

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ**

**УНИВЕРСИТЕТ»**

**(СПбГЭУ)**

Факультет информатики и прикладной математики

Кафедра прикладной математики и экономико-математических методов

**КУРСОВАЯ РАБОТА**

по дисциплине:

«**Системы управления базами данных**»

Тема: «Разработка базы данных для сети кинотеатров»

Направление: 01.03.02 Прикладная математика и информатика

Направленность: Прикладная математика и информатика в экономике и управлении

Обучающийся: Бронников Егор Игоревич

Группа: ПМ-1901 Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил: Иванова Виктория Валерьевна

Должность: доцент

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата: \_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2021

СОДЕРЖАНИЕ

[1. Характеристика предметной области 3](#_Toc90813393)

[1.1. Описание предметной области 3](#_Toc90813394)

[1.2. Определение проблемы, для которой будет создаваться продукт 4](#_Toc90813395)

[2. Разработка технического предложения 5](#_Toc90813396)

[2.1. Формулировка видения и назначения 5](#_Toc90813397)

[2.2. Бизнес-требования 5](#_Toc90813398)

[2.3. Концепция продукта 6](#_Toc90813399)

[2.4. Анализ требований пользователя с помощью User Stories 6](#_Toc90813400)

[3. Проектирование информационного обеспечения 8](#_Toc90813401)

[3.1. Анализ информационных потоков 8](#_Toc90813402)

[3.2. Определение информационных объектов 8](#_Toc90813403)

[3.3. Нормализация 11](#_Toc90813404)

[3.4. Информационно-логическая модель 15](#_Toc90813405)

[4. Реализация в среде mysql 17](#_Toc90813406)

[4.1. Формирование и заполнение таблиц 17](#_Toc90813407)

[4.2. Разработка запросов 21](#_Toc90813408)

[4.2.1. Ограничение на места в зале 21](#_Toc90813409)

[4.2.2. Список премьер 22](#_Toc90813410)

[4.2.3. Список самых коммерчески успешных фильмов в определённый период времени в данной сети кинотеатров 22](#_Toc90813411)

[4.2.4. Список фильмов по жанрам 23](#_Toc90813412)

[4.2.5. Ограничение на сеансы 23](#_Toc90813413)

[4.2.6. Подсчёт сборов фильма 23](#_Toc90813414)

[4.2.7. Анализ загруженности залов 25](#_Toc90813415)

[4.2.8. Выбор сеансов по критериям 27](#_Toc90813416)

[Заключение 28](#_Toc90813417)

[Список ЛИТЕРАТУРЫ 29](#_Toc90813418)

# **Характеристика предметной области**

## **Описание предметной области**

Существует множество кинотеатров в разных городах и странах. В более маленьких населённых пунктах зачастую всего пара кинотеатров, но в более крупных и густонаселённых городах существуют целые сети кинотеатров, которые создаются с целью удовлетворения спроса населения на данную услугу. В связи с этим, составление расписаний, финансовый учёт, продажа билетов в кассах и бронирование, аналитика и многое другое становятся достаточно громоздкими процессами, которые влияют на финансовую составляющую такой сети, поэтому необходимо правильно организовывать учёт компании.

Итак, допустим, что существует некоторая сеть кинотеатров в городе Санкт-Петербург. У этой сети есть несколько кинотеатров, в каждом кинотеатре есть касса и несколько залов. Залы можно разделить на два типа: обычные (regular) или VIP залы.

В кинотеатре имеются несколько фильмов, которые идут сейчас в прокате. Пользователь может просмотреть на сайте или в мобильном приложении информацию по данному фильму, а именно: название фильма, продолжительность (в минутах), возрастной рейтинг, жанр, описание фильма и рейтинг.

Клиент также может заказать несколько мест, купить билет на сайте, в приложении или в кассе кинотеатра. То есть он должен выбрать конкретный кинотеатр, выбрать зал и выбрать какое-то количество мест, также ему необходимо выбрать дату и время сеанса. После покупки, пользователю отдают билеты в электронном или печатном виде и также отдают чек. В билете указана дата сеанса, время сеанса, фильм, кинотеатр, зал и место. В чеке указана дата и стоимость заказа.

Также если пользователь покупает через мобильное приложение или сайт, то у него должен быть зарегистрирован аккаунт, в котором он должен заполнить ФИО, электронную почту, номер телефона и создать пароль.

## **Определение проблемы, для которой будет создаваться продукт**

Проблема, которую решает данная система состоит в том, чтобы автоматизировать учёт в кинотеатрах. Также данная система поможет избежать случаев, когда происходит путаница с покупкой билетов.

Мобильное приложение и сайт дадут клиентам дополнительную возможность делать заказы онлайн, что возможно повысит удобство для некоторых групп потребителей.

Данные полученные из системы можно использовать для дальнейшего анализа и таким образом можно оптимизировать некоторые сеансы, чтобы получить наибольшую прибыль и использовать набор фильмов эффективно при составлении расписания.

# **Разработка технического предложения**

## **Формулировка видения и назначения**

Рассматриваемой сети кинотеатров требуется автоматизация действий по обслуживанию клиентов с целью роста качества услуг, повышения эффективности работы персонала и получения максимально возможной прибыли.

Основное назначение разрабатываемой базы данных – обеспечение быстрого и удобного доступа к информации о расписании и о заказах. База данных позволит отображать всю информацию о загруженности залов и загруженности кинотеатров, а также о заказах и спросе на конкретный фильм, что поможет оптимально составлять расписание, получать максимально возможную прибыль и позволит увеличить пользовательский комфорт при использовании услуг данной сети кинотеатров.

Система будет направлена на сокращение влияния человеческого фактора и повышение эффективности работы компании. Она будет предлагать сбор, обработку, изменение, хранение и сравнение данных. Возможности системы должны помочь уменьшить издержки обслуживания клиентов и помогут при дальнейшей аналитике.

## **Бизнес-требования**

Целью проекта является формирование единой информационной среды компании, охватывающей все процессы жизненного цикла обслуживания клиентов и реализации заказов. Будут решены следующие задачи:

1. повышение удовлетворённости клиентов;
2. улучшение финансового показателя компании;
3. составление оптимального расписания с учётом спроса клиентов;
4. повышение качества контроля при заказе билетов;

С помощью такой системы можно будет определять автоматически загруженность кинотеатров и залов, смотреть на потребительский спрос конкретных фильмов. Предложенная система значительно облегчит внутренние организационные процессы и приведёт к оптимальному функционированию рассматриваемую сеть кинотеатров.

## **Концепция продукта**

Продукт разрабатывается для кассиров, руководителей кинотеатров и директоров, аналитиков и клиентов. Рассмотрим, каким требованиям будет отвечать разрабатываемый продукт (Таблица 1).

Таблица 1 – Требования пользователей к разрабатываемой системе

|  |  |
| --- | --- |
| Пользователь | Потребность |
| Кассир | Автоматизация оформления заказа |
| Снижение количества ошибок в работе |
| Быстрый доступ к данным по запросам клиентов |
| Повышение качества работы с клиентом |
| Руководители | Быстрый доступ к данным |
| Контроль за выполнением работы сотрудников |
| Принятие решений по поводу составления расписания |
| Аналитики | Быстрый доступ к данным |
| Составление оптимального плана расписаний |
| Клиенты | Автоматизация оформления заказа |
| Снижение количества ошибок в работе |
| Повышение качества обслуживания |

Таким образом, данная система оптимизирует работу кассиров, руководителей, аналитиков и клиентов, обеспечит им комфортную рабочую среду.

## **Анализ требований пользователя с помощью User Stories**

**Кто:** Кассир

**Что:** Оформление заказа

**Зачем:** Чтобы быстро и безошибочно (при участии клиента) оформлять заказ

**Кто:** Кассир

**Что:** Решение вопросов

**Зачем:** Чтобы быстро и оперативно получать информацию о заказах и клиентах в случае возникновения вопросов

**Кто:** Руководитель

**Что:** Контроль работы сотрудников

**Зачем:** Чтобы контролировать работу сотрудников и решать вопросы при их возникновении

**Кто:** Аналитик

**Что:** Составление расписания

**Зачем:** Чтобы быстро получать свежую информацию о заказах, наборе фильмов, загруженности кинотеатров, загруженности залов и других показателях

**Кто:** Клиент

**Что:** Оформление заказа

**Зачем:** Чтобы удобно, оперативно и безошибочно осуществлять бронирование или покупку билетов

**Кто:** Клиент

**Что:** Просмотр информации о фильме

**Зачем:** Чтобы можно было ознакомиться с кратким описание о фильме перед покупкой билета

# **Проектирование информационного обеспечения**

## **Анализ информационных потоков**

В проектируемой базе данных в качестве входной информации рассматриваются, набор фильмов и информация об этих фильмах, информация о кинотеатрах и залах, которые имеются в распоряжении сети, расписание сеансов.

Ко внутренней информации относится информация о заказах и данные о зарегистрировавшихся клиентах. С помощью данной информации мы можем анализировать предпочтения клиентов, также можно анализировать загруженность кинотеатров и залов, востребованность конкретных фильмов. Эти данные заполняют кассиры, далее она переходит к аналитикам, а решения принимают уже руководители.

В качестве выходных данных можно рассматривать отчеты от аналитиков, возможно более предпочтительное расписание сеансов с учётом спроса клиентов, информацию о качестве обслуживания клиентов и отзыв (обратную связь) от клиента.

## **Определение информационных объектов**

В реализуемой базе данных хранится и используется информация о пользователях: ID пользователя, ФИО пользователя, номер телефона пользователя, почта (email address), пароль от личного кабинета, информация о заказах. Таким образом, была сформирована сущность клиент (Таблица 2).

Таблица 2 – Сущность клиент

|  |
| --- |
| **Клиент** |
| ID Клиента |
| ФИО Клиента |
| Номер телефона |
| Email address |
| Пароль |
| Заказы |

Имеется набор фильмов: ID фильма, название фильма, продолжительность (минутах), возрастной рейтинг, жанр, описание фильма, рейтинг фильма, дата начала проката, дата окончания проката. В результате, была сформирована необходимая сущность фильмов (Таблица 3).

Таблица 3 – Сущность фильм

|  |
| --- |
| **Фильм** |
| ID Фильма |
| Название |
| Продолжительность |
| Возрастной рейтинг |
| Жанры |
| Описание фильма |
| Рейтинг фильма |
| Начало проката |
| Конец проката |

Также в базе имеется информация о кинотеатрах (помещениях): ID кинотеатра, название, адрес, залы. В результате, была сформирована необходимая сущность кинотеатр (Таблица 4).

Таблица 4 – Сущность кинотеатр

|  |
| --- |
| **Кинотеатр** |
| ID Кинотеатра |
| Название |
| Адрес |
| Залы |

В каждом кинотеатре у нас также имеются залы: ID зала, номер зала (локальный для каждого кинотеатра), количество мест, тип зала, кинотеатр. В результате, была сформирована необходимая сущность зал (Таблица 5).

Таблица 5 – Сущность зал

|  |
| --- |
| **Зал** |
| ID Зала |
| Номер зала |
| Кинотеатр |
| Кол-во мест |
| Тип зала |

И ещё у нас есть сеансы (расписание): ID сеанса, ID фильма, дата сеанса, время сеанса, цена билета, ID зала. В результате, была сформирована необходимая сущность сеанс (Таблица 6).

Таблица 6 – Сущность сеанс

|  |
| --- |
| **Сеанс** |
| ID Сеанса |
| ID Фильма |
| ID Зала |
| Цена билета |
| Дата сеанса |
| Время сеанса |

## **Нормализация**

Нормализация — процесс уменьшения избыточности информации в таблицах реляционной БД и, как следствие, построения оптимальной структуры таблиц и связей.

Рассмотрим сущность клиент (Таблица 7) и приведем её к третьей нормальной форме.

Таблица 7 – Сущность клиент

|  |
| --- |
| **Клиент** |
| ID Клиента |
| ФИО Клиента |
| Номер телефона |
| Email address |
| Пароль |
| Заказы |

Сущность находится в первой нормальной форме, так как для клиента может выполняться несколько заказов одновременно. В результате нормализации были получены следующие сущности (Таблица 8, Таблица 9, Таблица 10).

Таблица 8 – Нормализованная сущность клиент

|  |
| --- |
| **Клиент** |
| ID Клиента |
| ФИО Клиента |
| Номер телефона |
| Email address |
| Пароль |

Таблица 9 – Нормализованная сущность заказ

|  |
| --- |
| **Заказ** |
| ID заказа |
| ID клиента |
| ID сеанса |

Таблица 10 – Нормализованная сущность спецификация заказа

|  |
| --- |
| **Спец. заказа** |
| ID заказа |
| Места |

Далее рассмотрим сущность фильм (Таблица 11) и приведем её к третьей нормальной форме.

Таблица 11 – Сущность фильм

|  |
| --- |
| **Фильм** |
| ID Фильма |
| Название |
| Продолжительность |
| Возрастной рейтинг |
| Жанры |
| Описание фильма |
| Рейтинг фильма |
| Начало проката |
| Конец проката |

Сущность находится в первой нормальной форме, так как фильм может принадлежать сразу к нескольким жанрам. В результате нормализации были получены следующие сущности (Таблица 12, Таблица 13, Таблица 14).

Таблица 12 – Нормализованная сущность фильм

|  |
| --- |
| **Фильм** |
| ID Фильма |
| Название |
| Продолжительность |
| Возрастной рейтинг |
| Описание фильма |
| Рейтинг фильма |
| Начало проката |
| Конец проката |

Таблица 13 – Нормализованная сущность жанр

|  |
| --- |
| **Жанр** |
| ID Жанра |
| Название |

Таблица 14 – Нормализованная сущность фильм-жанр

|  |
| --- |
| **Фильм-жанр** |
| ID Фильма |
| ID Жанра |

На следующем этапе рассмотрим сущности кинотеатр и зал, (Таблица 15, Таблица 16) приведем её к третьей нормальной форме.

Таблица 15 – Сущность кинотеатр

|  |
| --- |
| **Кинотеатр** |
| ID Кинотеатра |
| Название |
| Адрес |
| Залы |

Таблица 16 – Сущность зал

|  |
| --- |
| **Зал** |
| ID Зала |
| Номер зала |
| Кинотеатр |
| Кол-во мест |
| Тип зала |

Сущность кинотеатр находится в первой нормальной форме, так как один кинотеатр может содержать сразу несколько залов. В результате нормализации были получены следующие сущности (Таблица 17, Таблица 18, Таблица 19).

Таблица 17 – Нормализованная сущность кинотеатр

|  |
| --- |
| **Кинотеатр** |
| ID Кинотеатра |
| Название |
| Адрес |

Таблица 18 – Нормализованная сущность зал

|  |
| --- |
| **Зал** |
| ID Зала |
| Номер зала |
| Кол-во мест |
| Тип зала |

Таблица 19 – Нормализованная сущность кинотеатр-зал

|  |
| --- |
| **Кинотеатр-зал** |
| ID Кинотеатра |
| ID Зала |

Осталась сущность сеанс, но она уже находится в третьей нормальной форме. (Таблица 20)

Таблица 20 – Сущность сеанс

|  |
| --- |
| **Сеанс** |
| ID Сеанса |
| ID Фильма |
| ID Зала |
| Цена билета |
| Дата сеанса |
| Время сеанса |

## **Информационно-логическая модель**

Информационно-логическая модель отображает данные предметной области в виде совокупности информационных объектов и связей между ними. Эта модель представляет данные, подлежащие хранению в базе данных. По нормализованным сущностям построим информационно-логическую модель или схему данных (Рисунок 1).

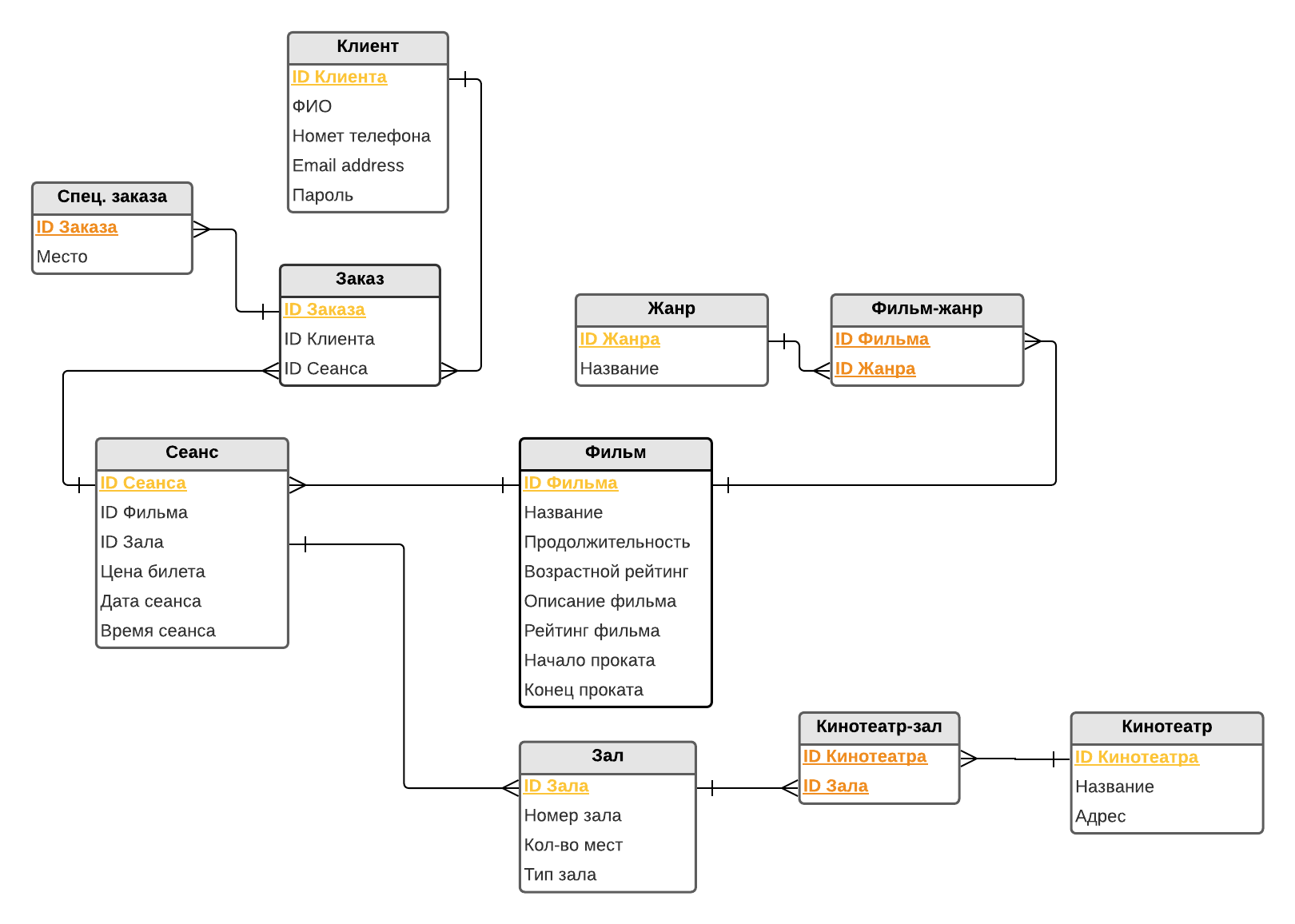


Рисунок 1 – Информационно-логическая модель предметной области

Таким образом, построенная информационно-логическая модель отражает сущности базы данных в третьей нормальной форме и связи между ними.

# **Реализация в среде mysql**

## **Формирование и заполнение таблиц**

Необходимо реализовать в среде MySQL таблицы, которые впоследствии будут хранить данные о сети кинотеатров. Таблицы были сформированы с помощью запросов, представленных в Таблице 21.

Таблица 21 – Запросы для создания соответствующих таблиц

|  |  |
| --- | --- |
| **CREATE TABLE** *client\_* **(**  *IDclient***int primary key,**  *lastname***varchar(60),**  *firstname***varchar(60),**  *middlename***varchar(60),**  *phone***varchar(15),**  *email***varchar(40),**  *password\_***varchar(100));** | Создание таблицы «Клиент» |
| **CREATE TABLE** *film* **(**  *IDfilm* **int primary key,**  *name\_* **varchar(100),**  *duration* **int,**  *age\_rating* **varchar(10),**  *description*\_ **longtext,**  *film\_rating* **float,**  *start\_distribution* **date,**  *end\_distribution* **date);** | Создание таблицы «Фильм» |
| **CREATE TABLE** *genre* **(**  *IDgenre* **int primary key,**  *name\_* **varchar(50));** | Создание таблицы «Жанр» |
| **CREATE TABLE** *cinema* **(**  *IDcinema* **int primary key,**  *name\_* **varchar(100),**  *address* **varchar(255));** | Создание таблицы «Кинотеатр» |
| **CREATE TABLE** *hall* **(**  *IDhall* **int primary key,**  *number\_* **float,**  *num\_seats* **int,**  *type* **varchar(15));** | Создание таблицы «Зал» |
| **CREATE TABLE** *cinema\_hall* **(**  *IDcinema* **int,**  *IDhall* **int,**  **primary key(***IDcinema***,** *IDhall***),**  **constraint** *fr1* **foreign key** *(IDcinema)* **references** *cinema(IDcinema)***,**  **constraint** *fr2* **foreign key** *(IDhall)* **references** *hall(IDhall)***);** | Создание таблицы «Кинотеатр-зал» |
| **CREATE TABLE** *film\_genre* **(**  *IDfilm* **int,**  *IDgenre* **int,**  **primary key(***IDfilm***,** *IDgenre***),**  **constraint** *fr3* **foreign key** *(IDfilm)* **references** *film(IDfilm)***,**  **constraint** *fr4* **foreign key** *(IDgenre)* **references** *genre(IDgenre)***);** | Создание таблицы «Фильм-жанр» |
| **CREATE TABLE** *session***\_ (**  *IDsession* **int primary key,**  *IDfilm* **int,**  *IDhall* **int,**  *price* **int,**  *date\_* **date,**  *time\_* **time,**  **constraint** *fr5* **foreign key** *(IDfilm)* **references** *film(IDfilm)***,**  **constraint** *fr6* **foreign key** *(IDhall)* **references** *hall(IDhall)***);** | Создание таблицы «Сеанс» |
| **CREATE TABLE** *order\_* **(**  *IDorder* **int primary key,**  *IDclient* **int,**  *IDsession* **int,**  **constraint** *fr7* **foreign key** *(IDclient)* **references** *client\_(IDclient)***,**  **constraint** *fr8* **foreign key** *(IDsession)* **references** *session\_(IDsession)***);** | Создание таблицы «Заказ» |
| **CREATE TABLE** *specorder* **(**  *IDorder* **int,**  *seat* **int,**  **primary key(***IDorder***,** *seat***),**  **constraint** *fr9* **foreign key** *(IDorder)* **references** *order\_(IDorder)***);** | Создание таблицы «Спецификация заказа» |

При создании таблиц учитывались типы хранимых данных, а также указывались первичные и внешние ключи.

При заполнении использовался MS Excel и конвертирование таблиц в *.csv* формат.

Таким образом, была получена следующая база данных (фрагменты базы представленные на Рисунках 2 – 11).

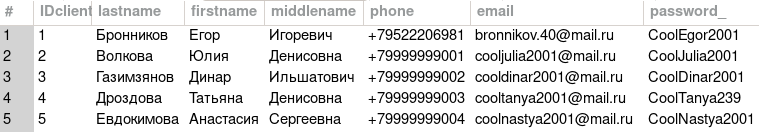


Рисунок 2 – Фрагмент из таблицы клиентов

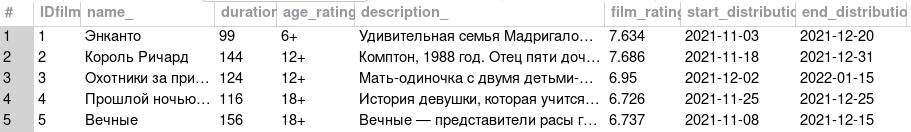


Рисунок 3 – Фрагмент из таблицы фильмов

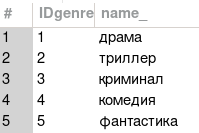


Рисунок 4 – Фрагмент из таблицы жанров

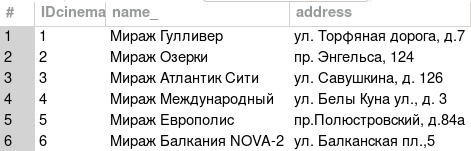


Рисунок 5 – Фрагмент из таблицы кинотеатров

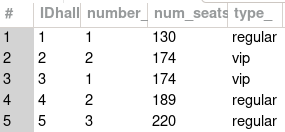


Рисунок 6 – Фрагмент из таблицы залов

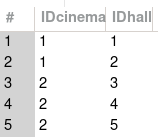


Рисунок 7 – Фрагмент из таблицы кинотеатров-залов

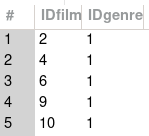


Рисунок 8 – Фрагмент из таблицы фильмов-жанров

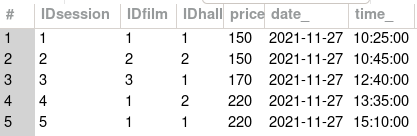


Рисунок 9 – Фрагмент из таблицы сеансов

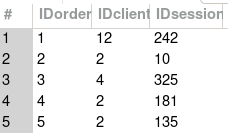


Рисунок 10 – Фрагмент из таблицы сеансов

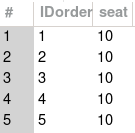


Рисунок 11 – Фрагмент из таблицы спецификации заказов

Таким образом, была реализована спроектированная база данных, в которой хранится информация, отвечающая требованиям предметной области.

## **Разработка запросов**

Необходимо сформулировать запросы, которые будут отвечать требованиям предметной области и описанным User Stories. В компании имеется четыре вида пользователей базой данных: кассир, клиент, руководитель, аналитик. Запросы были написаны для каждого вида пользователя, так как их цели использования отличаются.

### **Ограничение на места в зале**

Бизнес-правила запрещают нам выдавать клиентам места, которые заняты в текущем сеансе, также для безопасности нужно сделать проверку, что мы не можем ввести номер места превышающий последнее место в зале. Для этого был создан триггер в таблице *specorder* (1) и были созданы вспомогательные функции *max\_seats* (2), которая находит последнее место в зале по номеру заказа (*IDorder*), и функция *order\_session* (3), которая находит номер сеанса (*IDsession*) по номеру заказа (*IDorder*).

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` TRIGGER `specorder\_BEFORE\_INSERT` BEFORE INSERT ON `specorder` FOR EACH ROW BEGIN**  **declare NewSeat condition for sqlstate "45000";**  **if new.seat > (select cinema.max\_seats(new.IDorder)) then**  **signal NewSeat set message\_text = "Это значение превышает допустимое количество мест в зале!";**  **elseif new.seat in (select seat from order\_ join specorder on order\_.IDorder=specorder.IDorder where IDsession=(select cinema.order\_session(new.IDorder))) then**  **signal NewSeat set message\_text = "Это место уже занято!";**  **end if;**  **END** | (1) |

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` FUNCTION `max\_seats`(IDord int) RETURNS int**  **READS SQL DATA**  **BEGIN**  **declare num int;**  **set num = (select distinct num\_seats from hall join**  **(select order\_.IDorder, session\_.IDsession, IDhall from order\_ join session\_ on order\_.IDsession=session\_.IDsession) as q1**  **on q1.IDhall=hall.IDhall where q1.IDorder=IDord);**  **RETURN num;**  **END** | (2) |

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` FUNCTION `order\_session`(IDord int) RETURNS int**  **READS SQL DATA**  **BEGIN**  **declare ses int;**  **set ses = (select IDsession from order\_ where order\_.IDorder=IDord);**  **RETURN ses;**  **END** | (3) |

Данный триггер может быть полезен для кассиров и клиентов при оформлении заказа.

### **Список премьер**

Клиентам хотелось бы иметь возможность просмотреть список премьер. Для этого было создано представление – *premiere*. (4)

|  |  |
| --- | --- |
| **CREATE**  **ALGORITHM = UNDEFINED**  **DEFINER = `root`@`%`**  **SQL SECURITY DEFINER**  **VIEW `premiere` AS**  **SELECT**  **`film`.`IDfilm` AS `IDfilm`,**  **`film`.`name\_` AS `name\_`,**  **`film`.`start\_distribution` AS `start\_distribution`,**  **(TO\_DAYS(`film`.`start\_distribution`) - TO\_DAYS(CURDATE())) AS `daysToPremier`**  **FROM**  **`film`**  **WHERE**  **(CURDATE() < `film`.`start\_distribution`)** | (4) |

### **Список самых коммерчески успешных фильмов в определённый период времени в данной сети кинотеатров**

Для руководства и аналитиков было бы полезно отслеживать список популярных фильмов, это было бы применимо при составлении нового расписания. Для этого была создана процедура *top\_films* (5), которой на вход поступает начальная дата и конечная дата интересующего временного периода.

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` PROCEDURE `top\_films`(in startDate date, endDate date)**  **BEGIN**  **select IDfilm, name\_, SUM(sessionSeats\*price) as ticketsSold from**  **(select session\_.IDsession, session\_.IDfilm, price, date\_, name\_ from session\_ join film on session\_.IDfilm=film.IDfilm) as q1 left join**  **(select IDsession, SUM(countSeats) as sessionSeats from order\_countSeats group by IDsession) as q2**  **on q2.IDsession=q1.IDsession where date\_ between startDate and endDate group by IDfilm order by ticketsSold desc limit 5;**  **END** | (5) |

### **Список фильмов по жанрам**

Также у клиентов есть возможность просмотреть список фильмов по интересующим их жанру, которые идут сейчас в прокате. Для этого была создана процедура *film\_by\_genre* (6), которой на вход поступает название интересующего жанра.

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` PROCEDURE `film\_by\_genre`(in genre\_in text)**  **BEGIN**  **select film.IDfilm, name\_, name\_genre from film join**  **(select IDfilm, name\_ as name\_genre from genre join film\_genre on genre.IDgenre=film\_genre.IDgenre where name\_=genre\_in) as q1**  **on film.IDfilm=q1.IDfilm where current\_date() between start\_distribution and end\_distribution;**  **END** | (6) |

### **Ограничение на сеансы**

В соответствии с нашими бизнес-правилами, мы не можем составить сеанс с фильмом, который пока ещё не вышел и находится в премьере. Для того чтобы у аналитика не возникало ошибок во время заполнения расписания, был сделан триггер в таблице *session\_* (7).

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` TRIGGER `session\_\_BEFORE\_INSERT` BEFORE INSERT ON `session\_` FOR EACH ROW BEGIN**  **declare NewSession condition for sqlstate "45000";**  **if new.IDfilm in (select IDfilm from premiere) then**  **signal NewSession set message\_text = "Этот фильм ещё не вышел!";**  **end if;**  **END** | (7) |

### **Подсчёт сборов фильма**

Руководителям и аналитикам хотелось бы просмотреть сколько рублей собрал конкретный фильм и нужно сделать так, чтобы эта информация была постоянно обновляемой. Для этого была создана дополнительная колонка в таблице *film* – *gross*, в которой у нас и будет храниться информация о сборах фильма. Также была реализована процедура *gross* (8), которая обновляет соответствующую колонку в таблице *film*. В процедуре используются два вспомогательных представления – *view\_gross* (9) и *view\_gross2* (10), но при реализации *view\_gross* было решено создать ещё одно промежуточное представление – *order\_countSeats* (11), которое подсчитывает количество купленных мест по номеру заказа. Ещё чтобы эта информация постоянно обновлялась, нужно в таблице *specorder* добавить два триггера – на добавление (12) и на удаление (13).

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` PROCEDURE `gross`()**  **BEGIN**  **declare done int default 0;**  **declare per1 int;**  **declare per2 int;**  **declare cur cursor for select IDfilm, gross from view\_gross2;**  **declare exit handler for not found set done=1;**  **open cur;**  **while done=0 do**  **fetch cur into per1, per2;**  **update film set gross=per2 where IDfilm=per1;**  **end while;**  **close cur;**  **END** | (8) |

|  |  |
| --- | --- |
| **CREATE**  **ALGORITHM = UNDEFINED**  **DEFINER = `root`@`%`**  **SQL SECURITY DEFINER**  **VIEW `view\_gross` AS**  **SELECT**  **`session\_`.`IDfilm` AS `IDfilm`,**  **SUM((`q2`.`sessionSeats` \* `session\_`.`price`)) AS `gross`**  **FROM**  **(`session\_`**  **LEFT JOIN (SELECT**  **`order\_countSeats`.`IDsession` AS `IDsession`,**  **SUM(`order\_countSeats`.`countSeats`) AS `sessionSeats`**  **FROM**  **`order\_countSeats`**  **GROUP BY `order\_countSeats`.`IDsession`) `q2` ON ((`q2`.`IDsession` = `session\_`.`IDsession`)))**  **GROUP BY `session\_`.`IDfilm`** | (9) |

|  |  |
| --- | --- |
| **CREATE**  **ALGORITHM = UNDEFINED**  **DEFINER = `root`@`%`**  **SQL SECURITY DEFINER**  **VIEW `view\_gross2` AS**  **SELECT**  **`film`.`IDfilm` AS `IDfilm`,**  **IF((`view\_gross`.`gross` IS NULL),**  **0,**  **`view\_gross`.`gross`) AS `gross`**  **FROM**  **(`film`**  **LEFT JOIN `view\_gross` ON ((`film`.`IDfilm` = `view\_gross`.`IDfilm`)))** | (10) |

|  |  |
| --- | --- |
| **CREATE**  **ALGORITHM = UNDEFINED**  **DEFINER = `root`@`%`**  **SQL SECURITY DEFINER**  **VIEW `order\_countSeats` AS**  **SELECT**  **`order\_`.`IDorder` AS `IDorder`,**  **`order\_`.`IDsession` AS `IDsession`,**  **COUNT(`specorder`.`seat`) AS `countSeats`**  **FROM**  **(`order\_`**  **JOIN `specorder` ON ((`order\_`.`IDorder` = `specorder`.`IDorder`)))**  **GROUP BY `order\_`.`IDorder` , `order\_`.`IDsession`** | (11) |

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` TRIGGER `specorder\_AFTER\_INSERT` AFTER INSERT ON `specorder` FOR EACH ROW BEGIN**  **call cinema.gross();**  **END** | (12) |

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` TRIGGER `specorder\_AFTER\_DELETE` AFTER DELETE ON `specorder` FOR EACH ROW BEGIN**  **call cinema.gross();**  **END** | (13) |

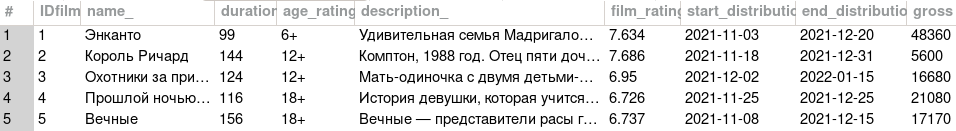


Рисунок 12 – Фрагмент из таблицы фильмов

### **Анализ загруженности залов**

Для руководителей и аналитиков необходим расчёт загруженности залов (%), эта информация будет полезна при составлении расписания и при решении возможного закрытия кинотеатра, это особенно актуально во времена пандемии. Для этого была создана дополнительная колонка *workload* в таблице *hall*. Также была реализована процедура *workload* (14), которая обновляет соответствующую колонку в таблице. При создании процедуры было создано вспомогательное представление *view\_workload* (15). Ещё, для автоматического обновления колонки создано два триггера в таблице *specorder* – на добавление (16) и на удаление (17).

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` PROCEDURE `workload`()**  **BEGIN**  **declare done bool default false;**  **declare per1 int;**  **declare per2 float;**  **declare cur cursor for select IDhall, workload from view\_workload;**  **declare continue handler for not found set done=true;**  **open cur;**  **my\_loop: loop**  **fetch cur into per1, per2;**  **if done then leave my\_loop;**  **end if;**  **update hall set workload=per2 where IDhall=per1;**  **end loop;**  **close cur;**  **END** | (14) |

|  |  |
| --- | --- |
| **CREATE**  **ALGORITHM = UNDEFINED**  **DEFINER = `root`@`%`**  **SQL SECURITY DEFINER**  **VIEW `view\_workload` AS**  **SELECT**  **`q1`.`IDhall` AS `IDhall`,**  **((`q2`.`occupiedPlaces` / `q1`.`countSession`) \* 100) AS `workload`**  **FROM**  **((SELECT**  **`session\_`.`IDhall` AS `IDhall`,**  **(COUNT(`session\_`.`IDhall`) \* `hall`.`num\_seats`) AS `countSession`**  **FROM**  **(`session\_`**  **JOIN `hall` ON ((`session\_`.`IDhall` = `hall`.`IDhall`)))**  **GROUP BY `session\_`.`IDhall`) `q1`**  **JOIN (SELECT**  **`session\_`.`IDhall` AS `IDhall`,**  **SUM(`order\_countSeats`.`countSeats`) AS `occupiedPlaces`**  **FROM**  **(`session\_`**  **JOIN `order\_countSeats` ON ((`session\_`.`IDsession` = `order\_countSeats`.`IDsession`)))**  **GROUP BY `session\_`.`IDhall`) `q2` ON ((`q1`.`IDhall` = `q2`.`IDhall`)))** | (15) |

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` TRIGGER `specorder\_AFTER\_INSERT` AFTER INSERT ON `specorder` FOR EACH ROW BEGIN**  **call cinema.gross();**  **call cinema.workload();**  **END** | (16) |

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` TRIGGER `specorder\_AFTER\_DELETE` AFTER DELETE ON `specorder` FOR EACH ROW BEGIN**  **call cinema.gross();**  **call cinema.workload();**  **END** | (17) |

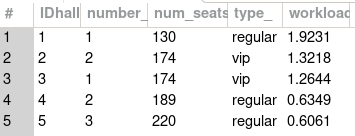


Рисунок 13 – Фрагмент из таблицы залов

### **Выбор сеансов по критериям**

Клиенту и кассиру будет удобно отбирать сеансы по нескольким критериям: номер кинотеатра (*IDcinema*), с какой даты интересуют сеансы (*date\_*), с какого времени (*time\_*) и какой номер жанра фильма (*IDgenre*). Для этого была создана процедура *sessions\_by\_catigories* (18), которая по входным критериям находит нужные сеансы. При реализации приложения, название кинотеатра и название жанра можно конвертировать в ID и использовать эту процедуру. Также допускается использование нескольких сценариев для запуска данной процедуры, когда не все входные параметры заданы.

|  |  |
| --- | --- |
| **CREATE DEFINER=`root`@`%` PROCEDURE `sessions\_by\_categories`(in INcinema int, INdate date, INtime time, INgenre int)**  **BEGIN**  **select IDsession, session\_.IDfilm, IDcinema, IDgenre, session\_.IDhall, price, date\_, time\_ from**  **(session\_ join film\_genre on session\_.IDfilm=film\_genre.IDgenre) join**  **(select IDhall, cinema.IDcinema, name\_ from cinema join cinema\_hall on cinema.IDcinema=cinema\_hall.IDcinema**  **where if(INcinema is null, cinema.IDcinema, cinema.IDcinema=INcinema)) as q1**  **on session\_.IDhall=q1.IDhall where**  **if(INtime is null, time\_, time\_ > INtime) and**  **if(INdate is null, date\_, date\_ > INdate) and**  **if(INgenre is null, IDgenre, IDgenre=INgenre);**  **END** | (18) |

# **Заключение**

В результате работы была реализована система, которая позволила хранить персональные данные клиентов; информацию о фильмах; информацию о кинотеатрах и расписание в единой связанной системе – базе данных.

Такая система поможет упорядочить работу сети кинотеатров и сделать её более эффективной, также в базе есть начальные процедуры, которые помогут в дальнейшей аналитике и как следствии получении большей прибыли. Разработанная база данных обеспечит быстрый и удобный доступ к информации о заказах и расписании.

База данных позволит отображать всю информацию о загруженности залов, может вывести сборы фильмов, что может помочь при аналитике и может послужить опорой при принятии решений руководством. Также база контролирует ввод данных, что поможет снизить человеческий фактор и контролировать деятельность персонала.

# **Список ЛИТЕРАТУРЫ**

1. Гарбер, М. Понимание SQL: учеб. пособие, 2019
2. MySQL 8.0 Reference Manual – Режим доступа: <https://dev.mysql.com/doc/refman/8.0/en/>