	2014-11-07	169	США	Английский	8.60	IMAX	800,00 ?
4	4	Легенда №17	История хоккеиста Валерия Харламова.	2013-01-01	110	Россия	Русский	7.50	2D	400,00 ?
5	5	Титаник	Романтическая драма о трагедии Титаника.	1997-12-19	195	США	Английский	7.80	3D	600,00 ?
6	6	Время первых	Фильм о космонавтах и их подвиге.	2017-04-20	140	Россия	Русский	8.10	2D	450,00 ?
7	7	Сталкер	Фантастическая драма о Зоне.	1979-04-14	163	СССР	Русский	8.30	2D	300,00 ?
8	8	Властелин колец: Братство...	Фэнтези о борьбе за кольцо.	2001-12-19	178	Новая Зеландия	Английский	8.80	IMAX	900,00 ?
9	9	Матрица	Научно-фантастический фильм о виртуаль...	1999-03-31	136	США	Английский	8.70	2D	500,00 ?
10	10	Пираты Карибского моря: ...	Приключения пиратов в Карибском море.	2003-07-09	143	США	Английский	8.00	2D	550,00 ?
11	11	Гарри Поттер и философск...	Фильм о приключениях молодого волшебн...	2001-11-16	152	Великобритания	Английский	7.90	2D	650,00 ?
12	12	Достучаться до небес	Комедия о двух друзьях с необычной мечт...	1997-04-17	103	Германия	Немецкий	8.50	2D	350,00 ?
13	13	Крепкий орешек	Боевик о полицейском в небоскребе.	1988-07-20	132	США	Английский	8.20	2D	400,00 ?
14	14	Шерлок Холмс: Игра теней	Приключения знаменитого детектива.	2011-12-16	129	США/Великобритания	Английский	7.50	2D	500,00 ?
15	15	Джуманджи: Зов джунглей	Приключенческий фильм о волшебной игре.	2017-12-20	119	США	Английский	6.90	3D	450,00 ?
16	16	Книга джунглей	Приключения Маугли в джунглях.	2016-04-15	106	США	Английский	7.40	3D	500,00 ?
17	17	Однажды в Голливуде	Фильм о золотой эпохе Голливуда.	2019-07-26	161	США	Английский	7.60	2D	600,00 ?
18	18	Мстители: Финал	Эпическая битва супергероев.	2019-04-26	181	США	Английский	8.40	IMAX	850,00 ?
19	19	Тайна третьей планеты	Анимационный фильм о приключениях в к...	1981-12-31	75	СССР	Русский	8.00	2D	200,00 ?
20	20	Небо над Берлином	Поэтичная история о ангелах и людях.	1987-02-18	128	Германия/Франция	Немецкий/Французский	8.30	2D	300,00 ?

Рис. 15. Таблица films



	film_id [PK] integer 	genre_id [PK] integer 
1	1	1
2	1	10
3	2	1
4	2	10
5	3	2
6	3	1
7	4	1
8	4	19
9	5	5
10	5	1
11	6	1
12	6	17
13	7	2
14	7	1
15	8	6
16	8	11
17	9	2
18	9	4
19	10	11
20	10	4
21	11	6
22	11	11
23	12	3
24	13	4
25	14	4
26	15	11
27	15	3
28	16	11
29	17	1
30	18	4

Рис. 16. Таблица film_genre



	film_id [PK] integer 	person_id [PK] integer 
1	1	3
2	1	4
3	1	5
4	1	6
5	2	3
6	2	6
7	2	7
8	3	4
9	3	10
10	3	8
11	4	5
12	4	11
13	4	12
14	5	6
15	5	9
16	5	7
17	6	3
18	6	8
19	6	4
20	7	4
21	7	8
22	7	7
23	8	10
24	8	11
25	8	8
26	9	4
27	9	9
28	9	12
29	10	5

Рис. 17. Таблица film_person

	id [PK] integer	film integer	hall integer	start_time timestamp without time zone	end_time timestamp without time zone	price money	screening_state character varying (32)
1	1	1	1	2024-12-22 10:00:00	2024-12-22 12:00:00	530,00 ?	Доступен
2	2	2	2	2024-12-22 14:00:00	2024-12-22 16:00:00	730,00 ?	Доступен
3	3	3	3	2024-12-23 10:00:00	2024-12-23 12:30:00	820,00 ?	Доступен
4	4	4	4	2024-12-23 14:00:00	2024-12-23 16:30:00	420,00 ?	Доступен
5	5	5	5	2024-12-24 10:00:00	2024-12-24 12:30:00	620,00 ?	Доступен
6	6	6	6	2024-12-24 14:00:00	2024-12-24 16:30:00	470,00 ?	Доступен
7	7	7	7	2024-12-25 10:00:00	2024-12-25 12:30:00	320,00 ?	Доступен
8	8	8	8	2024-12-25 14:00:00	2024-12-25 16:30:00	920,00 ?	Доступен
9	9	9	9	2024-12-26 10:00:00	2024-12-26 12:00:00	520,00 ?	Доступен
10	10	10	10	2024-12-26 14:00:00	2024-12-26 16:00:00	570,00 ?	Доступен
11	11	11	11	2024-12-27 10:00:00	2024-12-27 12:00:00	670,00 ?	Доступен
12	12	12	12	2024-12-27 14:00:00	2024-12-27 16:00:00	370,00 ?	Доступен
13	13	13	1	2024-12-28 10:00:00	2024-12-28 12:00:00	430,00 ?	Доступен
14	14	14	2	2024-12-28 14:00:00	2024-12-28 16:00:00	530,00 ?	Доступен
15	15	15	3	2024-12-29 10:00:00	2024-12-29 12:00:00	470,00 ?	Доступен
16	16	16	4	2024-12-29 14:00:00	2024-12-29 16:00:00	520,00 ?	Доступен
17	17	17	5	2024-12-30 10:00:00	2024-12-30 12:00:00	620,00 ?	Доступен
18	18	18	6	2024-12-30 14:00:00	2024-12-30 16:00:00	870,00 ?	Доступен
19	19	19	7	2024-12-31 10:00:00	2024-12-31 12:00:00	220,00 ?	Доступен
20	20	20	8	2024-12-31 14:00:00	2024-12-31 16:00:00	320,00 ?	Доступен
21	21	1	9	2025-01-01 10:00:00	2025-01-01 12:00:00	520,00 ?	Доступен
22	22	2	10	2025-01-01 14:00:00	2025-01-01 16:00:00	720,00 ?	Доступен
23	23	3	11	2025-01-02 10:00:00	2025-01-02 12:30:00	820,00 ?	Доступен
24	24	4	12	2025-01-02 14:00:00	2025-01-02 16:30:00	420,00 ?	Доступен
25	25	5	1	2025-01-03 10:00:00	2025-01-03 12:30:00	630,00 ?	Доступен
26	26	6	2	2025-01-03 14:00:00	2025-01-03 16:30:00	480,00 ?	Доступен
27	27	7	3	2025-01-04 10:00:00	2025-01-04 12:30:00	320,00 ?	Доступен
28	28	8	4	2025-01-04 14:00:00	2025-01-04 16:30:00	920,00 ?	Доступен
29	29	19	5	2025-01-05 10:00:00	2025-01-05 12:30:00	220,00 ?	Доступен
30	30	3	6	2025-01-05 14:00:00	2025-01-05 16:30:00	820,00 ?	Доступен

Рис. 18. Таблица film_screenings

При попытке создать строку в таблице Билеты, в которой указаны неверные значения ряда и номера места (несуществующие значения) происходит выбрасывание исключения, потому что срабатывает соответствующий триггер:

```

1  INSERT INTO tickets (screening_id, row, seat) VALUES
2  (1, 5, 5);

```

Data Output Messages Notifications

```

ERROR:  The seat 5 in row 5 does not exist for screening 1
CONTEXT:  функция PL/pgSQL check_overbooking(), строка 12, оператор RAISE

```

```

ОШИБКА:  The seat 5 in row 5 does not exist for screening 1
SQL state: P0001

```

Рис. 19. Действие триггера на таблицу tickets

Также происходит и при добавлении билета с уже существующими значениями:

```

1  ▾ INSERT INTO tickets (screening_id, row, seat) VALUES
2    (1, 1, 1);
3  ▾ INSERT INTO tickets (screening_id, row, seat) VALUES
4    (1, 1, 1);

```

Data Output Messages Notifications

ERROR: The seat 1 in row 1 is already booked for screening 1
CONTEXT: функция PL/pgSQL check_overbooking(), строка 22, оператор RAISE

ОШИБКА: The seat 1 in row 1 is already booked for screening 1
SQL state: P0001

Рис. 20. Действие триггера на таблицу tickets

При добавлении корректных значений срабатывает триггер, который устанавливает цену билету той, которая находится в таблице Сеансы:

	id [PK] uuid	screening_id integer	row smallint	seat smallint	price money
1	26c7a2ce-7655-4098-af8f-141196f95ae9	1	1	1	530,00 ?

Рис. 21. Действие триггера на таблицу tickets

	id [PK] uuid	screening_id integer	row smallint	seat smallint	price money
1	26c7a2ce-7655-4098-af8f-141196f95ae9	1	1	1	530,00 ?
2	900ba0b6-e91f-43d5-ad24-84196fd1bd...	1	1	11	530,00 ?
3	4a916284-89ec-41c7-8de5-3d095c3ad2...	2	2	2	730,00 ?
4	eca721ca-27a9-434d-b5c6-c3cf080f6284	3	1	2	820,00 ?
5	ffdcfa7f-7b44-4740-bdee-6107b2e23b45	5	1	3	620,00 ?
6	90833f70-756e-4d80-b498-072a18098b...	6	1	4	470,00 ?
7	dfa5094d-5dcb-4dc9-9850-e68a43a6ce...	7	1	5	320,00 ?
8	e64606d3-80e4-404a-91ac-d17ee15fa0...	8	1	6	920,00 ?
9	51d438c4-10ac-4362-9841-ea3985c687...	9	1	7	520,00 ?
10	f6e64e33-2114-4609-8115-c3b7d6993df5	10	1	8	570,00 ?
11	27e2c40d-19f4-44b6-b063-b1af2bb339...	11	1	9	670,00 ?
12	bdf1c21e-0426-4ece-9fdb-0f3d9ea8a986	12	1	10	370,00 ?
13	4864d279-9cfd-4828-b865-24b6fc6dfa8	13	2	1	430,00 ?
14	2b2145c1-c24d-4af7-952e-2068414277df	14	2	2	530,00 ?
15	b94cdc45-7cd1-44c3-82ae-24a0990b1a...	15	2	3	470,00 ?
16	59d57358-89b4-439a-b1be-896ffcce92ce	16	2	4	520,00 ?

Рис. 22. Таблица tickets

Создание индексов

Для данной базы данных можно создать различные индексы, которые ускоряют выборку данных при поиске, сортировке и фильтрации по часто используемым столбцам. Индексы ускоряют выполнение запросов, снижая время их обработки, особенно для больших объемов данных. В данном случае, мы ограничимся четырьмя индексами, которые будут наиболее полезны для типичных запросов, таких как поиск кинотеатров, фильмов, сеансов и билетов. Ниже приведены индексы, их назначение, объяснение, как они ускоряют выборку, а также примеры запросов для тестирования этих индексов.

Индекс на таблицу film_screenings по полю start_time

```
CREATE INDEX idx_film_screenings_start_time ON film_screenings(start_time);
```

Для чего он:

- Этот индекс помогает ускорить запросы, которые фильтруют или сортируют сеансы по времени начала (start_time).

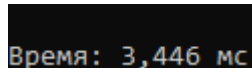
Как ускоряет выборку:

- Когда вы ищете сеансы, которые начинаются в определенный день или после определенного времени, индекс позволяет сразу перейти к нужным строкам, без необходимости сканировать всю таблицу. Это особенно полезно при работе с большими таблицами сеансов.

Пример запроса:

```
SELECT * FROM film_screenings WHERE start_time > '2024-12-23 15:00:00'  
ORDER BY start_time;
```

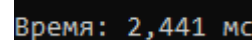
Выполнение данного запроса без индекса:



Время: 3,446 мс

Рис. 23. Выполнение запроса без индекса

Выполнение данного запроса с индексом



Время: 2,441 мс

Рис. 24. Выполнение запроса с индексом

Индекс на таблицу films по полю title

```
CREATE INDEX idx_films_title ON films(title);
```

Для чего он:

- Этот индекс ускоряет поиск фильмов по названию (title).

Как ускоряет выборку:

- Когда вы ищете фильм по его названию, индекс позволяет быстро найти строки, удовлетворяющие условию, вместо того чтобы просматривать всю таблицу. Это особенно важно при большом количестве фильмов.

Пример запроса:

```
SELECT * FROM films WHERE title ILIKE '%Матрица%';
```

Выполнение запроса без использования индекса:

```
cinema=# SELECT * FROM films WHERE title ILIKE '%Матрица%';
 id | title | description
----+-----+-----
  9 | Матрица | Научно-фантастический фильм о виртуальной реальности.
    | 8.70 | 2D      | 500,00 ?
(1 строка)

Время: 2,559 мс
```

Рис. 25. Выполнение запроса без индекса

Выполнение запроса с использованием индекса:

```
cinema=# SELECT * FROM films WHERE title ILIKE '%Матрица%';
 id | title | description
----+-----+-----
  9 | Матрица | Научно-фантастический фильм о виртуальной реальности.
    | 8.70 | 2D      | 500,00 ?
(1 строка)

Время: 0,710 мс
```

Рис. 26. Выполнение запроса с индексом

Композитный индекс на таблицу tickets по полям screening_id, row, seat

```
CREATE INDEX idx_tickets_screening_row_seat ON tickets(screening_id, row,
seat);
```

Для чего он:

- Этот индекс помогает ускорить поиск билетов на конкретное место (ряд и место) для определенного сеанса.

Как ускоряет выборку:

- Когда вам нужно найти билеты для определенного сеанса и места, индекс по комбинации этих трех полей значительно ускоряет поиск. Он эффективно поддерживает запросы с несколькими условиями, что уменьшает количество строк, которые нужно просматривать.

Пример запроса:

```
SELECT * FROM tickets WHERE screening_id = 1 AND row = 5 AND seat = 10;
```

Выполнение запроса без индекса:

```
cinema=# SELECT * FROM tickets WHERE screening_id = 13 AND row = 2 AND seat = 1;
          id          | screening_id | row | seat | price
-----+-----+-----+-----+-----
4864d279-9cfd-4828-b865-24b6fcf6dfa8 |          13 |   2 |   1 | 430,00 ?
(1 строка)

Время: 2,453 мс
```

Рис. 27. Выполнение запроса без индекса

Выполнение запроса с индексом:

```
cinema=# SELECT * FROM tickets WHERE screening_id = 13 AND row = 2 AND seat = 1;
          id          | screening_id | row | seat | price
-----+-----+-----+-----+-----
4864d279-9cfd-4828-b865-24b6fcf6dfa8 |          13 |   2 |   1 | 430,00 ?
(1 строка)

Время: 0,480 мс
```

Рис. 28. Выполнение запроса с индексом

Индексы позволяют базе данных быстро находить строки, удовлетворяющие условиям в запросах, и избежать полного сканирования таблицы. Это значительно ускоряет выполнение запросов, особенно если таблицы содержат большое количество данных.

Создание ролей

Для реализации ролей с соответствующими правами доступа на работу с базой данных, нужно создать три роли — администратор, менеджер и посетитель — и предоставить им необходимые права (Чтение, Добавление, Обновление, Удаление). Права будут раздаваться на основе их должности.

Создадим три роли с помощью команды CREATE ROLE.

Создание роли администратора

```
CREATE ROLE admin LOGIN PASSWORD 'admin';
```

Создание роли менеджера

```
CREATE ROLE manager LOGIN PASSWORD 'manager';
```

Создание роли посетителя

```
CREATE ROLE visitor LOGIN PASSWORD 'visitor';
```

Назначение прав для каждой роли.

1. Роль “Администратор” (full access)

Администратор должен иметь полный доступ ко всем таблицам: право на чтение (SELECT), добавление (INSERT), обновление (UPDATE) и удаление (DELETE) для всех таблиц.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO admin;
```

2. Роль “Менеджер” (ограниченный доступ)

Менеджер будет иметь право на чтение данных (SELECT), добавление новых записей (INSERT) и обновление данных (UPDATE), но не будет иметь права на удаление данных (DELETE). Он не будет иметь права управлять ролями или схемами.

```
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA public TO manager;
```

```
GRANT USAGE, SELECT ON SEQUENCE additional_services_id_seq TO manager;
```

3. Роль “Посетитель” (только чтение)

Посетитель должен иметь доступ только для чтения (SELECT) данных, без возможности изменения или удаления данных.

```
GRANT SELECT ON films TO visitor;  
GRANT SELECT ON cinema_houses TO visitor;  
GRANT SELECT ON cinema_halls TO visitor;  
GRANT SELECT ON film_screenings TO visitor;  
GRANT SELECT ON tickets TO visitor;  
GRANT SELECT ON seats TO visitor;
```

Для того чтобы новые таблицы, создаваемые в будущем, автоматически получали нужные права для каждой роли, можно настроить права по умолчанию.

Настройка прав по умолчанию для администратора

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT,  
INSERT, UPDATE, DELETE ON TABLES TO admin;
```

Настройка прав по умолчанию для менеджера

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT,  
INSERT, UPDATE ON TABLES TO manager;
```

Теперь, когда роли созданы и им назначены соответствующие права, мы можем проверить, что они работают корректно.

При подключении к базе данных в роли Администратора мы имеем полный доступ к базе данных:

```
cinema=# \connect cinema admin  
Пароль пользователя admin:  
Вы подключены к базе данных "cinema" как пользователь "admin".  
cinema=> SELECT * FROM cinema_houses;  
 id |      name      | house_state | rating | ad  
ty | additional_services | start_work | end_work  
-----+-----+-----+-----+-----  
  1 | Кинотеатр Люкс | Работает   | 8.50 | Казань, ул.  
45 | t              | 10:00:00   | 22:00:00  
  2 | СинемаПарк     | Работает   | 7.20 | Казань, пр.  
32 | t              | 09:30:00   | 23:00:00  
  3 | Кинотеатр Галактика | Работает   | 9.10 | Казань, ул.  
56 | t              | 11:00:00   | 00:00:00  
  4 | Синемаград     | Работает   | 6.80 | Казань, ул.  
64 | t              | 10:30:00   | 21:30:00  
  5 | Вестерн Кино   | Работает   | 8.20 | Казань, пр.  
72 | t              | 12:00:00   | 23:30:00  
(5 строк)
```

Рис. 29. Роль admin

При подключении в роли менеджера мы не имеем возможности удалять данные из таблиц:

```
cinema=# \connect cinema manager
Пароль пользователя manager:
Вы подключены к базе данных "cinema" как пользователь "manager".
cinema=> INSERT INTO additional_services(name, description) VALUES
cinema-> ('Test', 'test');
INSERT 0 1
Время: 2,930 мс
cinema=> SELECT * FROM additional_services;
```

id	name	description
1	Билеты с доставкой	Удобная доставка билетов на указанный адрес.
2	VIP-зал	Специальный зал с улучшенными условиями просмотра.
3	Попкорн и напитки	Заказ попкорна и напитков прямо в зал.
4	Парковка	Обеспеченная парковка для зрителей.
5	Детская комната	Комната для детей с присмотром во время сеанса.
6	Test	test

```
(6 строк)

Время: 1,992 мс
cinema=> DELETE FROM additional_services WHERE id = 6;
ОШИБКА: нет доступа к таблице additional_services
```

Рис. 30. Роль manager

При подключении в роли посетителя мы имеем доступ не ко всем таблицам и имеем возможность только просматривать данные:

```
cinema=> \connect cinema visitor
Пароль пользователя visitor:
Вы подключены к базе данных "cinema" как пользователь "visitor".
cinema=> SELECT * FROM genres;
ОШИБКА: нет доступа к таблице genres
Время: 5,907 мс
```

Рис. 31. Роль visitor

Объяснение прав доступа:

1. Администратор (*admin*):

- Полный доступ ко всем таблицам базы данных: создание, чтение, обновление и удаление данных.
- Могут управлять всеми аспектами базы данных.

2. Менеджер (*manager*):

- Может читать данные и добавлять/обновлять записи в таблицах, но не может удалять данные.

- Обработывает повседневные операции, связанные с билетами и сеансами, но не может изменять структуру базы данных.

3. *Посетитель (visitor):*

- Может только читать данные, например, просматривать фильмы, сеансы и кинотеатры.
- Не может изменять базу данных или просматривать внутренние данные, такие как платежные детали.

Запросы

Для нашей базы данных я подготовил 10 нетривиальных запросов, которые используют различные виды подзапросов и JOIN, а также два представления для репертуара и дополнительной выборки. Давайте рассмотрим запросы по порядку.

1. Список фильмов, которые были выпущены в определенном диапазоне лет, с дополнительной информацией о жанрах и актерах

```
SELECT f.title AS film_title,  
       f.release_year,  
       f.rating,  
       ARRAY_AGG(g.genre) AS genres,  
       ARRAY_AGG(p.firstname || ' ' || p.secondname) AS actors  
FROM films f  
JOIN film_genre fg ON f.id = fg.film_id  
JOIN genres g ON fg.genre_id = g.id  
JOIN film_person fp ON f.id = fp.film_id  
JOIN persons p ON fp.person_id = p.id  
WHERE EXTRACT(YEAR FROM f.release_year) BETWEEN 2000 AND 2010  
GROUP BY f.id  
ORDER BY f.release_year DESC;
```

	film_title text	release_year date	rating numeric (4,2)	genres text[]	actors text[]
1	В поисках счас...	2006-01-01	8.00	{Драма,"Социальная ...	{"Владимир Сафонов","Владими...
2	Пираты Карибс...	2003-07-09	8.00	{Экшен,Приключенч...	{"Сергей Сидоров","Сергей Сидо...
3	Властелин кол...	2001-12-19	8.80	{Фэнтези,Приключен...	{"Ольга Морозова","Ольга Моро...
4	Гарри Поттер и ...	2001-11-16	7.90	{Фэнтези,Приключен...	{"Владимир Сафонов","Владими...

Рис. 32. Выполнение запроса

2. Топ 3 самых популярных фильма с наибольшим количеством сеансов в каждом кинотеатре

```
SELECT ch.name AS cinema_house_name,  
       f.title AS film_title,  
       COUNT(fs.id) AS num_screenings
```



```

FROM cinema_houses ch
JOIN cinema_halls chh ON ch.id = chh.cinema_house_id
JOIN film_screenings fs ON chh.id = fs.hall
JOIN films f ON fs.film = f.id
GROUP BY ch.name, f.title
ORDER BY num_screenings DESC
LIMIT 3;

```

	cinema_house_name character varying (128)	film_title text	num_screenings bigint
1	Синемаград	Небо над Берлином	3
2	Кинотеатр Галактика	Тайна третьей планеты	3
3	Синемаград	Мстители: Финал	3

Рис. 33. Выполнение запроса

3. Сеансы фильмов, которые будут показываться в течение следующих 30 дней в определенном кинотеатре

```

SELECT fs.start_time,
       f.title AS film_title,
       f.price AS film_price,
       ch.name AS cinema_hall_name
FROM film_screenings fs
JOIN films f ON fs.film = f.id
JOIN cinema_halls ch ON fs.hall = ch.id
WHERE fs.start_time BETWEEN CURRENT_DATE AND CURRENT_DATE +
INTERVAL '30 days'
ORDER BY fs.start_time;

```

	start_time timestamp without time zone	film_title text	film_price money	cinema_hall_name character varying (64)
1	2024-12-23 10:00:00	Интерстеллар	800,00 ?	Зал 2
2	2024-12-23 14:00:00	Легенда №17	400,00 ?	Зал 1
3	2024-12-24 10:00:00	Титаник	600,00 ?	Зал 2
4	2024-12-24 14:00:00	Время первых	450,00 ?	Зал 3
5	2024-12-25 10:00:00	Сталкер	300,00 ?	Зал 1
6	2024-12-25 14:00:00	Властелин колец: Братство кольца	900,00 ?	Зал 2
7	2024-12-26 10:00:00	Матрица	500,00 ?	Зал 3
8	2024-12-26 14:00:00	Пираты Карибского моря: Проклятие черной жемчужин...	550,00 ?	Зал 4
9	2024-12-27 10:00:00	Гарри Поттер и философский камень	650,00 ?	Зал 1
10	2024-12-27 14:00:00	Достучаться до небес	350,00 ?	Зал 2
11	2024-12-28 10:00:00	Крепкий орешек	400,00 ?	Зал 1
12	2024-12-28 14:00:00	Шерлок Холмс: Игра теней	500,00 ?	Зал 1
13	2024-12-29 10:00:00	Джуманджи: Зов джунглей	450,00 ?	Зал 2
14	2024-12-29 14:00:00	Книга джунглей	500,00 ?	Зал 1
15	2024-12-30 10:00:00	Однажды в Голливуде	600,00 ?	Зал 2
16	2024-12-30 14:00:00	Мстители: Финал	850,00 ?	Зал 3
17	2024-12-31 10:00:00	Тайна третьей планеты	200,00 ?	Зал 1
18	2024-12-31 14:00:00	Небо над Берлином	300,00 ?	Зал 2
19	2025-01-01 10:00:00	В поисках счастья	500,00 ?	Зал 3
20	2025-01-01 14:00:00	Зеленая миля	700,00 ?	Зал 4

Рис. 34. Выполнение запроса

4. Средняя стоимость билетов для каждого фильма в зависимости от его жанра и кинотеатра

```

SELECT f.title AS film_title,
       g.genre AS film_genre,
       ch.name AS cinema_hall_name,
       AVG(t.price::NUMERIC) AS avg_ticket_price
FROM films f
JOIN film_genre fg ON f.id = fg.film_id
JOIN genres g ON fg.genre_id = g.id
JOIN film_screenings fs ON f.id = fs.film
JOIN cinema_halls ch ON fs.hall = ch.id
JOIN tickets t ON fs.id = t.screening_id
GROUP BY f.id, g.genre, ch.id
ORDER BY avg_ticket_price DESC;

```

	film_title text	film_genre text	cinema_hall_name character varying (64)	avg_ticket_price numeric
1	Властелин колец: Братство кольца	Фэнтези	Зал 2	920.0000000000000000
2	Властелин колец: Братство кольца	Приключенческий	Зал 2	920.0000000000000000
3	Интерстеллар	Драма	Зал 2	820.0000000000000000
4	Интерстеллар	Фантастика	Зал 2	820.0000000000000000
5	Зеленая миля	Социальная драма	Зал 1	730.0000000000000000
6	Зеленая миля	Драма	Зал 1	730.0000000000000000
7	Гарри Поттер и философский камень	Приключенческий	Зал 1	670.0000000000000000
8	Гарри Поттер и философский камень	Фэнтези	Зал 1	670.0000000000000000
9	Титаник	Драма	Зал 2	620.0000000000000000
10	Титаник	Романтика	Зал 2	620.0000000000000000
11	Пираты Карибского моря: Проклятие черной жемчужин...	Приключенческий	Зал 4	570.0000000000000000
12	Пираты Карибского моря: Проклятие черной жемчужин...	Экшен	Зал 4	570.0000000000000000
13	В поисках счастья	Социальная драма	Зал 1	530.0000000000000000
14	Шерлок Холмс: Игра теней	Экшен	Зал 1	530.0000000000000000
15	В поисках счастья	Драма	Зал 1	530.0000000000000000

Рис. 35. Выполнение запроса

5. Фильмы, у которых нет сеансов в следующем месяце

```

SELECT f.title
FROM films f
WHERE NOT EXISTS (
    SELECT 1
    FROM film_screenings fs
    WHERE fs.film = f.id
        AND fs.start_time BETWEEN DATE_TRUNC('MONTH',
CURRENT_DATE + INTERVAL '1 month') AND DATE_TRUNC('MONTH',
CURRENT_DATE + INTERVAL '2 month')
)
ORDER BY f.title;

```

	title text
1	Шерлок Холмс: Игра теней

Рис. 36. Выполнение запроса

6. Количество билетов, проданных по каждому фильму в каждом кинотеатре

```

SELECT f.title AS film_title, ch.name AS cinema_hall_name, COUNT(t.id) AS
tickets_sold

```

```

FROM tickets t
JOIN film_screenings fs ON t.screening_id = fs.id
JOIN films f ON fs.film = f.id
JOIN cinema_halls ch ON fs.hall = ch.id
GROUP BY f.title, ch.name
ORDER BY tickets_sold DESC;

```

	film_title text	cinema_hall_name character varying (64)	tickets_sold bigint
1	В поисках счастья	Зал 1	2
2	Время первых	Зал 3	1
3	Гарри Поттер и философский камень	Зал 1	1
4	Пираты Карибского моря: Проклятие черной жемчужин...	Зал 4	1
5	Интерстеллар	Зал 2	1
6	Матрица	Зал 3	1
7	Титаник	Зал 2	1
8	Книга джунглей	Зал 1	1
9	Властелин колец: Братство кольца	Зал 2	1
10	Зеленая миля	Зал 1	1
11	Сталкер	Зал 1	1
12	Шерлок Холмс: Игра теней	Зал 1	1
13	Крепкий орешек	Зал 1	1
14	Достучаться до небес	Зал 2	1
15	Джуманджи: Зов джунглей	Зал 2	1

Рис. 37. Выполнение запроса

7. Средний рейтинг фильмов по годам выпуска

```

SELECT f.release_year, AVG(f.rating) AS avg_rating
FROM films f
GROUP BY f.release_year
ORDER BY f.release_year DESC;

```

	release_year date	avg_rating numeric
1	2019-07-26	7.6000000000000000
2	2019-04-26	8.4000000000000000
3	2017-12-20	6.9000000000000000
4	2017-04-20	8.1000000000000000
5	2016-04-15	7.4000000000000000
6	2014-11-07	8.6000000000000000
7	2013-01-01	7.5000000000000000
8	2011-12-16	7.5000000000000000
9	2006-01-01	8.0000000000000000
10	2003-07-09	8.0000000000000000
11	2001-12-19	8.8000000000000000
12	2001-11-16	7.9000000000000000
13	1999-12-10	9.2000000000000000
14	1999-03-31	8.7000000000000000
15	1997-12-19	7.8000000000000000
16	1997-04-17	8.5000000000000000
17	1988-07-20	8.2000000000000000
18	1987-02-18	8.3000000000000000
19	1981-12-31	8.0000000000000000
20	1979-04-14	8.3000000000000000

Рис. 38. Выполнение запроса

8. *Количество сеансов для каждого фильма, который имеет больше 1 жанра, отсортированных по количеству сеансов*

```

SELECT f.title AS film_title, COUNT(fs.id) AS num_screenings
FROM films f
JOIN film_genre fg ON f.id = fg.film_id
JOIN genres g ON fg.genre_id = g.id
JOIN film_screenings fs ON f.id = fs.film
GROUP BY f.id
HAVING COUNT(DISTINCT g.id) > 1
ORDER BY num_screenings DESC;

```

	film_title text	num_screenings bigint
1	В поисках счастья	14
2	Титаник	12
3	Джуманджи: Зов джунглей	10
4	Время первых	10
5	Интерстеллар	8
6	Пираты Карибского моря: Проклятие черной жемчужин...	8
7	Сталкер	8
8	Зеленая миля	8
9	Матрица	6
10	Властелин колец: Братство кольца	6
11	Гарри Поттер и философский камень	4
12	Легенда №17	4

Рис. 39. Выполнение запроса

9. Сеансы фильмов с учетом времени работы кинотеатра

```

SELECT fs.start_time, f.title AS film_title, ch.name AS cinema_hall_name
FROM film_screenings fs
JOIN films f ON fs.film = f.id
JOIN cinema_halls ch ON fs.hall = ch.id
JOIN cinema_houses chh ON ch.cinema_house_id = chh.id
WHERE fs.start_time::TIME BETWEEN chh.start_work AND chh.end_work
ORDER BY fs.start_time;

```

	start_time timestamp without time zone	film_title text	cinema_hall_name character varying (64)
1	2024-12-22 10:00:00	В поисках счастья	Зал 1
2	2024-12-22 14:00:00	Зеленая миля	Зал 1
3	2024-12-23 10:00:00	Интерстеллар	Зал 2
4	2024-12-25 14:00:00	Властелин колец: Братство кольца	Зал 2
5	2024-12-26 14:00:00	Пираты Карибского моря: Проклятие черной жемчужин...	Зал 4
6	2024-12-27 14:00:00	Достучаться до небес	Зал 2
7	2024-12-28 10:00:00	Крепкий орешек	Зал 1
8	2024-12-28 14:00:00	Шерлок Холмс: Игра теней	Зал 1
9	2024-12-29 10:00:00	Джуманджи: Зов джунглей	Зал 2
10	2024-12-31 14:00:00	Небо над Берлином	Зал 2
11	2025-01-01 14:00:00	Зеленая миля	Зал 4
12	2025-01-02 14:00:00	Легенда №17	Зал 2
13	2025-01-03 10:00:00	Титаник	Зал 1
14	2025-01-03 14:00:00	Время первых	Зал 1
15	2025-01-04 10:00:00	Сталкер	Зал 2
16	2025-01-06 14:00:00	Время первых	Зал 2
17	2025-01-07 14:00:00	Небо над Берлином	Зал 3
18	2025-01-08 14:00:00	Книга джунглей	Зал 4
19	2025-01-09 10:00:00	Джуманджи: Зов джунглей	Зал 1
20	2025-01-10 14:00:00	Крепкий орешек	Зал 2
21	2025-01-12 14:00:00	Зеленая миля	Зал 2
22	2025-01-13 10:00:00	В поисках счастья	Зал 1
23	2025-01-13 14:00:00	В поисках счастья	Зал 1

Рис. 40. Выполнение запроса

10. Фильмы, показываемые в кинотеатре с дополнительными услугами (например, VIP-зал)

```

SELECT DISTINCT f.title AS film_title, ch.name AS cinema_hall_name, asv.name
AS service_name
FROM films f
JOIN film_screenings fs ON f.id = fs.film
JOIN cinema_halls ch ON fs.hall = ch.id
JOIN addit_hall ah ON ch.id = ah.hall_id
JOIN additional_services asv ON ah.service_id = asv.id
WHERE asv.name IN ('Билеты с доставкой', 'VIP-зал')
ORDER BY f.title;

```

	film_title text	cinema_hall_name character varying (64)	service_name character varying (256)
1	В поисках счастья	Зал 3	Билеты с доставкой
2	В поисках счастья	Зал 3	VIP-зал
3	В поисках счастья	Зал 1	Билеты с доставкой
4	В поисках счастья	Зал 2	Билеты с доставкой
5	В поисках счастья	Зал 1	VIP-зал
6	Властелин колец: Братство кольца	Зал 1	VIP-зал
7	Властелин колец: Братство кольца	Зал 2	Билеты с доставкой
8	Время первых	Зал 3	VIP-зал
9	Время первых	Зал 2	Билеты с доставкой
10	Время первых	Зал 1	Билеты с доставкой
11	Гарри Поттер и философский камень	Зал 1	Билеты с доставкой
12	Джуманджи: Зов джунглей	Зал 1	VIP-зал
13	Джуманджи: Зов джунглей	Зал 1	Билеты с доставкой
14	Джуманджи: Зов джунглей	Зал 3	VIP-зал
15	Джуманджи: Зов джунглей	Зал 2	Билеты с доставкой
16	Достучаться до небес	Зал 3	VIP-зал
17	Достучаться до небес	Зал 2	VIP-зал

Рис. 41. Выполнение запроса

Объяснение запросов:

1. Список фильмов с жанрами и актерами: Использует JOIN для связывания фильмов с жанрами и актерами, а также агрегирует данные.
2. Топ 3 популярных фильма по количеству сеансов: Группирует фильмы по кинотеатрам и подсчитывает количество сеансов для каждого фильма.
3. Сеансы на следующие 30 дней: Запрос для вывода сеансов, которые будут проходить в течение следующего месяца.
4. Средняя стоимость билетов по жанрам и кинотеатрам: Рассчитывает среднюю стоимость билетов, используя JOIN и AVG().
5. Фильмы без сеансов в следующем месяце: Использует подзапрос с NOT EXISTS, чтобы найти фильмы без сеансов в следующем месяце.
6. Количество проданных билетов по фильмам и кинотеатрам: Считает количество проданных билетов для каждого фильма в кинотеатре.
7. Средний рейтинг фильмов по годам: Использует агрегацию для подсчета среднего рейтинга по годам.

8. Фильмы с более чем 2 жанрами: Фильмы с множеством жанров, фильтруются через HAVING.

9. Сеансы в пределах рабочего времени кинотеатров: Фильтрация по времени работы кинотеатров с использованием ::TIME.

10. Фильмы с дополнительными услугами: Ищет фильмы, для которых предоставляются специальные услуги (например, 3D или IMAX).

Создание представлений

Для получения репертуара на месяц я решил создать представление, которое выводит репертуар на следующие 30 дней:

```
CREATE VIEW repertoire_next_30_days AS
SELECT fs.start_time,
       fs.end_time,
       f.title AS film_title,
       f.price AS ticket_price,
       ch.name AS cinema_hall_name,
       chh.name AS cinema_house_name
FROM film_screenings fs
JOIN films f ON fs.film = f.id
JOIN cinema_halls ch ON fs.hall = ch.id
JOIN cinema_houses chh ON ch.cinema_house_id = chh.id
WHERE fs.start_time BETWEEN CURRENT_DATE AND CURRENT_DATE +
INTERVAL '30 days'
ORDER BY fs.start_time;
```

	start_time timestamp without time zone	end_time timestamp without time zone	film_title text	ticket_price money	cinema_hall_name character varying (64)	cinema_house_name character varying (128)
1	2024-12-23 10:00:00	2024-12-23 12:30:00	Интерстеллар	800,00 ?	Зал 2	СинемаПарк
2	2024-12-23 14:00:00	2024-12-23 16:30:00	Легенда №17	400,00 ?	Зал 1	Кинотеатр Галактика
3	2024-12-24 10:00:00	2024-12-24 12:30:00	Титаник	600,00 ?	Зал 2	Кинотеатр Галактика
4	2024-12-24 14:00:00	2024-12-24 16:30:00	Время первых	450,00 ?	Зал 3	Кинотеатр Галактика
5	2024-12-25 10:00:00	2024-12-25 12:30:00	Сталкер	300,00 ?	Зал 1	Синемаград
6	2024-12-25 14:00:00	2024-12-25 16:30:00	Властелин колец: Братство кольца	900,00 ?	Зал 2	Синемаград
7	2024-12-26 10:00:00	2024-12-26 12:00:00	Матрица	500,00 ?	Зал 3	Синемаград
8	2024-12-26 14:00:00	2024-12-26 16:00:00	Пираты Карибского моря: Проклятие черной жемчужин...	550,00 ?	Зал 4	Синемаград
9	2024-12-27 10:00:00	2024-12-27 12:00:00	Гарри Поттер и философский камень	650,00 ?	Зал 1	Вестерн Кино
10	2024-12-27 14:00:00	2024-12-27 16:00:00	Достучаться до небес	350,00 ?	Зал 2	Вестерн Кино
11	2024-12-28 10:00:00	2024-12-28 12:00:00	Крепкий орешек	400,00 ?	Зал 1	Кинотеатр Люкс
12	2024-12-28 14:00:00	2024-12-28 16:00:00	Шерлок Холмс: Игра теней	500,00 ?	Зал 1	СинемаПарк
13	2024-12-29 10:00:00	2024-12-29 12:00:00	Джуманджи: Зов джунглей	450,00 ?	Зал 2	СинемаПарк
14	2024-12-29 14:00:00	2024-12-29 16:00:00	Книга джунглей	500,00 ?	Зал 1	Кинотеатр Галактика
15	2024-12-30 10:00:00	2024-12-30 12:00:00	Однажды в Голливуде	600,00 ?	Зал 2	Кинотеатр Галактика
16	2024-12-30 14:00:00	2024-12-30 16:00:00	Мстители: Финал	850,00 ?	Зал 3	Кинотеатр Галактика
17	2024-12-31 10:00:00	2024-12-31 12:00:00	Тайна третьей планеты	200,00 ?	Зал 1	СинемаПарк

Рис. 42. Демонстрация представления

Для вывода фильмов с рейтингом выше 8 я создал представление:

```
CREATE VIEW highRatedFilms AS
SELECT f.title, f.release_year, f.rating, f.price
FROM films f
```

WHERE f.rating > 8

ORDER BY f.rating DESC;

	title text	release_year date	rating numeric (4,2)	price money
1	Зеленая миля	1999-12-10	9.20	700,00 ?
2	Властелин колец: Братство кольца	2001-12-19	8.80	900,00 ?
3	Матрица	1999-03-31	8.70	500,00 ?
4	Интерстеллар	2014-11-07	8.60	800,00 ?
5	Достучаться до небес	1997-04-17	8.50	350,00 ?
6	Мстители: Финал	2019-04-26	8.40	850,00 ?
7	Сталкер	1979-04-14	8.30	300,00 ?
8	Небо над Берлином	1987-02-18	8.30	300,00 ?
9	Крепкий орешек	1988-07-20	8.20	400,00 ?
10	Время первых	2017-04-20	8.10	450,00 ?

Рис. 43. Демонстрация представления

Заключение

В ходе выполнения курсовой работы была разработана реляционная база данных для управления системой кинотеатров, включающая создание схемы базы данных, проектирование и реализацию таблиц, а также написание запросов для выполнения различных операций. В рамках работы были решены следующие задачи:

1. Проектирование базы данных: созданы таблицы для хранения информации о кинотеатрах, залах, фильмах, сеансах, билетах и других сущностях, таких как жанры и актеры. Применены подходящие ограничения и связи между таблицами, что позволило обеспечить целостность данных и избежать избыточности.

2. Реализация пользовательских ролей и прав доступа: для обеспечения разграничения доступа были созданы три роли (администратор, менеджер, посетитель) с различными правами на чтение, добавление, обновление и удаление данных. Это позволяет управлять системой в зависимости от должностных обязанностей пользователей.

3. Написание сложных SQL-запросов: были созданы запросы для выборки и агрегации данных, такие как получение списка фильмов с жанрами и актерами, подсчет средней цены билетов, анализ статистики по количеству проданных билетов и т.д. Запросы включают различные виды соединений (JOIN), подзапросы и агрегатные функции, что позволяет эффективно обрабатывать данные.

4. Оптимизация запросов с использованием индексов: для повышения производительности выборки данных были созданы индексы, обеспечивающие быстрый доступ к данным, что важно для работы с большими объемами информации.

5. Создание представлений: для удобства работы с данными были созданы представления, например, для получения репертуара на следующие 30 дней. Это позволяет пользователям получать заранее подготовленные результаты запросов без необходимости многократного выполнения сложных операций.

В результате работы была разработана система, которая может быть использована для управления информацией в кинотеатре, включая управление фильмами, сеансами и билетами. Процесс разработки базы данных продемонстрировал важность правильного проектирования структуры данных и оптимизации запросов для обеспечения высокой производительности системы.

Перспективы развития

В дальнейшем систему можно дополнить новыми функциональными возможностями, такими как:

- Расширение функционала для управления скидками и акциями.
- Интеграция с онлайн-платежными системами для продажи билетов через интернет.
- Внедрение анализа пользовательских предпочтений и рекомендаций по фильмам.

Таким образом, данная работа является основой для создания полноценной системы управления кинотеатром, с возможностью дальнейшего расширения и улучшения.

Список используемых источников

1. PostgreSQL [Электронный ресурс]. – Режим доступа: <https://postgrespro.ru/> (дата обращения: 23.12.2024).
2. Как выбрать PostgreSQL для вашего проекта [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/companies/otus/articles/706346/> (дата обращения: 23.12.2024).
3. Блог о PostgreSQL [Электронный ресурс]. – Режим доступа: <https://gb.ru/blog/postgresql/?ysclid=m50sbcxmj376164860> (дата обращения: 23.12.2024).
4. Ванина, М. Ф., Ерохин, А. Г., Тутова, Н. В. и др. PostgreSQL. Разработка баз данных: учебник. — Москва: Русайнс, 2024. — 227 с. — ISBN 978-5-466-06974-7. — URL: <https://book.ru/book/954200> (дата обращения: 23.12.2024). — Текст: электронный.
5. Ткаченко, С. Н. Основы проектирования баз данных: учебник. — Москва: КноРус, 2024. — 176 с. — ISBN 978-5-406-12054-5. — URL: <https://book.ru/book/950600> (дата обращения: 23.12.2024). — Текст: электронный.
6. Малков, О. Б., Маркова, М. П., Девятерикова, М. В. Работа с СУБД PostgreSQL : учебное пособие. — Омск: ОмГТУ, 2023. — 175 с. — ISBN 978-5-8149-3707-0. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/421547> (дата обращения: 23.12.2024).
7. Романова, И. П., Романов, П. С. Базы данных: работа с PostgreSQL: учебное пособие. — Москва: МУИВ, 2023. — 193 с. — ISBN 978-5-9580-0705-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/443078> (дата обращения: 23.12.2024).
8. Рогов, Е. В. PostgreSQL 15 изнутри: руководство. — Москва: ДМК Пресс, 2023. — 662 с. — ISBN 978-5-93700-178-8. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/348089> (дата обращения: 23.12.2024).
9. Маркин, А. В. СУБД PostgreSQL. Основы SQL: учебное пособие. — Алматы, Москва: EDP Hub (Идипи Хаб), Ай Пи Ар Медиа, 2024. — 658 с. —

ISBN 978-5-4497-3642-0. — Текст: электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/143170.html> (дата обращения: 23.12.2024).

10. Как выбрать PostgreSQL для вашего проекта [Электронный ресурс]. — Режим доступа: <https://habr.com/ru/articles/170045/> (дата обращения: 23.12.2024).

11. How to design and build SQL database in PostgreSQL [Электронный ресурс]. — Режим доступа: <https://www.dbvis.com/thetable/how-to-design-and-build-sql-database-in-postgres/> (дата обращения: 23.12.2024).

12. CREATE TRIGGER - PostgreSQL Documentation [Электронный ресурс]. — Режим доступа: <https://www.postgresql.org/docs/current/sql-createtrigger.html> (дата обращения: 23.12.2024).

13. Как выбрать PostgreSQL для вашего проекта [Электронный ресурс]. — Режим доступа: <https://habr.com/ru/companies/otus/articles/857396/> (дата обращения: 23.12.2024).

14. Новиков, Б. А., Горшкова, Е. А., Графеева, Н. Г. и др. Основы технологий баз данных: учебное пособие. — 2-е изд. — М.: ДМК Пресс, 2020. — 582 с. — ISBN 978-5-97060-841-8.

15. Триггеры в PostgreSQL [Электронный ресурс]. — Режим доступа: https://ciu.nstu.ru/kaf/persons/1914/study/baz_dannh/trigger (дата обращения: 23.12.2024).

16. Презентация по теме триггеров в PostgreSQL [Электронный ресурс]. — Режим доступа: <https://ppt-online.org/1108687> (дата обращения: 23.12.2024).

17. Как выбрать PostgreSQL для вашего проекта [Электронный ресурс]. — Режим доступа: <https://habr.com/ru/companies/timeweb/articles/661771/> (дата обращения: 23.12.2024).

18. Основы PostgreSQL. Роли и привилегии [Электронный ресурс]. — Режим доступа: https://sql-ex.com/blogs/?/Osnovy_PostgreSQL_rol_i_privilegii.html&ysclid=m50svyo8tb245567433 (дата обращения: 23.12.2024).

19. PostgreSQL Indexes - PostgreSQL Documentation [Электронный ресурс].
– Режим доступа: <https://www.postgresql.org/docs/current/indexes.html> (дата обращения: 23.12.2024).