**Содержание**

ВВЕДЕНИЕ

1 Концептуальное описание предметной области

2 Характеристики ИС

3 Анализ существующих решений для предметной области

4 Разработка общей структуры ИС

5 Разработка серверной части информационной системы

5.1 Инфологическое проектирование БД

5.2 Даталогическое проектирование БД

5.3 Программирование объектов БД

6 Разработка клиентского приложения

6.1 Выбор программных компонентов клиентской части

6.2 Разработка интерфейса пользователя

6.2.1 Разработка форм

6.3 Разработка сценария инсталляции клиентской программы

7 Разработка программной документации

8 Тестирование ИС

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

ПРИЛОЖЕНИЕ А: сценарий создания объектов БД.

ПРИЛОЖЕНИЕ Б: сценарий заполнения таблиц БД.

ПРИЛОЖЕНИЕ В: исходный текст клиентской программы.

ПРИЛОЖЕНИЕ Г: сценарий инсталляции программы.

**Введение**

В современном мире с повсеместном развитием способов получения информации для людей острым встает вопрос затраты времени. Поэтому разрабатываются системы, позволяющие быстро получить необходимую информацию. Сейчас существуют крупные системы, обладающие большой скоростью, значительным объемом поисковых данных и их доступностью.

В данной курсовой работе, призванной ознакомить студентов с проектированием подобной системы, но куда более меньших размеров, была разработана система для кинотеатра, которая позволяет управлять основными объектами. Использование приложения для кинотеатра позволяет узнавать о расписаниях и покупать билеты фильмы там, где удобно зрителю. То есть нет необходимости в посещении кинотеатра, кроме как для просмотра киноленты.

База данных разработана в среде Microsoft SQL Server 2018. Клиентское приложение (сайт) разработано в среде Microsoft Visual Studio. Использованные языки: Transact-SQL, C#, JS.

1. **Концептуальное описание предметной области**

Предметная область – кинотеатр, предоставляющий прокат различных фильмов.

Кинотеатр должен обладать определенным административным и оперативным штатом, в который входят: администраторы, менеджеры, киномеханики, кассиры (работники у залов), уборщики.

Основная деятельность – показ фильмов, поэтому у кинотеатра должна быть база кинолент, со временем меняющая свой состав. В состав так же входят кинозалы, кассы.

Кинотеатр имеет возможность предоставление проката реклам перед сеансами.

Покупка билетов может осуществляться через кассы, либо через интернет-сайт.

1. **Характеристики ИС**

Хранимые данные в базе:

* информация о кинозалах:
  + номер зала, наличие 3D, количество рядов и мест;
* информация о прокатываемым фильмам:
  + название фильма, год выхода, режиссер, длительность, жанр, рейтинг;
* информация о сотрудниках:
  + ФИО, должность, паспорт, опыт, номер телефона;
* информация о должностях:
  + название позиции, описание, ранг, зарплата;
* информация о сеансах:
  + номер фильма, номер зала, время показа, возрастной рейтинг, тип сеанса, стоимость билета;
* информация о рекламодателях:
  + ФИО, название компании, номер телефона;
* информация о прокатываемой рекламе:
  + номер сеанса, ответственный сотрудник, рекламодатель, название рекламы, длительность, стоимость;
* информация о кассах
  + номер кассы, сотрудник, время смены, время работы;
* информация о билетах
  + номер сеанса, номер кассы, тип билета, номер ряда и места;
* информация о типах сеансов, билетов
  + название типа, описание, скидка (для билетов);
* так же информация о том, к чему прикреплены сотрудники (кинозалы, кассы):
  + идентификационный номер сотрудника, номер объекта;

Бизнес-правила, накладываемые на базу:

* в одном сеансе может прокатываться только один кинофильм;
* билет на сеанс не стандартного типа может быть куплен только в кинотеатре;
* в одно и тоже время в одном зале не могут идти разные сеансы;
* сеансы могут выставляться только на будущие даты;
* рекламный ролик должен стоить не меньше 5 тыс. рублей;
* рекламный ролик должен длиться не более 180 секунд;
* сотрудник может занимать только одну должность;
* билет может быть куплен только на один сеанс, но сеанс может иметь множество билетов;
* один работник-кассир может обслуживать множество кинозалов, но кинозал может иметь множество работников;
* реклама может проигрываться на множестве сеансов, но сеанс может иметь множество рекламных роликов;
* принимать рекламу и назначать ее на сеансы могут только сотрудники, чей ранг равен трем;
* работать на кассах могут только работники с рангом равным двум;
* билеты, купленные через интернет-сайт, могут быть только стандартного типа;
* ФИО, паспорт, номер телефона для сотрудников должны быть уникальными;
* ФИО и номер телефона рекламодателя должны быть уникальными;

1. **Анализ существующих решений для предметной области**
2. **Разработка общей структуры ИС**

В данной курсовой работе для построения информационной системы была взята «Клиент-Серверная» архитектура. Система на этой архитектуре делится на две подсистемы: клиентскую и серверную.

Клиентская часть представляет собой компьютер, на котором находится приложение (программа, сайт и т.п.), с помощью которого пользователь получает и запрашивает необходимую информацию, которая «обрисовывается» в приложении.

На серверной части расположены обработчик, СУБД и база данных. Они принимают запрос с клиентской части, обрабатывают его и возвращают результат. Сервер – это так же компьютер, который мощнее и производительнее. На нем располагается основная логика.

Подобное разделение позволяет снизить требования к клиентским компьютерам, облегчить управление и наладку сервера, обезопасить хранимую информацию баз данных, снизить нагрузку на сеть.

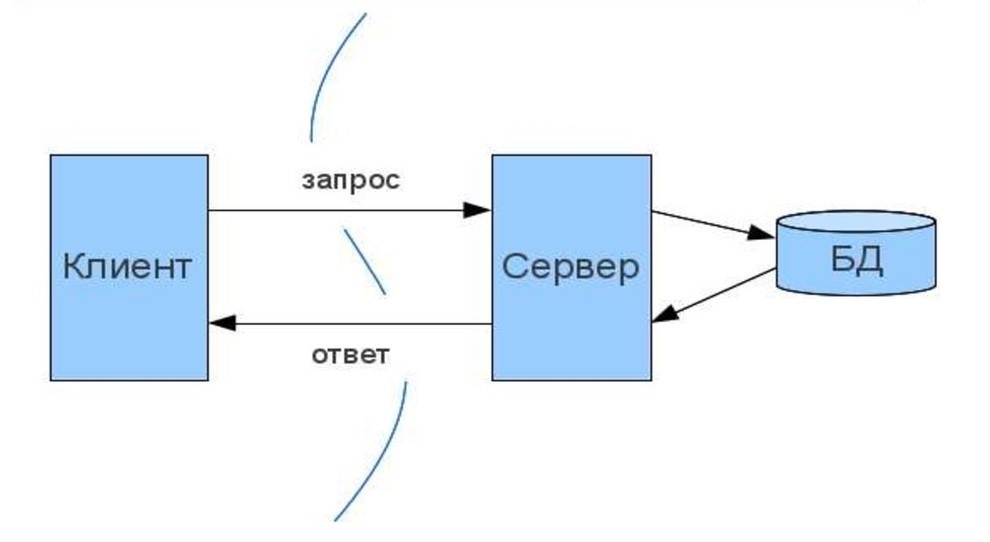


Рисунок 1 – Клиент-Серверная архитектура

1. **Разработка серверной части ИС**
   1. **Инфологическое проектирование БД**

Инфологическое проектирование – это процесс создания инфологической модели данных о предметной области, не зависящее от любых физических аспектов ее представления.

Подобная модель данных не подчиняется условностям какой-либо СУБД, вычислительной среды или языков программирования. Модель представляет собой структуру знаний об области: выделяются и классифицируются компоненты, связи между ними.

В данной курсовой работе для концептуального проектирования выбрана модель «Сущность-Связь» − ER-модель, позволяющая выделять ключевые сущности и связи между ними. Нотация П. Чена.

**Выявление сущностей и связей**

В выбранной предметной области можно выявить следующие сущности:

* Кинолента;
* Сотрудник;
* Рекламодатель;
* Кинозал;
* Сеанс;
* Билет;
* Касса;
* Реклама;
* Должность.

Между сущностями можно выделить следующие связи:

* Сеанс содержит кинофильм;
* Сеанс проигрывается в кинозале;
* Сеанс содержит рекламу;
* Билет используется для прохода на сеанс;
* Билет покупается на кассе;
* Рекламодатель покупает рекламу;
* Сотрудник продает рекламу;
* Сотрудник обладает должностью;
* Сотрудник работает на кассе;
* Сотрудник работает в кинозале;
* Сеанс обладает типом;
* Билет обладает типом

**Построение ER-диаграмм**

1. Сеанс содержит кинофильм



Рисунок 5.1.1 – ER-диаграмма: Сеанс содержит кинофильм

Во время сеанса может проигрываться только один кинофильм. Один кинофильм может проигрываться в множестве сеансов.

Кинофильм не обязательно содержит сеанс. Сеанс обязательно содержит кинофильм.

1. Сеанс проигрывается в кинозале

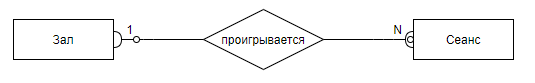


Рисунок 5.1.2 – ER-диаграмма: Сеанс проигрывается в кинозале

Сеанс может проигрываться только в одном кинозале. В кинозале может проигрываться множество сеансов.

Сеанс обязательно содержит кинозал. Зал не обязательно содержит сеанс.

1. Сеанс содержит рекламу

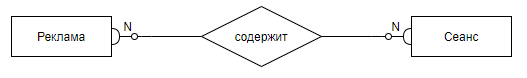


Рисунок 5.1.3 – ER-диаграмма: Сеанс содержит рекламу

В одном сеансе может быть много рекламы. Одна реклама может быть на множестве сеансов.

Реклама и сеанс не обязательно содержат друг друга

1. Билет используется для прохода на сеанс

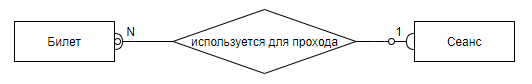


Рисунок 5.1.4 – ER-диаграмма: Билет используется для прохода на сеанс

Билет содержит только один сеанс. На сеанс может быть продано множество билетов.

Билет обязательно содержит сеанс. Сеанс не обязательно содержит билеты.

1. Билет покупается на кассе

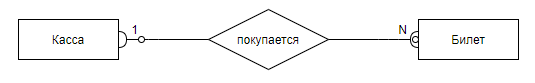


Рисунок 5.1.5 – ER-диаграмма: Билет покупается на кассе

На кассе можно купить множество билетов. Билет содержит только одну кассу.

Касса не обязательно содержит билет. Билет обязательно содержит кассу.

1. Рекламодатель покупает рекламу



Рисунок 5.1.6 – ER-диаграмма: Рекламодатель покупает рекламу

Рекламодатель может покупать множество рекламы. Реклама содержит только одного рекламодателя.

Рекламодатель не обязательно покупает рекламу. Реклама обязательно содержит рекламодателя.

1. Сотрудник продает рекламу



Рисунок 5.1.7 – ER-диаграмма: Сотрудник продает рекламу

Сотрудник может продавать множество рекламы. Реклама содержит множество сотрудников.

Сотрудник не обязательно продает рекламу. Реклама обязательно содержит сотрудника.

1. Сотрудник обладает должностью

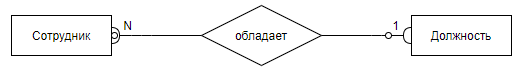


Рисунок 5.1.8 – ER-диаграмма: Сотрудник обладает должностью

Сотрудник может обладать только одной должностью. Должностью может обладать множество сотрудников.

Сотрудник обязательно содержит должность. Должность не обязательно содержит сотрудника.

1. Сотрудник работает на кассе;

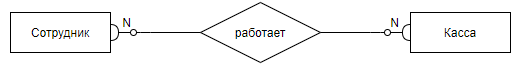


Рисунок 5.1.9 – ER-диаграмма: Сотрудник работает на кассе

Сотрудник может работать на множестве касс. На кассе может работать множество сотрудников.

Сотрудник не обязательно работает на кассе. Касса не обязательно содержит сотрудника.

1. Сотрудник работает в кинозале



Рисунок 5.1.10 – ER-диаграмма: Сотрудник работает в кинозале

Сотрудник может работать во множестве кинозалов. В зале может работать множество сотрудников.

Сотрудник не обязательно работает в зале. Кинозал не обязательно содержит сотрудника.

1. Сеанс обладает типом



Рисунок 5.1.11 – ER-диаграмма: Сеанс обладает типом

Сеанс может обладать только одним типом. Тип может быть у множества сеансов.

Сеанс обязательно обладает типом. Тип не обязательно содержит сеанс.

1. Билет обладает типом

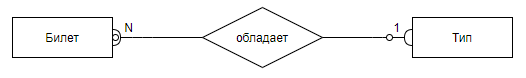


Рисунок 5.1.12 – ER-диаграмма: Билет обладает типом

Билет может обладать только одним типом. Тип может быть у множества билетов.

Билет обязательно обладает типом. Тип не обязательно содержит билет.

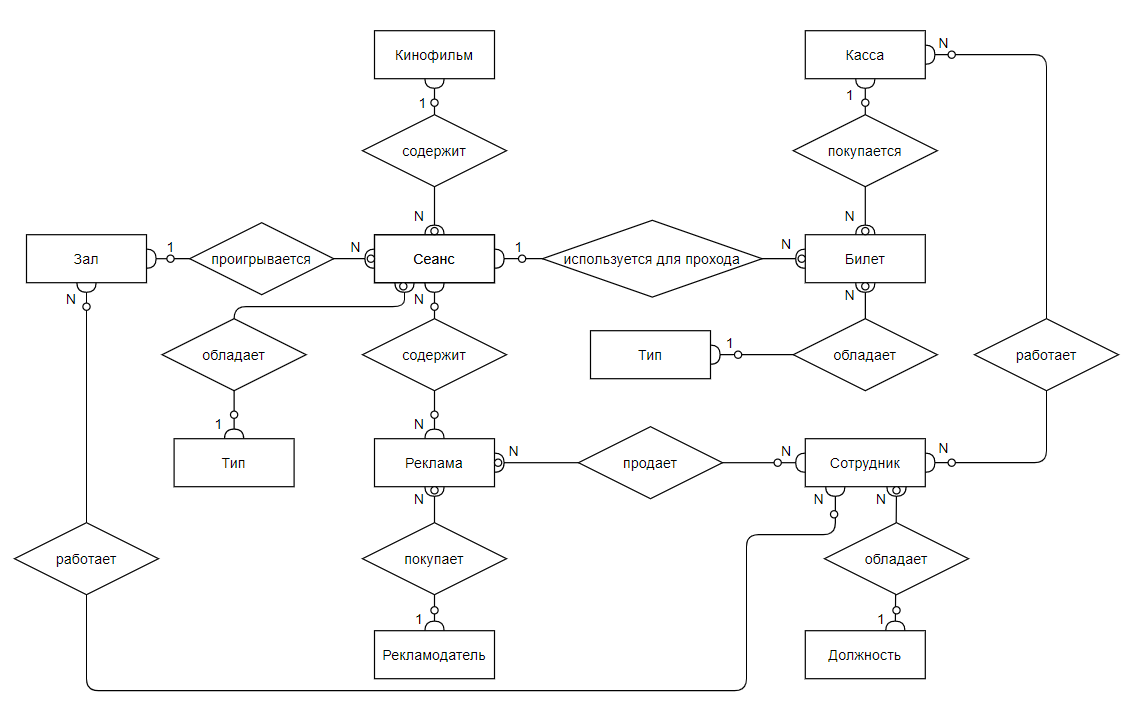


Рисунок 5.1.13 – Общая ER-диаграмма

* 1. **Даталогическое проектирование БД**

Даталогическое проектирование – это процесс создания даталогической модели, которая отражает логические связи между элементами данных независимо от их содержания и физической структуры. При этом модель должна быть представлена в терминах модели данных концептуального уровня для выбранной СУБД.

Существует несколько представлений структур данных. Для данной курсовой работы был выбран реляционный подход, где отношения представлены таблицами и связями между ними. Для таких баз применяется нормализация, целью которой является устранение недостатков структуры базы данных, приводящих к избыточности, которая, в свою очередь, потенциально приводит к различным аномалиям и нарушениям целостности данных.

При формировании отношений по ER-диаграммам используются определенные правила, описывающие отношения и зависимости между ними.

**Переход от ER-диаграмм к предварительным отношениям**

Проанализировав составленные ранее ER-диаграммы, можно сформировать следующие отношения:

1. Сеанс содержит кинофильм. По правилу 4 получаем:

* Сеанс (SeanceID, FilmID);
* Кинофильм (FilmID);

1. Сеанс проигрывается в кинозале. По правилу 4 получаем:

* Сеанс (SeanceID, HollID);
* Кинозал (HollID);

1. Сеанс содержит рекламу:

* Сеанс (SeanceID);
* Реклама (AdID);
* СеансРеклама (SeanceID, AdID);

1. Билет используется для прохода на сеанс. По правилу4 получаем:

* Сеанс (SeanceID);
* Билет (TicketID, SeanceID);

1. Билет покупается на кассе. По правилу 4 получаем:

* Билет (TicketID, CashboxID);
* Касса (CashboxID);

1. Рекламодатель покупает рекламу. По правилу 4 получаем:

* Рекламодатель (AdvertiserID, AdID);
* Реклама (AdID);

1. Сотрудник продает рекламу. По правилу 6 получаем:

* Сотрудник (EmployeeID);
* Реклама (AdID);
* РекламаСотрудник (EmployeeID, AdID);

1. Сотрудник обладает должностью. По правилу 4 получаем:

* Сотрудник (EmployeeID, PositionName);
* Должность (PositionName);

1. Сотрудник работает на кассе. По правилу 6 получаем:

* Сотрудник (EmployeeID);
* Касса (CashboxID);
* КассаСотрудник (CashboxID, EmployeeID);

1. Сотрудник работает в кинозале. По правилу 6 получаем:

* Сотрудник (EmployeeID);
* Кинозал (HollID);
* КинозалСотрудник (HollID, EmployeeID);

1. Сеанс обладает типом. По правилу 4 получаем:

* Сеанс (SeanceID, TypeName);
* Тип (TypeName);

1. Билет обладает типом. По правилу 4 получаем:

* Билет (TicketID, TypeName);
* Тип (TypeName).

**Заполнение предварительных отношений атрибутами**

1. Сеанс (SeanceID, FilmID, HollID, ShowTime, AgeRating, SeanceType, TicketCost);
2. Кинофильм (FilmID, FilmName, ReleaseYear, Director, Duration, Genre, Rating, FilmImage);
3. Кинозал (HollID, TdEnable, RowNumber, SeatNumber);
4. Реклама (AdID, Advertiser, AdvertisingName, AdvertisingDuration, AdvertisingCost);
5. Рекламодатель (AdvertiserID, AdvertiserName, CompanyName, AdvertiserPhone);
6. Билет (TicketID, TicketType, SeanceID, CashboxID, RowNumber, SeatNumber, Cost);
7. Касса (CashboxID, StaffChangeTime, WorkTime);
8. Сотрудник (EmployeeID, EmployeeName, Position, Passport, Experience, Phone);
9. Должность (PositionName, Responsibilities, EmployeeRank, Salary);
10. КассаСотрудник (EmployeeID, CashboxID);
11. КинозалСотрудник (HollID, EmployeeID, StaffChangeTime);
12. РекламаСотрудник (EmployeeID, AdID);
13. Тип для билета (TypeName, TypeDescription, Discount);
14. Тип для сеанса (TypeName, TypeDescription);
15. СеансРеклама (SeanceID, AdID);

**Проверка предварительных отношений на соответствие нормальным формам**

1. Сеанс (SeanceID, FilmID, HollID, ShowTime, AgeRating, SeanceType, TicketCost):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

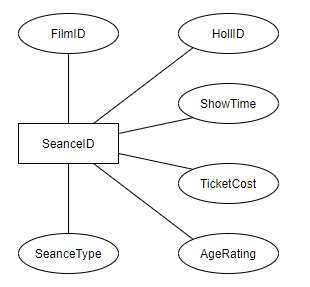


Рисунок 5.2.1 – Отношение Сеанс

1. Кинофильм (FilmID, FilmName, ReleaseYear, Director, Duration, Genre, Rating, FilmImage):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

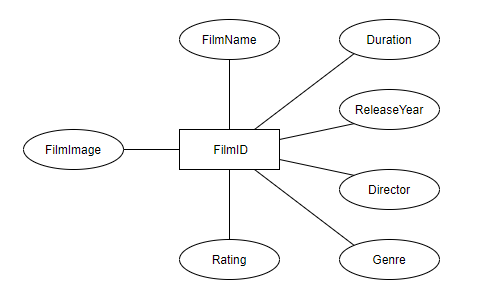


Рисунок 5.2.2 – Отношение Кинофильм

1. Кинозал (HollID, TdEnable, RowNumber, SeatNumber):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

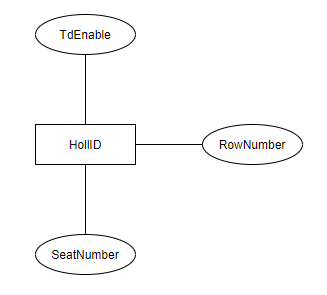


Рисунок 5.2.3 – Отношение Кинозал

1. Реклама (AdID, Advertiser, AdvertisingName, AdvertisingDuration, AdvertisingCost):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

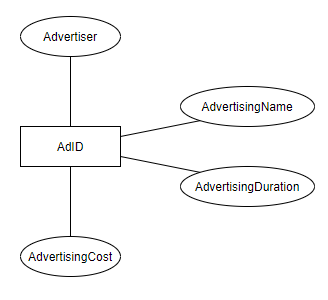


Рисунок 5.2.4 – Отношение Реклама

1. Рекламодатель (AdvertiserID, AdvertiserName, CompanyName, AdvertiserPhone):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

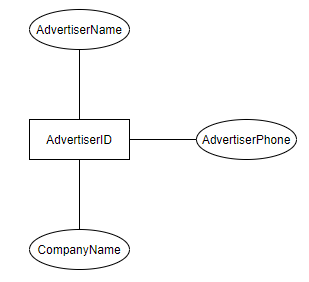


Рисунок 5.2.5 – Отношение Рекламодатель

1. Билет (TicketID, TicketType, SeanceID, CashboxID, RowNumber, SeatNumber, Cost):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

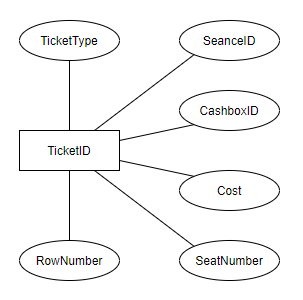


Рисунок 5.2.6 – Отношение Билет

1. Касса (CashboxID, StaffChangeTime, WorkTime):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

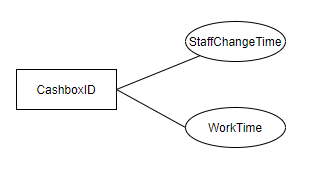


Рисунок 5.2.7 – Отношение Касса

1. Сотрудник (EmployeeID, EmployeeName, Position, Passport, Experience, Phone):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

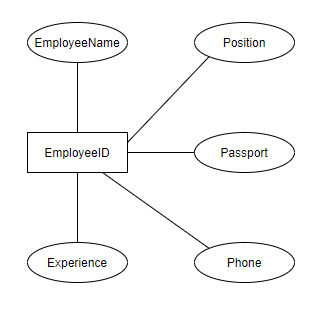


Рисунок 5.2.8 – Отношение Сотрудник

1. Должность (PositionName, Responsibilities, EmployeeRank, Salary):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

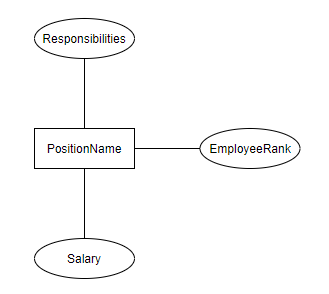


Рисунок 5.2.9 – Отношение Должность

1. КассаСотрудник (EmployeeID, CashboxID):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

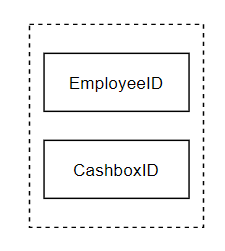


Рисунок 5.2.10 – Отношение КассаСотрудник

1. КинозалСотрудник (HollID, EmployeeID, StaffChangeTime):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

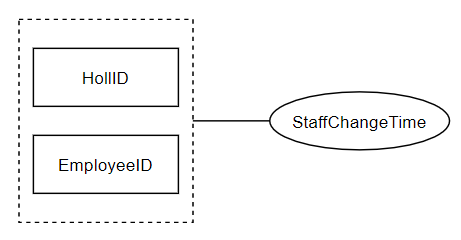


Рисунок 5.2.11 – Отношение КинозалСотрудник

1. РекламаСотрудник (EmployeeID, AdID):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

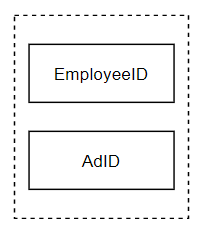


Рисунок 5.2.12 – Отношение РекламаСотрудник

1. Тип для билета (TypeName, TypeDescription, Discount):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

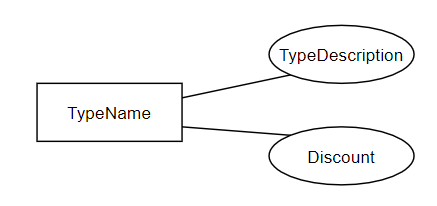


Рисунок 5.2.13 – Отношение Тип

1. Тип для сеанса (TypeName, TypeDescription):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

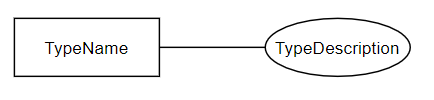


Рисунок 5.2.14 – Отношение Тип

1. СеансРеклама (SeanceID, AdID):

* Отношение находится в 2НФ, так как оно находится в 1НФ и каждый не ключевой атрибут функционально полно зависит от потенциального ключа;
* Отношение находится в 3НФ, так как оно находится в 2НФ и отсутствуют транзитивные зависимости не ключевых атрибутов от ключевых;
* Отношение находится в БКНФ, так как каждая функциональная зависимость имеет в качестве своего детерминанта потенциальный ключ;

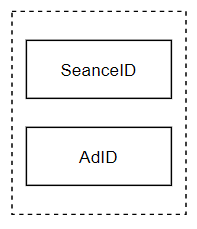


Рисунок 5.2.15 – Отношение СеансРеклама

**Построение схемы базы данных**

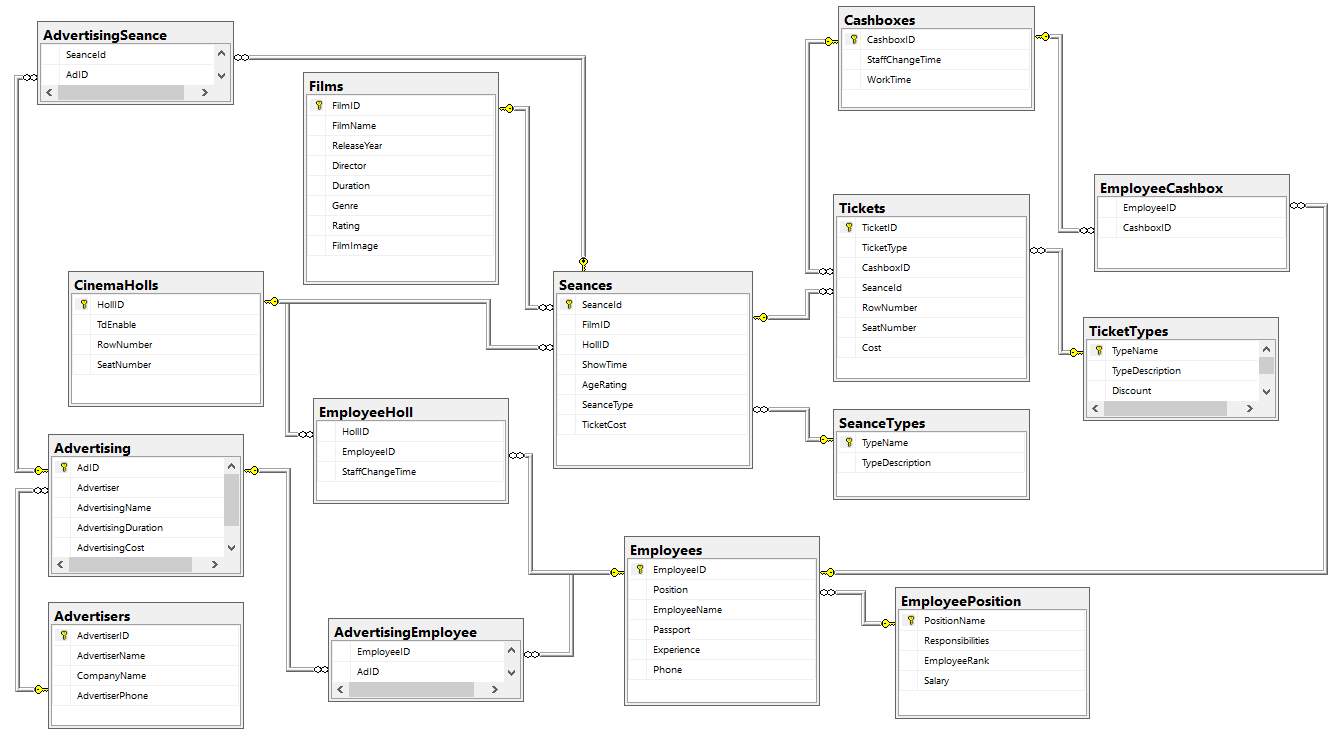


Рисунок 5.2.16- Схема базы данных

* 1. **Программирование объектов БД**

**Создание ограничений**

Ограничения – это специальные объекты в СУБД, которые позволяют задать допустимые значения для определенных столбцов с целью обеспечения автоматической безопасности базы данных.

В Microsoft SQL Server реализовано несколько типов ограничений, при этом пользователю доступно создание собственных правил.

Основные ограничения:

* [NOT] NULL – запрет или разрешение NULL значений в столбце;
* PRIMARY KEY – ограничение первичного ключа;
* FOREIGN KEY – ограничение внешнего ключа;
* UNIQUE – ограничение уникальности значения столбца;
* CHECK – проверка значения столбца на соответствие условию;
* DEFAULT – задание стандартного значения для столбца;

Задание пользовательских правил производится посредством команды CREATE RULE «name» AS («condition»). Чтобы привязать правило к столбцу таблицы нужно исполнить процедуру sp\_bindrule.

Чтобы создать пользовательские умолчания, необходимо использовать команду CREATE DEFAULT «» AS («»). Для привязки – sp\_bindefault.

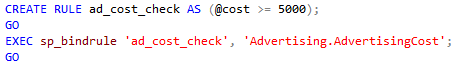
Данная комбинация команд будет использована для задания ограничений в некоторых таблицах.

**Ограничения, реализованные в базе данных:**

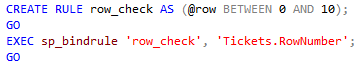
* Правило, ограничивающее рейтинг фильма:



* Правило, ограничивающее минимальную стоимость рекламы:



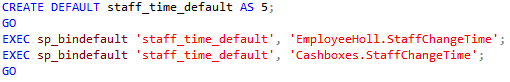
* Правило, ограничивающее номер ряда в кинозале:



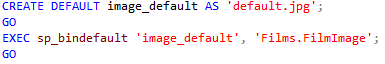
* Умолчание, задающее опыт сотрудника:



* Умолчание, задающее стандартное время смены персонала:



* Умолчание, задающее стандартный путь к изображению фильма:

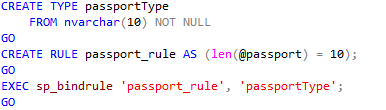


**Создание пользовательских типов**

Данные типы позволяют реализовывать пользовательскую логику на основе стандартных типов базы. Создание типа производится посредством команды CREATE TYPE «name» FROM \*type\* [NULL | NOT NULL]. На столбцы, обладающие пользовательским типом, так же можно навешивать ограничения.

Пользовательские типы, реализованные в базе данных:

* Паспорт, основанный на символьном типе:



* Номер телефона, основанный на символьном типе:



* Путь к изображению фильма, основанный на символьном типе:



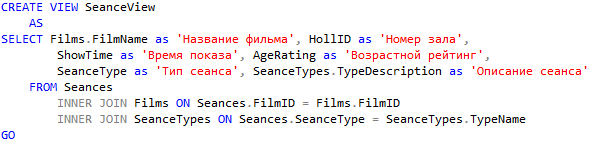
**Разработка представлений**

Представление — это виртуальная таблица, содержимое которой определяется запросом. Как и таблица, представление состоит из ряда именованных столбцов и строк данных. Пока представление не будет проиндексировано, оно не существует в базе данных как хранимая совокупность значений.

Для создания представления используется команда   
CREATE VIEW «name» AS SELECT…

Представления, разработанные для базы данных:

* Информация о сеансе:



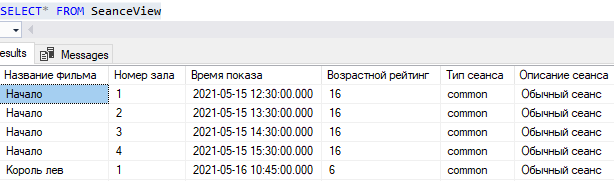
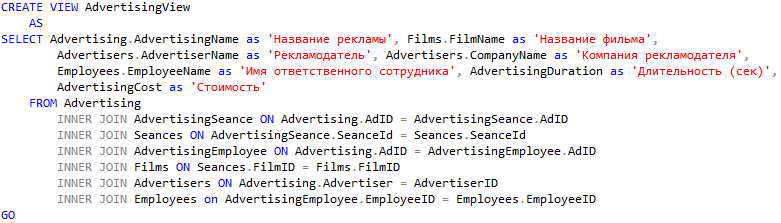


Рисунок 5.3.1 – Представление SeanceView

* Информация о рекламе:



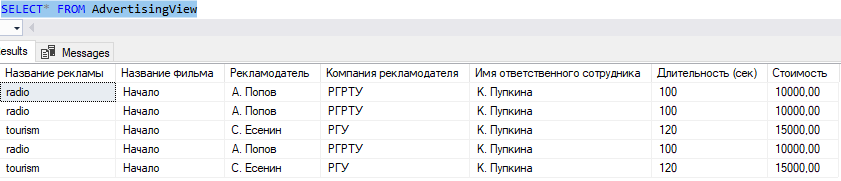
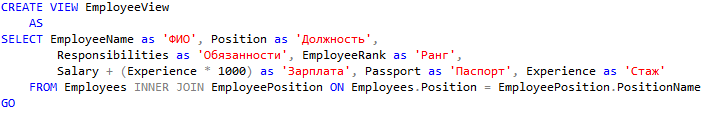


Рисунок 5.3.2 – Представление AdvertisingView

* Информация о сотрудниках:



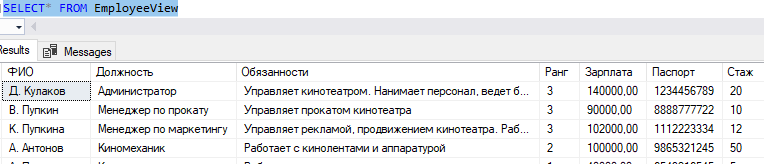


Рисунок 5.3.3 – Представление EmployeeView

**Разработка сценария создания БД**

Скрипт создания новой базы данных находится в Приложении 1.

**Разработка ХП, триггеров**

Хранимая процедура – это модуль, состоящий из SQL-операторов и хранящийся в БД на SQL-сервере. Вызывается с помощью команды EXEC.

Триггеры – это хранимая процедура, которая исполняется автоматически при возникновении определенных событий на сервере БД. Такими событиями могут являются попытки изменения данных в базе: INSERT, UPDATE, DELETE, которые применяются к таблицам или представлениям. В данной работе реализованы триггеры только для DML.  
 Есть два типа триггеров - INSERT OF и AFTER. Первый вызывается каждый раз заместо операций DML. Второй – после успешного исполнения операций DML.

Скрипты создания ХП и триггеров находятся в Приложении 2

1. **Разработка клиентского приложения**

**Выбор программных компонентов клиентской части**

Клиентское приложение для данной курсовой работы было разработано в среде разработки Microsoft Visual Studio на языке С# с использованием фреймворка ASP.NET.

Данный фреймворк обладает несколькими модулями для разработки веб-приложений. В работе был использован ASP.NET MVC, который предоставляет создавать веб-страницы с использованием шаблона проектирования MVC.

Платформа ASP.NET MVC базируется на взаимодействии трех компонентов: контроллера, модели и представления.

Контроллер принимает запросы, обрабатывает пользовательский ввод, взаимодействует с моделью и представлением и возвращает пользователю результат обработки запроса.

Модель представляет слой, описывающий логику организации данных в приложении. В модель могут быть включены несколько экземпляров разных классов, что позволяет использовать лишь одну модель для получения данных.

Представление получает данные из контроллера и генерирует элементы пользовательского интерфейса для отображения информации. Для этого применяется смесь языка C# и HTML – cshtml. Это позволяет получать и обрабатывать данные посредством как языка C#, так и JavaScript, а также отрисовывать страницы с помощью CSS.

**Разработка интерфейса пользователя**