Государственное автономное профессиональное образовательное учреждение   
Чувашской Республики «Межрегиональный центр компетенций –  
 Чебоксарский электромеханический колледж» Министерства образования Чувашской Республики

Дисциплина МДК.11.01 Технология разработки и защита базы данных

КУРСОВОЙ ПРОЕКТ

Технология разработки и защиты базы данных

автоматизированной информационной системы билетов в кинотеатре

КП.Ип5-21.18.МДК.11.01.ПЗ

Выполнил студент 3 курса, группы Ип5-21

Софронов Е. В.

(Фамилия И. О.)

(подпись) (чч.мм.гггг)

Преподаватель Ордяков Д.Е.

(Фамилия И. О.)

Защищен

(чч.мм.гггг)

с оценкой

Подпись

(подпись) (расшифровка подписи)

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc153033665)

[1 Инфологическое проектирование 5](#_Toc153033666)

[1.1 Анализ и описание предметной области 5](#_Toc153033667)

[1.2 Разработка концептуальной модели 6](#_Toc153033668)

[2. Определение требований к операционной системе 9](#_Toc153033669)

[3. Выбор субд и других программных средств 12](#_Toc153033670)

[4. Логическое проектирование реляционной базы данных 13](#_Toc153033671)

[5. Реализация проекта базы данных 23](#_Toc153033672)

[Заключение 30](#_Toc153033673)

[Список использованных источников 31](#_Toc153033674)

[Приложение А 32](#_Toc153033675)

[Приложение Б 33](#_Toc153033676)

[Приложение B 34](#_Toc153033677)

[Приложение Г 38](#_Toc153033678)

ВВЕДЕНИЕ

В настоящее время наличие у предприятий большого количества информации стало нормой. Для того чтобы грамотно работать с этой информацией, необходима информационная система, которая будет работать с ней и отвечать определенным параметрам. Такой системой может выступать база данных.

База данных (БД) – именная совокупность данных, отражающая состояние объектов и их отношения в рассматриваемой предметной области.

Кинотеатральная отрасль не стала исключением. Сегодня кинотеатральный бизнес является развивающейся отраслью экономики. В связи с этим возрастает потребность в эффективных и надежных информационных системах для управления кинотеатральными комплексами.

Использование автоматизированных информационных систем позволяет ускорить и упростить работу с базой данных кинотеатрального комплекса. Это в свою очередь повышает эффективность управления кинотеатральным бизнесом и улучшает качество обслуживания клиентов.

Внедрение работ с базами данных в деятельность предприятия создает возможность обрабатывать обширные объемы данных. В связи с ростом количества информации, хранимой в базе данных, становится все более важной защита этой информации от несанкционированного доступа и утечек. Разработка технологий защиты баз данных становится неотъемлемой частью работы специалистов в этой области.

В связи с этим возникает актуальность выбранной темы – «Технология разработки и защиты базы данных автоматизированной информационной системы продажи билетов в кинотеатре». Создание приложения даст возможность в любой удобный момент времени просматривать информацию, редактировать, добавлять, удалять и фильтровать данные БД.

Цель: разработать базу данных для автоматизированной информационной системы кинотеатрального комплекса.

Задачи:

* произвести анализ и описать предметную область;
* разработать концептуальную и логическую модели; составить список сущностей и атрибутов, построить реляционную модель на основе логической модели;
* спроектировать базу данных с использованием средств SQL;
* разработать пользовательский интерфейс.

1 Инфологическое проектирование

1.1 Анализ и описание предметной области

Концептуальное проектирование – это построение модели предметной   
области, ориентированной на восприятие человека и независимой от типа системы управления базами данных (СУБД) и типа самой базы данных.

Под предметной областью (ПО) понимается совокупность связанных между собой функций, задач управления в некоторой области деятельности предприятия, с помощью которых достигается выполнение поставленной цели.

Множество фрагментов составляют из себя предметную область. Каждый фрагмент предметной области характеризуется множеством объектов и процессов, использующих объекты, а также множеством пользователей, с разными взглядами на ПО.

Модель предметной области представляет собой знания о предметной   
области. Знания могут быть неформальными и формальными. Модель предметной области описывает скорее процессы, происходящие в предметной области и данные, используемые этими процессами. От того, насколько правильно смоделирована предметная область, зависит успех дальнейшей разработки приложений.

1.2 Разработка концептуальной модели

Для разработки базы данных необходимо определить основные сущности и необходимые поля. Так же следует выделить сопоставимые поля для определения связей между таблицами.

База данных должна содержать данные:

* о кинозалах в кинотеатре;
* о клиентах;
* о билетах;
* о фильмах;

Учитывая все вышеперечисленное, база данных должна содержать следующие сущности и их атрибуты.

Сущность «Rooms» (Кинозалы):

* «Room\_ID» (Персональный код кинозала);
* «Room\_status» (Наличие киносеанса в данный момент).

Сущность «Films» (Фильмы):

* «Film\_ID» (Персональный код фильма);
* «Film\_Name» (Название фильма);
* «Film\_age\_restriction» (Возрастное ограничение фильма).

Сущность «Customers» (Клиенты):

* «Customer\_ID» (Персональный код клиента);
* «Customer\_Name » (Имя клиента);
* «Customer\_age » (Возраст клиента).

Сущность «Tickets» (Билеты):

* «Ticket\_ID» (Персональный код билета);
* «Ticket\_Data » (Дата билета);
* «Ticket\_Cost » (Стоимость билета);
* «Customer\_ID» (Персональный код клиента);
* «Film\_ID » (Персональный код фильма);
* «Room\_ID » (Персональный код кинозала).

Таким образом, были определены основные объекты будущей информационной системы.

Установим связи между сущностями и распишем их с помощью ERD-инструкции:

«Customers» и «Tickets», связь «один-ко-многим». Бизнес-правила:

1. каждый клиент (Customers) может иметь один или более билетов (Tickets).
2. каждый билет (Tickets) должен иметь одного и только одного клиента (Customers).

«Tickets» и «Films», связь «один-ко-многим». Бизнес-правила:

1. каждый фильм (Films) может быть в одном или более билетах (Tickets).
2. каждый билет (Tickets) должен содержать один и только один фильм (Films).

«Tickets» и «Rooms», связь «многие-ко-многим». Бизнес-правила:

1. каждый билет (Tickets) может содержать один или более залов (Rooms).
2. каждый кинозал (Rooms) должен содержаться в одном или более билетах (Tickets).

При осуществлении нормализации текущих отношений было осуществлено приведение к нормальным формам:

* первая нормальная форма (1NF) — все атрибуты должны иметь только одно значение;
* вторая нормальная форма (2NF) — отношения находятся в первой нормальной форме и все не ключевые атрибуты зависят от первичного ключа;
* третья нормальная форма (3NF) — отношения находятся во второй нормальной форме и все неключевые атрибуты не транзитивно зависят от первичного ключа.

Нормализация – это процесс преобразования отношений базы данных к виду, отвечающему нормальным формам.

Нормализация предназначена для приведения структуры баз данных к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение физического объёма базы данных.

Под избыточностью понимают повторение данных в разных строках одной таблицы или в разных таблицах БД. Итогом нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации.

В заключение главы выделим все выполненные в процессе создания концептуальной модели действия:

* были определены объекты базы данных и их атрибуты;
* были определены связи между объектами;
* были определены действия необходимые для приведения отношений в базе данных к нормальной форме.

2. Определение требований к операционной системе

По данной формуле будет вычисляться объем памяти, требуемый для хранения данных, на основании данных из таблицы 2.1:

𝑛

Мд = 2 ∑ 𝑙𝑖 ∗ (𝑁𝑖 + 𝑁𝑎𝑖 )

𝑖=1

где 𝑙𝑖 - длина записи в i-ой таблице (в байтах);

𝑁𝑖 - примерное (максимально возможное) количество записей в i-ой таблице;

𝑁𝑎 – количество записей в архиве i-ой таблицы;

*n* – количество таблиц в БД.

Таблица 2.1 – Атрибуты, отображающие длину записи в каждой таблице

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Отношения | Атрибут | Тип данных | Размер,  байт | Среднее  кол-во | Объем,  байт |
| Rooms | Room\_ID | Int | 2 | 1,5 | 60 |
| Room\_status | Boolean | 1 |
| Tickets | Ticket\_Id | Int | 2 | 2,5 | 600 |
| Ticket\_Data | date | 3 |
| Ticket\_Cost | float | 4 |
| Customer\_ID | Int | 2 |  |  |
| Film\_id | Int | 2 |
| Room\_id | int | 2 |
| Films | Film\_ID | Int | 2 | 67,3 | 4040 |
| Film\_Name | Varchar(100) | 100 |
| Film\_age\_restriction | Varchar(100) | 100 |
| Customers | customer\_Id | int | 2 | 67,3 | 8080 |
| customer\_name | Varchar(100) | 100 |
| Customer\_age | Varchar(100) | 100 |

Таким образом, объем памяти хранения данных за один год, составит:

МС = 2 × (60+600+4040+8080) = 25560 байт

Полученный объем памяти не будет являться постоянным, потому что он увеличивается или уменьшается по ходу работы с базой данных.

После проведенных вычислений становится ясно, что объем данных в информационной системе небольшой. Следовательно, для реализации проекта подойдет любой современный компьютер.

Объем оперативной памяти, который необходим, будет определяться требованиями выбранной системой управления баз данных, следовательно, СУБД MS SQL подходит для введения данной базы данных.

Для поддержания целостности данных выполняется резервное копирование баз данных. План резервного копирования приведен в таблице 2.2.

Таблица 2.2 – План резервного копирования

|  |  |
| --- | --- |
| День недели | Вид резервного копирования |
| Понедельник | Разностное резервное копирование (билетов и клиентов) |
| Вторник | Разностное резервное копирование (билетов и клиентов) |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| День недели | Вид резервного копирования |
| Среда | Разностное резервное копирование (билетов и клиентов) |
| Четверг | Разностное резервное копирование (билетов и клиентов) |
| Пятница | Разностное резервное копирование (билетов и клиентов) |
| Суббота | Разностное резервное копирование (билетов и клиентов) |
| Воскресенье | Полное резервное копирование всей БД |

В Microsoft SQL Server принята практика разных типов резервного копирования. Пользователям доступно:

1. полное – делается резервная копия всей БД;
2. дифференциальное или разностное – осуществляется копирование данных с того момента, когда осуществлялось ее последнее полное резервирование;
3. логов или инкрементальное.

3. Выбор СУБД и других программных средств

При создании базы данных и клиентского приложения были решены важные и сложные задачи, для успешного выполнения которых нужно было правильно сделать выбор средства реализации проекта. Для создания концептуальной и логической моделей базы данных Oracle SQL Developer Data Modeler предоставил много возможностей. Приложение является бесплатным графическим программным обеспечением, с помощью которого можно сконструировать различные модели данных, такие как логические, реляционные, многомерные, сетевые и т. д.

СУБД MS SQL Server стала отличным выбором для создания, использования и управления будущей базы данных. Обладая не только всеми необходимыми функциями для создания клиентского приложения, но и имея высокую производительность, она стала самым удачным выбором, так как другие приложения компании Microsoft могут работать с ней без каких-либо проблем.

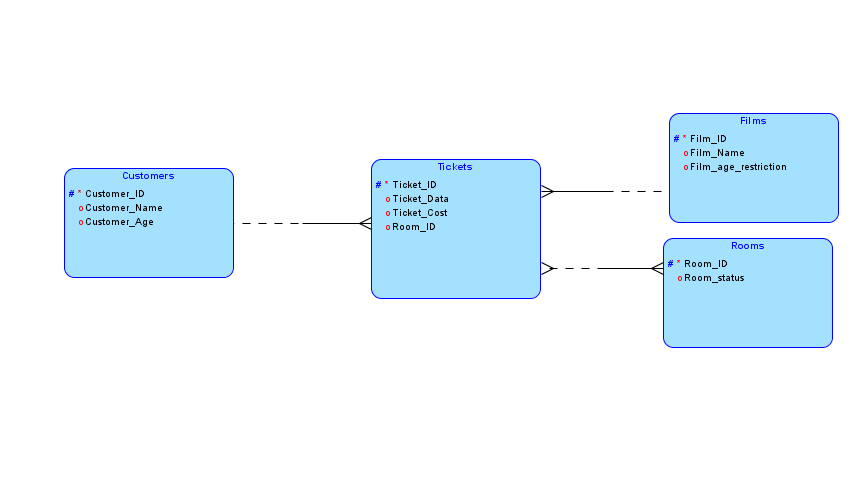
Следовательно, наилучшим вариантом после выбора СУБД стала программная среда Microsoft Visual Studio с использованием объектно-ориентированного языка программирования C# для разработки пользовательского интерфейса и программного кода, и комфортной работы пользователей с базой данных.

Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows.

Язык программирования C# был выбран из-за его перспективности среди других языков и обстоятельств, облегчающих дальнейшее сопровождение программного обеспечения.

4. Логическое проектирование реляционной базы данных

Логическое проектирование является созданием схемы баз данных на основе конкретной модели данных, то есть в данном случае реляционной модели данных, представленных на рисунках 4.1 и 4.2.

Рисунок 4.1 – Логическая модель базы данных

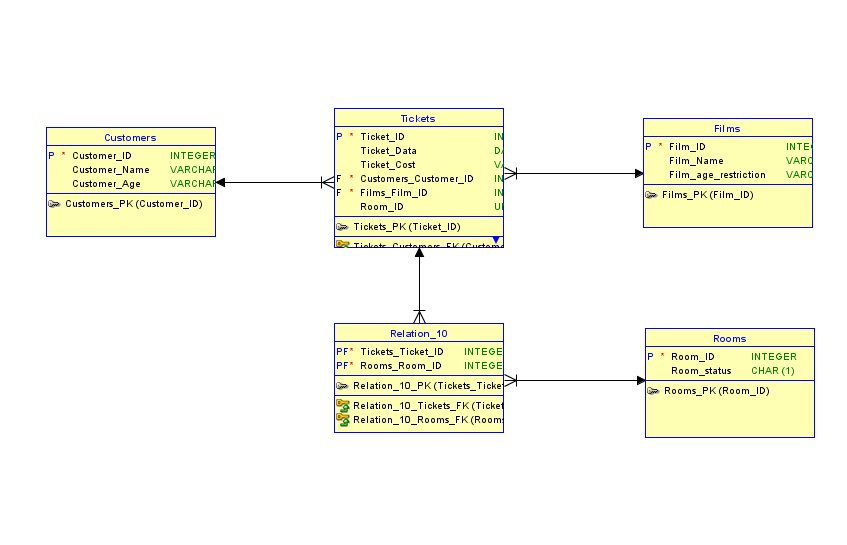


Рисунок 4.2 – Реляционная модель базы данных

Сущностью в базе данных может являться практически любой объект, который возможно логически выделить из предметной области, а ее атрибутом именованный столбец таблицы. Особыми атрибутами считаются первичные и внешние ключи.

Первичный ключ – это столбец, значения которого однозначно определяет строку.

Внешний ключ – это столбец в данной таблице, значения которого совпадает со значениями столбца, являющийся первичным ключем в другой таблице.

Во время разработки структуры базы данных были созданы таблицы, представления, хранимые процедуры и триггеры.

В базе данных таблица является совокупностью, связанной между собой информации, хранящейся в определенном порядке. Любая таблица состоит из кортежей (строк) и атрибутов (столбцов).

Представленная база данных состоит из четырех таблиц:

1. Таблица «Customers»;
2. Таблица «Tickets»;
3. Таблица «Films»;
4. Таблица «Rooms».

Таблица «Customers» представляет собой совокупность информации о клиентах кинотеатра. Таблица «Tickets» хранит информацию о билетах, таблица «Films» хранит информацию о фильмах. Таблица «Rooms» хранит данные о кинозалах.

Все таблицы созданы в одной базе данных. Описание структуры таблиц и их полей приведены в таблицах 4.1 – 4.5.

Таблица 4.1 – Структура таблицы «Customers»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Customer\_ID | Int | 2 | Идентифицирующий  номер клиента | Первичный  ключ |
| Customer\_Name | Varchar(100) | 100 | Имя клиента | NOT NULL |
| Customer\_Age | Varchar(100) | 100 | Возраст | NOT NULL |

Таблица 4.2 – Структура таблицы «Films»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Film\_ID | Int | 2 | Идентифицирующий  номер фильма | Первичный  ключ |
| Film\_Name | Varchar(100) | 100 | Название фильма | NOT NULL |
| Film\_age\_restriction | Varchar(100) | 100 | Возрастное ограничение | NOT NULL |

Таблица 4.3 – Структура таблицы «Rooms»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Room\_ID | Int | 2 | Идентифицирующий  номер кинозала | Первичный  ключ |
| Room\_status | bool | 1 | Статус кинозала | NOT NULL |

Таблица 4.4 – Структура таблицы «Tickets»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Ticket\_ID | Int | 2 | Идентифицирующий  номер билета | Первичный  ключ |
| Ticket\_Data | Date | 3 | Дата билета | NOT NULL |

Продолжение таблицы 4.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ticket\_Cost | float | 4 | Стоимость заказа | NOT NULL |
| Customer\_Id | Int | 2 | ID клиента | NOT NULL |
| Film\_ID | Int | 2 | ID фильма | NOT NULL |
| Room\_ID | Int | 2 | ID кинозала | NOT NULL |

После создания таблиц и добавления данных в MS SQL Server база данных выглядит следующим образом (рисунки 4.3 – 4.4).

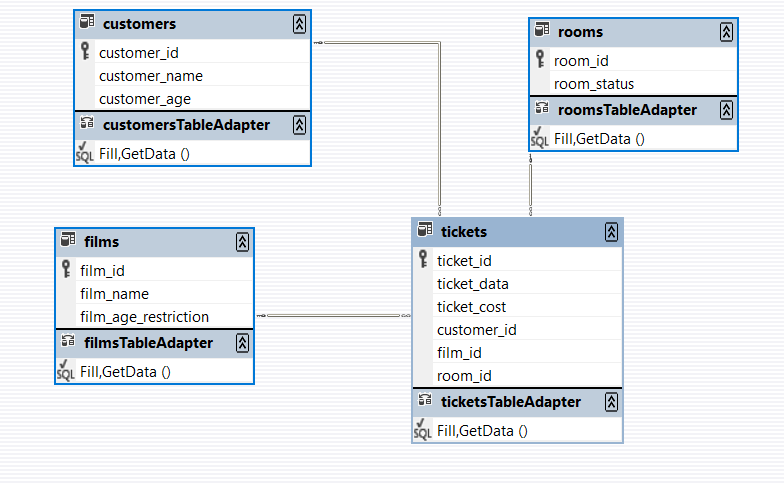


Рисунок 4.3 – Диаграмма базы данных

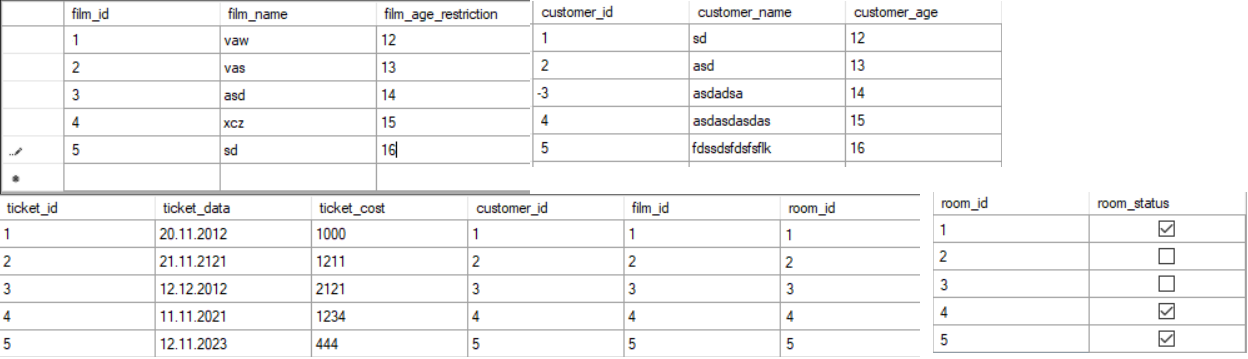


Рисунок 4.4 – Таблицы базы данных

Представление по своему внешнему виду напоминает таблицу, но на самом деле является не самостоятельным объектом, а результатом запроса на выборку к существующей в базе данных таблице и полностью зависит от нее, так как само по себе представление не может содержать никаких данных.

Созданные представления в данном проекте выглядят следующим образом:

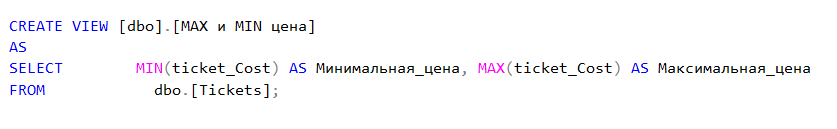


Рисунок 4.5 – Представление «MAX и MIN цена»

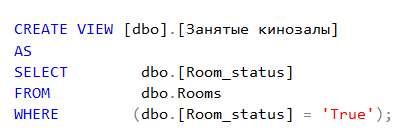


Рисунок 4.6 – Представление «Занятые кинозалы»

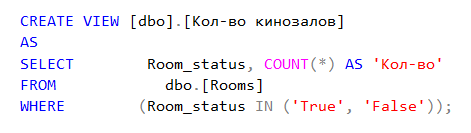


Рисунок 4.7 – Представление «Кол-во кинозалов»

Хранимая процедура – это набор команд, состоящий из одного или нескольких операторов SQL, сохраняющий в БД в откомпилированном виде. Такие процедуры могут иметь входные и выходные параметры, а также локальные переменные.

При разработке курсового проекта были созданы следующие хранимые процедуры:

Cost2tickets рисунок 4.8. Данная хранимая процедура позволяет добавлять данные в таблице Tickets.

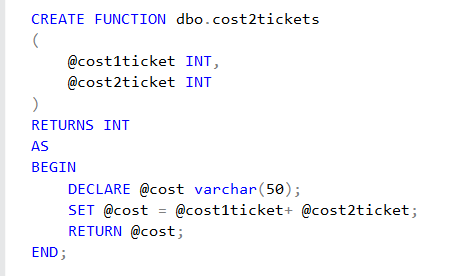


Рисунок 4.8 – Скрипт на создание хранимой процедуры LogTrigger

film\_id рисунок 4.9. Данная хранимая процедура выборочно посмотреть данные из таблицы Films.

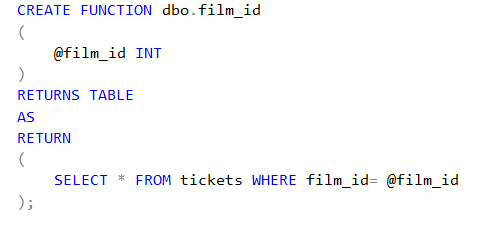


Рисунок 4.9 – Скрипт на создание хранимой процедуры LogTrigger

Avgcost рисунок 4.10. Данная хранимая процедура позволяет посмотреть среднюю стоимость билетов в таблице Tickets.

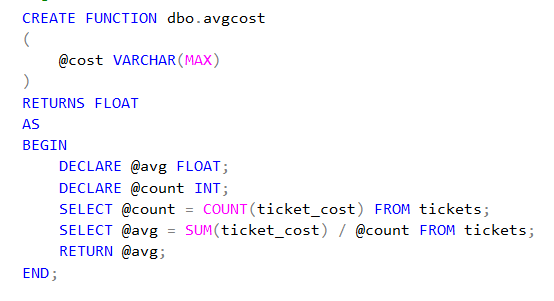


Рисунок 4.10 – Скрипт на создание хранимой процедуры LogTrigger

Триггер – это вид хранимой процедуры, выполняющийся автоматически при возникновении событий языка обработки данных в БД. Триггеры используются для проверки целостности данных, а также для отката транзакций. Существует несколько видов: вставка, удаление или редактирование информации   
в определенной таблице или во всей базе данных.

Также в курсовом проекте были созданы триггеры:

Logtrigger, реализующий добавление и удаление, код создания триггера представлен на рисунке 4.11.

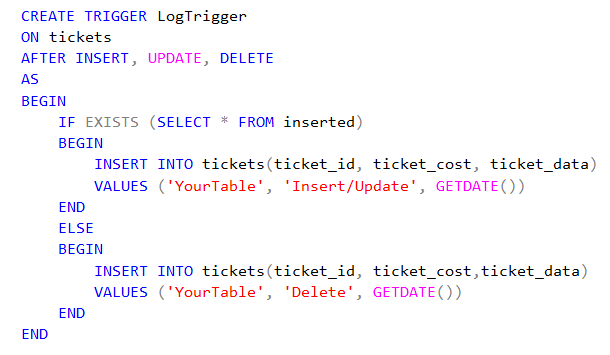


Рисунок 4.11 – Скрипт на создание триггера Logtrigger

UniqueValueTrigger, реализующий невозможность добавление повторяющегося id, код создания триггера представлен на рисунке 4.12.

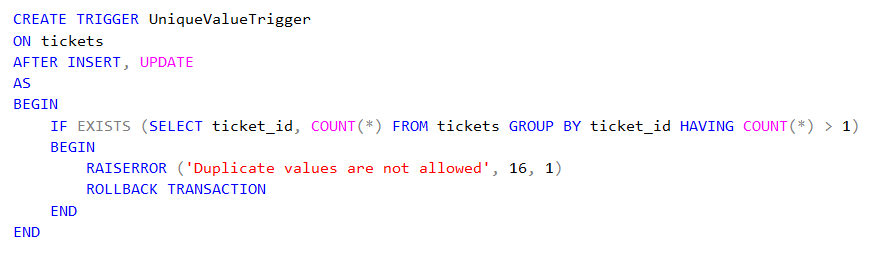
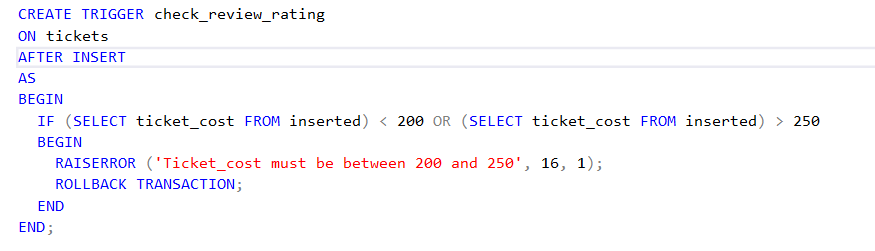


Рисунок 4.12 – Скрипт на создание триггера UniqueValueTrigger

Check\_review\_rating, реализующий невозможность добавление билета выше стоимость 250 или ниже 200, код создания триггера представлен на рисунке 4.13.

 Рисунок 4.13 – Скрипт на создание триггера Check\_review\_rating

Фрагменты создания таблиц, представлений, хранимых процедур и триггеров приведены в приложении В.

5 Реализация проекта базы данных

Для полного комфорта и удобства пользователя при работе с базой данных было создано пользовательское приложение, представляющее собой форму Windows Forms, созданную в среде разработки Microsoft Visual Studio 2019.

Подключение к базе данных осуществляется с помощью технологии ADO.NET Entity Framework, то есть объектно-ориентированной технологией доступа к данным, являясь object-relational mapping решением для .NET Framework от Microsoft.

При запуске приложения отрывается окно авторизации рисунок 5.1, в котором предлагается вести свой логин и пароль. Если вводится неправильный пароль, появляется окно с оповещением рисунок 5.2.

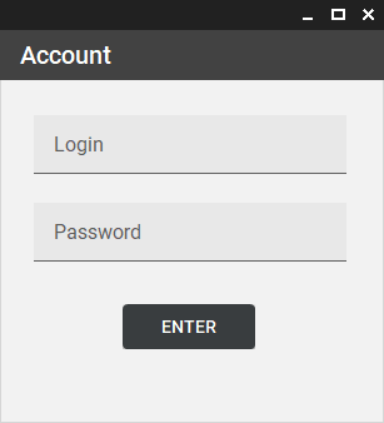


Рисунок 5.1 – Окно авторизации

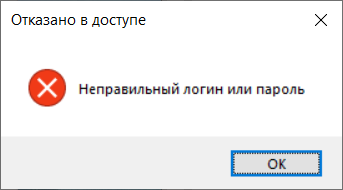


Рисунок 5.2 – Сообщение об ошибке

Если пароль указан верно и авторизация происходит через администратора БД, то происходит переход на следующие формы: рисунок 5.3 – рисунок 5.6.

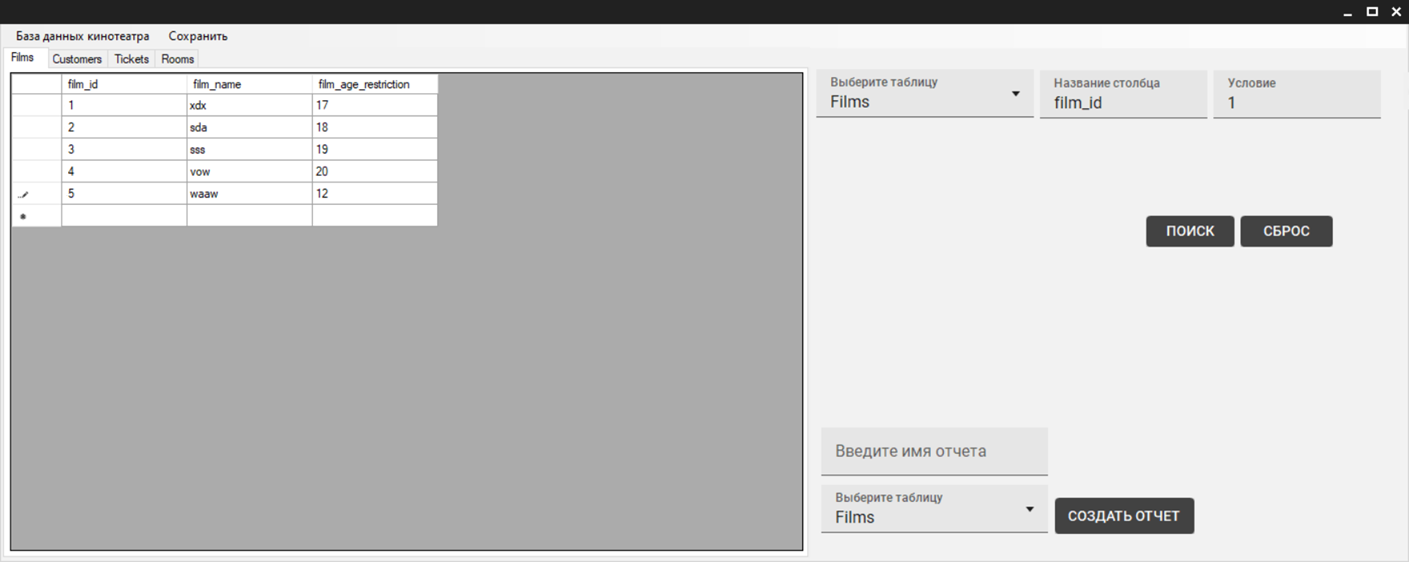


Рисунок 5.3 – Окно администратора работы с таблицами, вкладка «Films»

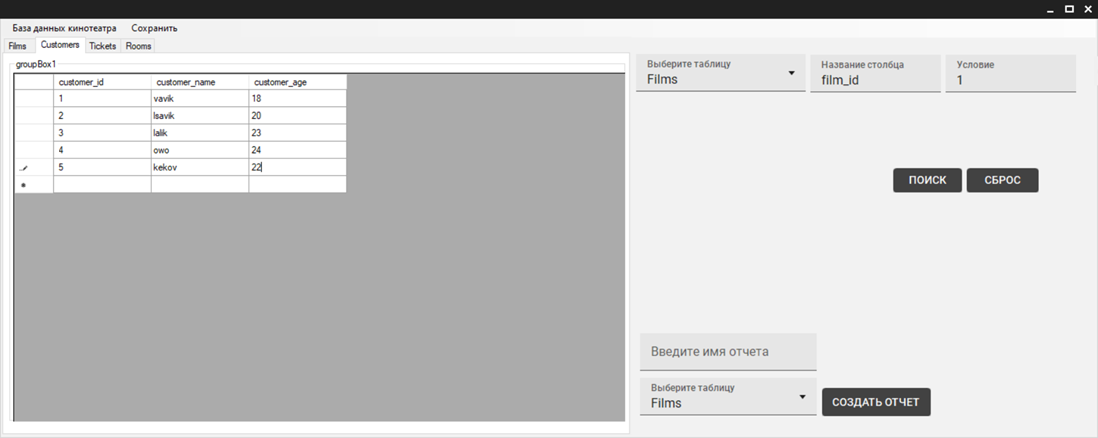


Рисунок 5.4 – Окно администратора работы с таблицами, вкладка «Customers»

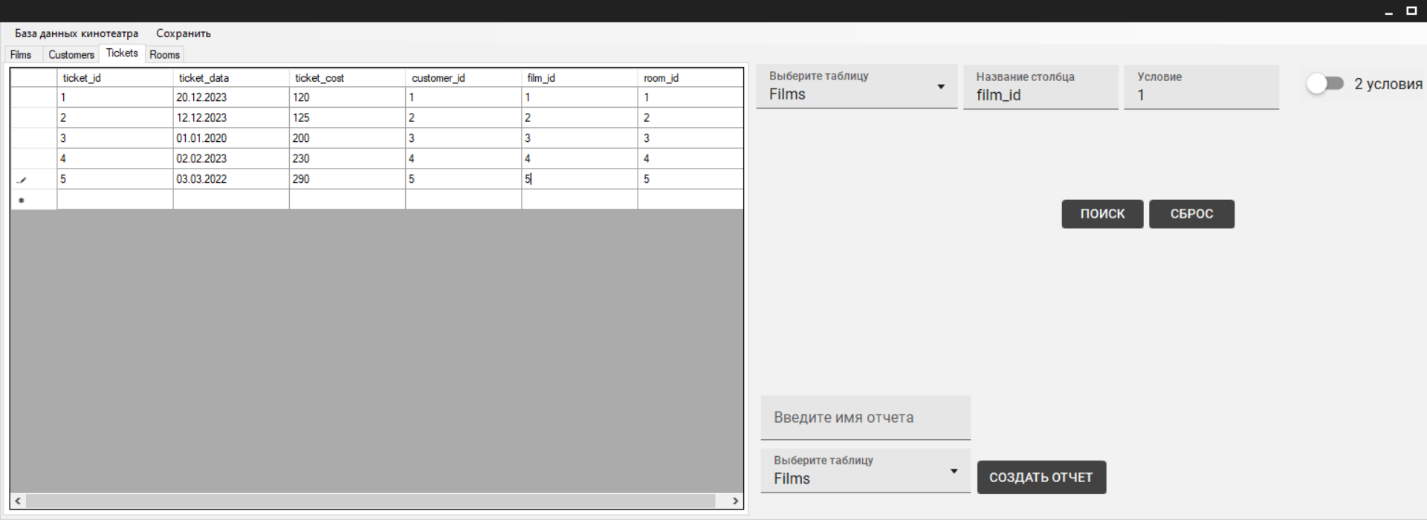


Рисунок 5.5 – Окно администратора работы с таблицами, вкладка «Tickets»

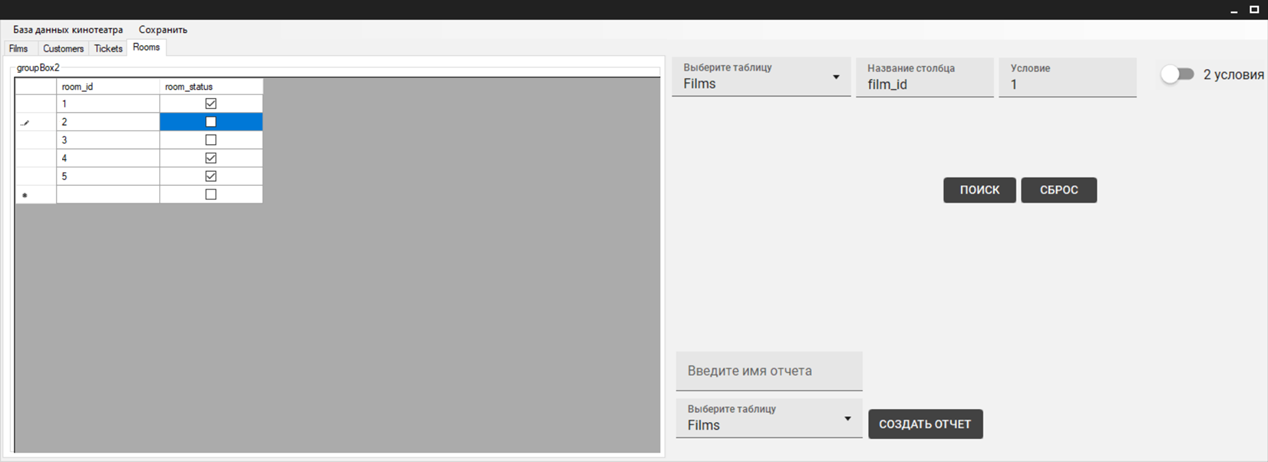


Рисунок 5.6 – Окно администратора работы с таблицами, вкладка «Rooms»

Если пароль указан верно и авторизация происходит через обычного пользователя БД, то происходит переход на следующие формы: рисунок 5.7 – рисунок 5.10.

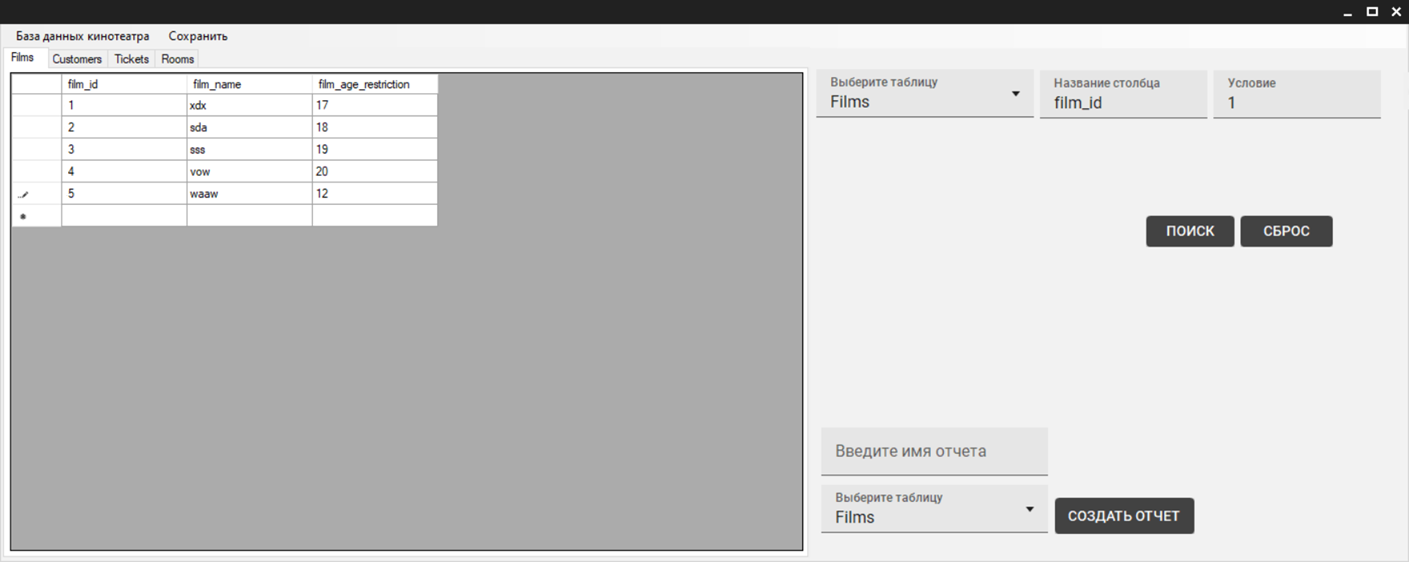


Рисунок 5.7 – Окно обычного пользователя для работы с таблицами, вкладка «Films»

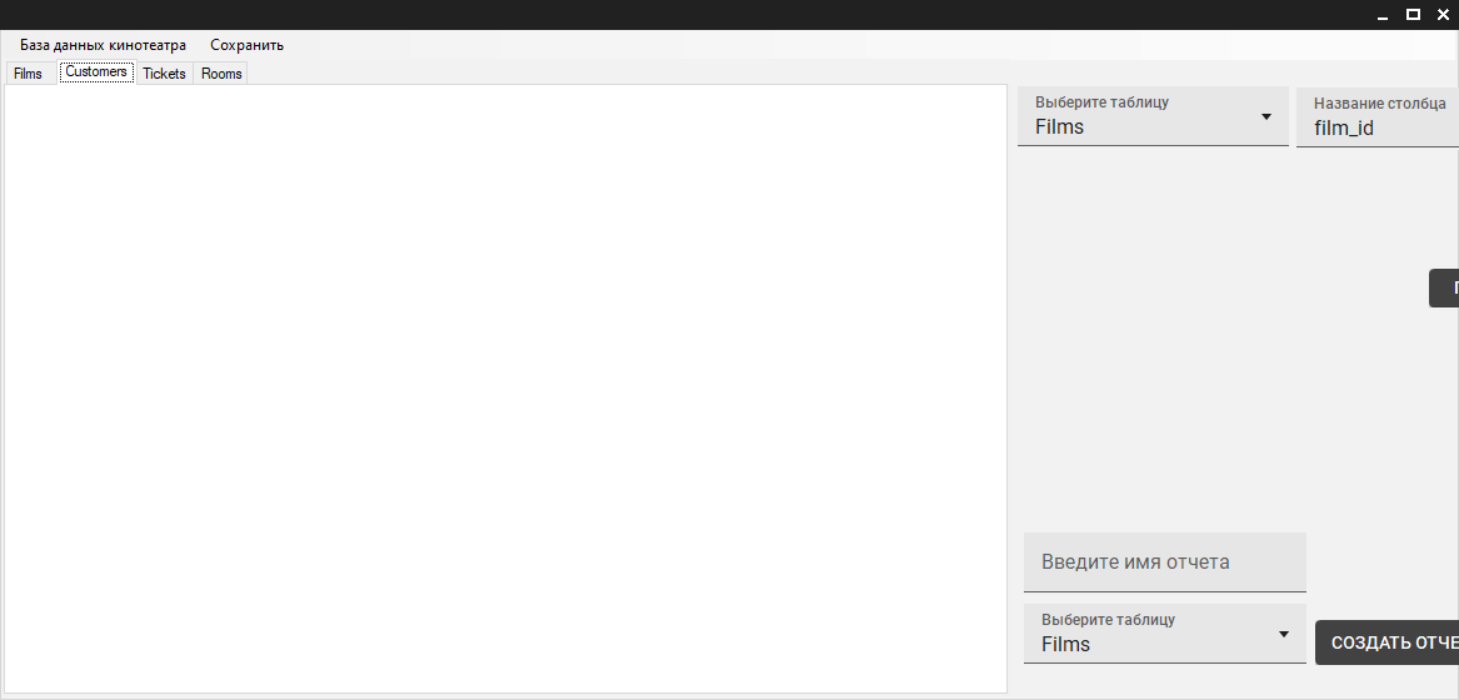


Рисунок 5.8 – Окно обычного пользователя для работы с таблицами, вкладка «Customers»

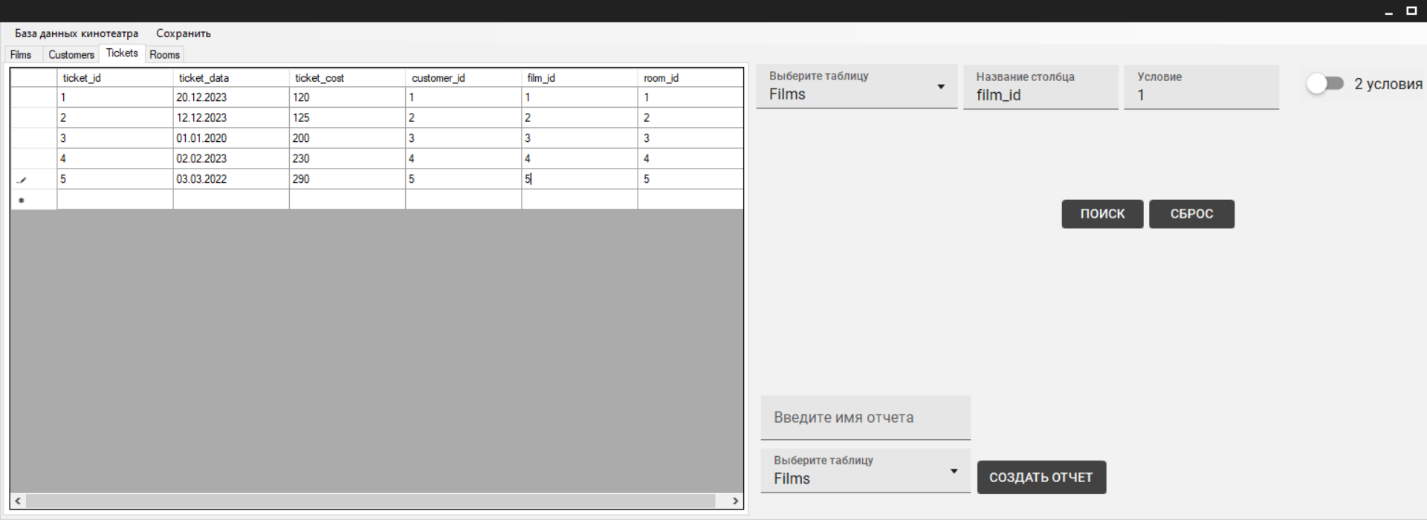


Рисунок 5.9 – Окно обычного пользователя для работы с таблицами, вкладка «Tickets»

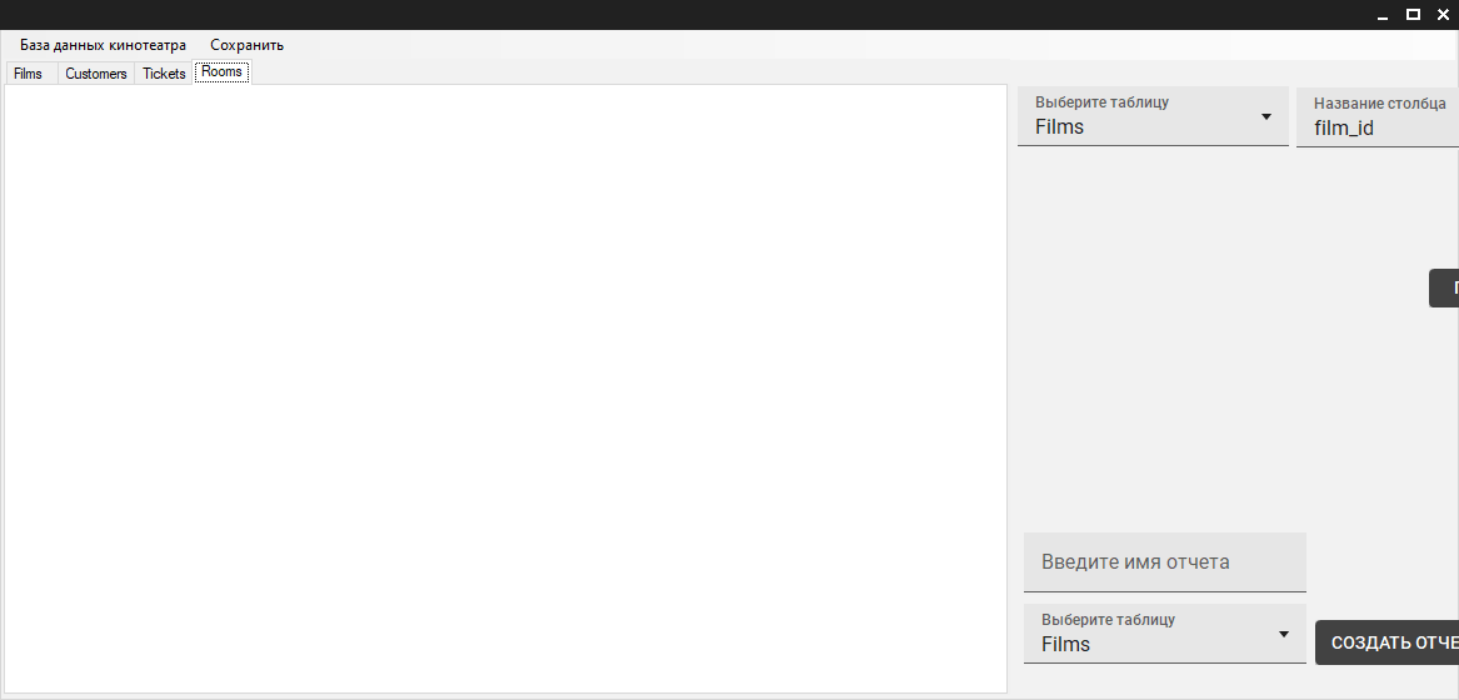


Рисунок 5.10 – Окно обычного пользователя для работы с таблицами, вкладка «Rooms»

Кнопка «Отчет» на форме «Form2» создает отчет в Microsoft Excel –– рисунок 5.11

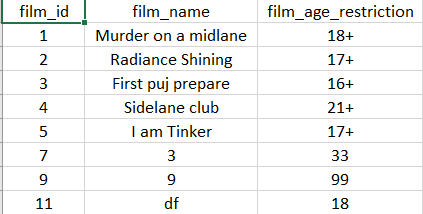


Рисунок 5.11 – Отчет в Microsoft Excel

На главной форме реализован фильтр по 2 условиям с возможностью выбора дополнительного условия фильтрации между «И» и «ИЛИ»: рисунок 5.12 – рисунок 5.14

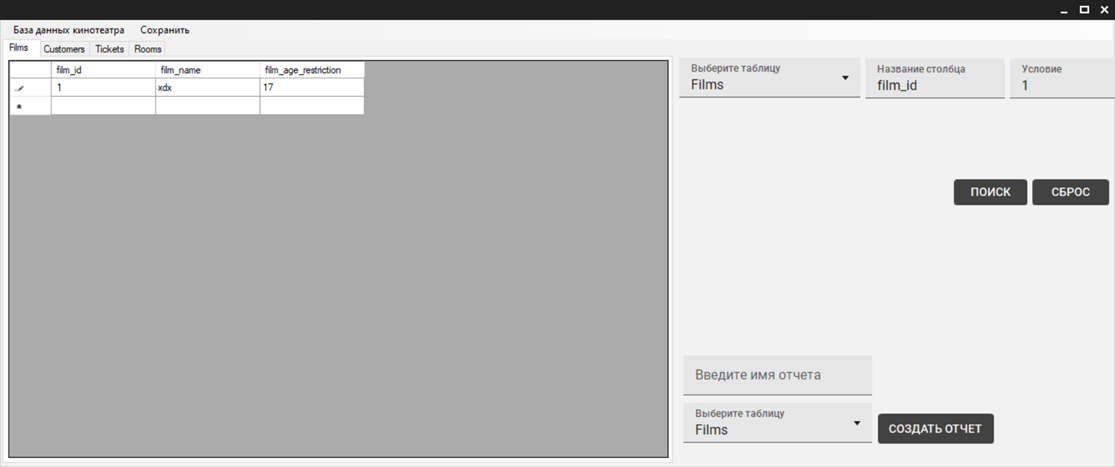


Рисунок 5.12 – Поиск по одному условию

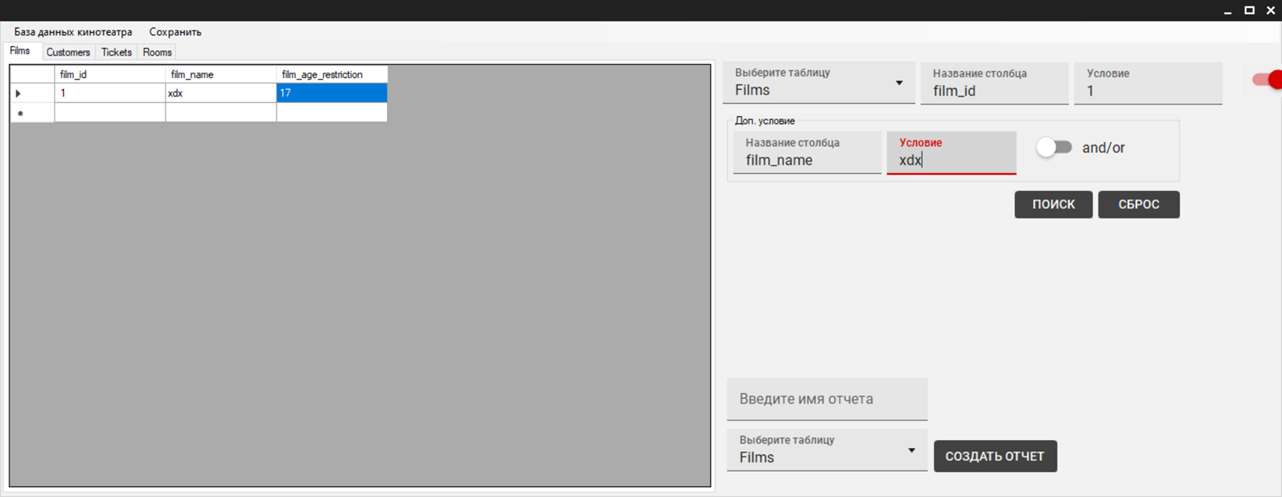


Рисунок 5.13 – Поиск по двум условиям через «ИЛИ»

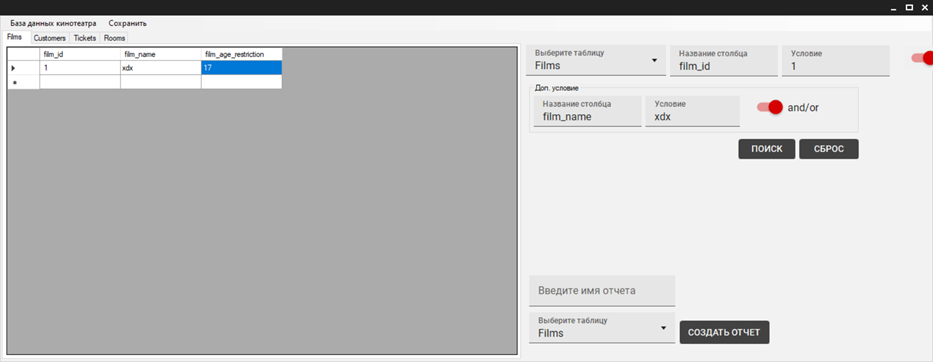


Рисунок 5.14 – Поиск по двум условиям через «И»

Результатом пройденного пути стало создание приложения, полностью удовлетворяющим все требования. Данное приложение позволяет свободно работать с созданной ранее БД.

ЗАКЛЮЧЕНИЕ

В ходе разработки курсового проекта были решены все поставленные задачи:

* были произведены анализ и описание предметной области;
* разработаны концептуальная и логическая модель базы данных;
* спроектирована база данных с использованием средств SQL;
* разработан пользовательский интерфейс.

А также получены новые навыки по работе с такими программами как: Oracle SQL Developer Data Modeler, Microsoft Visual Studio 2019 и СУБД Microsoft SQL Server.

Итоговое приложение полностью удовлетворяет запросы, так как позволяет свободно вносить правки в базу данных, создавать информацию о новых фильмах, клиентах, билетах и кинозалах. Программный продукт способен создавать итоговые отчеты и обрабатывать большое количество информации.

После решения всех поставленных задач была достигнута цель курсового проекта – разработана база данных автоматизированной информационной системы билетов в кинотеатре.

Можно и дальше улучшать приложение, например, создание следующий версии исходя из приобретенного опыта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Microsoft. Практическое руководство. Сортировка и фильтрация данных ADO.NET с помощью компонента BindingSource в Windows Forms. — URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/>controls/ sort-and-filter-ado-net-data-with-wf-bindingsource-component?view=netframeworkdes ktop-4.8 (дата обращения: 09.12.2023);
2. Документация Microsoft. Создание базы данных и добавление таблиц в приложениях платформа .NET Framework с помощью Visual Studio — URL: https://learn.microsoft.com/ru-ru/visualstudio/data-tools/create-a-sql-database-by-using-a-designer?view=vs-2022 (дата обращения: 7.10.2023);
3. Руководство по использованию Oracle Data Modeler — URL: <https://docs.oracle.com/database/sql-developer-data-modeler-18.1/DMDUG/toc.htm> (дата обращения: 10.09.2023);
4. Подключение и запрос экземпляра SQL Server с помощью SQL Server Management Studio (SSMS) — URL: <https://learn.microsoft.com/ru-ru/sql/ssms/quickstarts/ssms-connect-query-sql-server?view=sql-server-ver16>

(дата обращения: 12.11.2023);

### ПРИЛОЖЕНИЕ А

Логическая модель

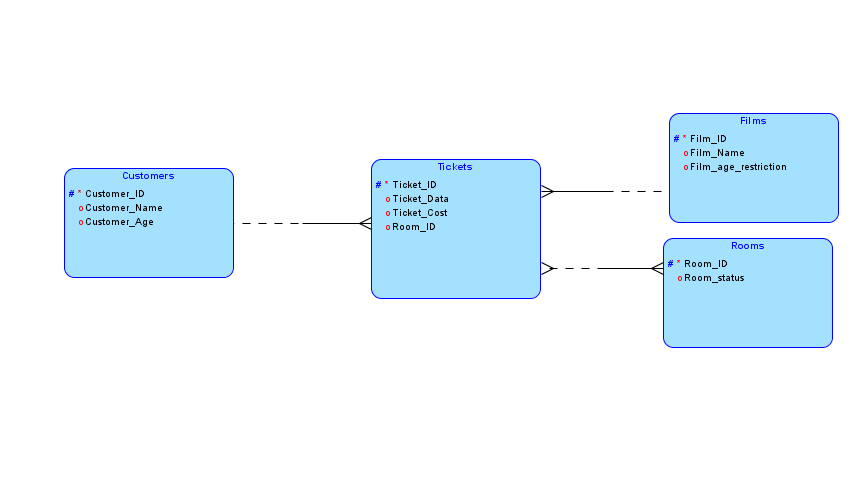


Рисунок А.1 – Логическая модель

# ПРИЛОЖЕНИЕ Б

Реляционная модель

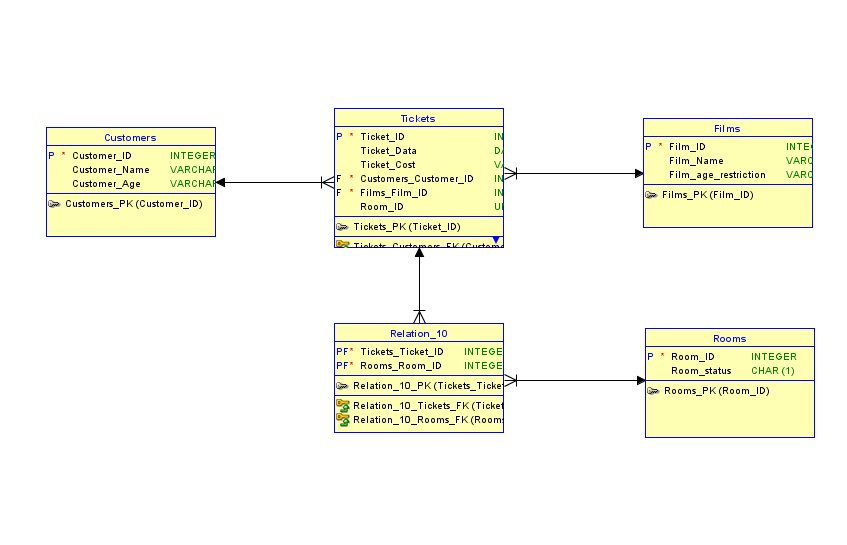


Рисунок Б.1 – Логическая модель

ПРИЛОЖЕНИЕ B

Код создания таблиц:

CREATE TABLE customers (

customer\_id INTEGER NOT NULL,

customer\_name VARCHAR,

customer\_age VARCHAR

);

ALTER TABLE customers ADD CONSTRAINT customers\_pk PRIMARY KEY ( customer\_id );

CREATE TABLE films (

film\_id INTEGER NOT NULL,

film\_name VARCHAR,

film\_age\_restriction VARCHAR

);

ALTER TABLE films ADD CONSTRAINT films\_pk PRIMARY KEY ( film\_id );

CREATE TABLE relation\_10 (

tickets\_ticket\_id INTEGER NOT NULL,

rooms\_room\_id INTEGER NOT NULL

);

ALTER TABLE relation\_10 ADD CONSTRAINT relation\_10\_pk PRIMARY KEY ( tickets\_ticket\_id,

rooms\_room\_id );

CREATE TABLE rooms (

room\_id INTEGER NOT NULL,

room\_status CHAR(1)

);

ALTER TABLE rooms ADD CONSTRAINT rooms\_pk PRIMARY KEY ( room\_id );

CREATE TABLE tickets (

ticket\_id INTEGER NOT NULL,

ticket\_data DATE,

ticket\_cost VARCHAR,

customers\_customer\_id INTEGER NOT NULL,

films\_film\_id INTEGER NOT NULL,

room\_id INTEGER NOT NULL,

);

ALTER TABLE tickets ADD CONSTRAINT tickets\_pk PRIMARY KEY ( ticket\_id );

ALTER TABLE relation\_10

ADD CONSTRAINT relation\_10\_rooms\_fk FOREIGN KEY ( rooms\_room\_id )

REFERENCES rooms ( room\_id );

ALTER TABLE relation\_10

ADD CONSTRAINT relation\_10\_tickets\_fk FOREIGN KEY ( tickets\_ticket\_id )

REFERENCES tickets ( ticket\_id );

ALTER TABLE tickets

ADD CONSTRAINT tickets\_customers\_fk FOREIGN KEY ( customers\_customer\_id )

REFERENCES customers ( customer\_id );

ALTER TABLE tickets

ADD CONSTRAINT tickets\_films\_fk FOREIGN KEY ( films\_film\_id )

REFERENCES films ( film\_id );

Код вставки данных в таблицы:

INSERT INTO [dbo].[Customers] ([Customer\_Name], [Customer\_Age])

VALUES

('Иван Иванов', '12'),

('Елена Смирнова', '12'),

('Александр Кузнецов', '23'),

('Анна Петрова', '34'),

('Дмитрий Соколов', '45'),

('Екатерина Попова', '67'),

('Михаил Новиков', '7'),

('Ольга Морозова', '90'),

('Игорь Козлов', '16'),

('Светлана Иванова', '18'),

INSERT INTO [dbo].[Films] ([Film\_Name], [Film\_age\_restriction])

VALUES

('Рыг ругулькин', '12'),

('Нашествие впо', '13'),

('Сбэу комар', '14'),

('Никита никита', '18'),

('Крысы Д2', '21');

INSERT INTO [dbo].[Tickets] ([Ticket\_data], [Ticket\_Cost])

VALUES

('2023-12-06', '90'),

('2023-12-07', '444'),

('2023-12-08', '333'),

('2023-12-09', '422'),

('2023-12-10', '234'),

('2023-12-11', '789'),

('2023-12-12', '444'),

('2023-12-13', '344'),

('2023-12-14', '233'),

('2023-12-15', '123'),

INSERT INTO [dbo].[Rooms] ([Room\_status] )

VALUES

(1),

(0),

(0),

(1),

(1),

Код создания хранимых процедур:

CREATE PROCEDURE CreateRecord

@param1 int,

@param2 int

AS

BEGIN

INSERT INTO tickets (customer\_id, film\_id)

VALUES (@param1, @param2)

END

--2

CREATE PROCEDURE ReadRecord

@id int

AS

BEGIN

SELECT \* FROM tickets

WHERE ticket\_id = @id

END

--3

CREATE PROCEDURE UpdateRecord

@id int,

@param1 varchar(50),

@param2 varchar(50)

AS

BEGIN

UPDATE tickets

SET ticket\_cost = @param1, customer\_id = @param2

WHERE customer\_id = @id

END

Код создания триггеров:

CREATE TRIGGER LogTrigger

ON tickets

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted)

BEGIN

INSERT INTO tickets(ticket\_id, ticket\_cost, ticket\_data)

VALUES ('YourTable', 'Insert/Update', GETDATE())

END

ELSE

BEGIN

INSERT INTO tickets(ticket\_id, ticket\_cost,ticket\_data)

VALUES ('YourTable', 'Delete', GETDATE())

END

END

CREATE TRIGGER UniqueValueTrigger

ON tickets

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT ticket\_id, COUNT(\*) FROM tickets GROUP BY ticket\_id HAVING COUNT(\*) > 1)

BEGIN

RAISERROR ('Duplicate values are not allowed', 16, 1)

ROLLBACK TRANSACTION

END

END

CREATE TRIGGER UpdateTimestampTrigger

ON tickets

AFTER UPDATE

AS

BEGIN

UPDATE tickets

SET ticket\_data = GETDATE()

FROM tickets

INNER JOIN rooms ON ticket\_id = customer\_id

END

Код создания представлений:

CREATE VIEW [dbo].[MAX и MIN цена]

AS

SELECT MIN(ticket\_Cost) AS Минимальная\_цена, MAX(ticket\_Cost) AS Максимальная\_цена

FROM dbo.[Tickets];

CREATE VIEW [dbo].[Занятые кинозалы]

AS

SELECT dbo.[Room\_status]

FROM dbo.Rooms

WHERE (dbo.[Room\_status] = 'True');

CREATE VIEW [dbo].[Кол-во кинозалов]

AS

SELECT Room\_status, COUNT(\*) AS 'Кол-во'

FROM dbo.[Rooms]

WHERE (Room\_status IN ('True', 'False'));

Код создания ролей и пользователей:

CREATE USER [Bin2] FOR LOGIN [Bin2] WITH DEFAULT\_SCHEMA=[db\_datawriter]

GO

CREATE USER [lesnik] FOR LOGIN [lesnik] WITH DEFAULT\_SCHEMA=[dbo]

GO

ПРИЛОЖЕНИЕ Г

Код создания приложения:

Код Form1:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MaterialSkin.Controls;

using MaterialSkin;

namespace kursRab

{

public partial class Form1 : MaterialForm

{

static public string User;

public Form1()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

materialSkinManager.ColorScheme = new ColorScheme(Primary.Grey800, Primary.Grey900, Primary.BlueGrey900, Accent.Red700, TextShade.WHITE);

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void materialButton1\_Click(object sender, EventArgs e)

{

if (materialTextBox21.Text == "admin" && materialTextBox22.Text == "admin")

{

User = "admin";

Form2 f2 = new Form2();

f2.Show();

this.Hide();

}

else if (materialTextBox21.Text == "user" && materialTextBox22.Text == "user")

{

User = "user";

Form2 f2 = new Form2();

f2.Show();

this.Hide();

}

else

{

MessageBox.Show("Неправильный логин или пароль", "Отказано в доступе", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void materialTextBox22\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

if (materialTextBox21.Text == "admin" && materialTextBox22.Text == "admin")

{

Form2 f2 = new Form2();

f2.Show();

this.Hide();

}

else if (materialTextBox21.Text == "user" && materialTextBox22.Text == "user")

{

Form2 f2 = new Form2();

f2.Show();

this.Hide();

}

else

{

MessageBox.Show("Неправильный логин или пароль", "Отказано в доступе", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void materialTextBox21\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

if (materialTextBox21.Text == "admin" && materialTextBox22.Text == "admin")

{

Form2 f2 = new Form2();

f2.Show();

this.Hide();

}

else if (materialTextBox21.Text == "user" && materialTextBox22.Text == "user")

{

Form2 f2 = new Form2();

f2.Show();

this.Hide();

}

else

{

MessageBox.Show("Неправильный логин или пароль", "Отказано в доступе", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

Код Form2:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MaterialSkin.Controls;

using MaterialSkin;

using OfficeOpenXml;

using OfficeOpenXml.Style;

using System.IO;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace kursRab

{

public partial class Form2 : MaterialForm

{

public Form2()

{

InitializeComponent();

}

private void Form2\_Load(object sender, EventArgs e)

{

/\*

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet2.logpass". При необходимости она может быть перемещена или удалена.

this.logpassTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet2.logpass);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.Фильмы\_для\_лиц\_старше\_17\_лет". При необходимости она может быть перемещена или удалена.

this.фильмы\_для\_лиц\_старше\_17\_летTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.Фильмы\_для\_лиц\_старше\_17\_лет);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.самый\_первый\_купленный\_билет". При необходимости она может быть перемещена или удалена.

this.самый\_первый\_купленный\_билетTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.самый\_первый\_купленный\_билет);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.Посетители\_старше\_18\_лет". При необходимости она может быть перемещена или удалена.

this.посетители\_старше\_18\_летTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.Посетители\_старше\_18\_лет);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.Минимальная\_стоимость\_билета". При необходимости она может быть перемещена или удалена.

this.минимальная\_стоимость\_билетаTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.Минимальная\_стоимость\_билета);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.Максимальная\_стоимость\_билета". При необходимости она может быть перемещена или удалена.

this.максимальная\_стоимость\_билетаTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.Максимальная\_стоимость\_билета);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.\_Кол\_во\_сеансов\_в\_1\_зале\_за\_все\_время". При необходимости она может быть перемещена или удалена.

this.кол\_во\_сеансов\_в\_1\_зале\_за\_все\_времяTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.\_Кол\_во\_сеансов\_в\_1\_зале\_за\_все\_время);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.\_Кол\_во\_свободных\_залов". При необходимости она может быть перемещена или удалена.

this.кол\_во\_свободных\_заловTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.\_Кол\_во\_свободных\_залов);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.\_Кол\_во\_купленных\_билетов\_дороже\_500\_рублей". При необходимости она может быть перемещена или удалена.

this.кол\_во\_купленных\_билетов\_дороже\_500\_рублейTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.\_Кол\_во\_купленных\_билетов\_дороже\_500\_рублей);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.\_Кол\_во\_занятых\_залов". При необходимости она может быть перемещена или удалена.

this.кол\_во\_занятых\_заловTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.\_Кол\_во\_занятых\_залов);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet1.Билеты\_купленные\_за\_20\_лет". При необходимости она может быть перемещена или удалена.

this.билеты\_купленные\_за\_20\_летTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet1.Билеты\_купленные\_за\_20\_лет);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet.tickets". При необходимости она может быть перемещена или удалена.

this.ticketsTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet.tickets);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet.rooms". При необходимости она может быть перемещена или удалена.

this.roomsTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet.rooms);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet.customers". При необходимости она может быть перемещена или удалена.

this.customersTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet.customers);

// TODO: данная строка кода позволяет загрузить данные в таблицу "ip521\_Sofronov\_KRDataSet.films". При необходимости она может быть перемещена или удалена.

this.filmsTableAdapter.Fill(this.ip521\_Sofronov\_KRDataSet.films);

\*/

if (Form1.User == "user")

{

dataGridView1.ReadOnly = true;

dataGridView2.ReadOnly = true;

dataGridView3.ReadOnly = true;

dataGridView4.ReadOnly = true;

groupBox1.Visible = false;

groupBox2.Visible = false;

}

else

if (Form1.User == "admin")

{

}

}

private void Form2\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void таблицыToolStripMenuItem\_Click(object sender, EventArgs e)

{

tabControl1.Visible = true;

tabControl2.Visible = false;

}

private void представленияToolStripMenuItem\_Click(object sender, EventArgs e)

{

tabControl1.Visible = false;

tabControl2.Visible = true;

}

private void сохранитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

filmsBindingSource.EndEdit();

filmsTableAdapter.Update(ip521\_Sofronov\_KRDataSet.films);

customersBindingSource.EndEdit();

customersTableAdapter.Update(ip521\_Sofronov\_KRDataSet.customers);

roomsBindingSource.EndEdit();

roomsTableAdapter.Update(ip521\_Sofronov\_KRDataSet.rooms);

ticketsBindingSource.EndEdit();

ticketsTableAdapter.Update(ip521\_Sofronov\_KRDataSet.tickets);

}

private void фильтрToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void materialMaskedTextBox1\_Click(object sender, EventArgs e)

{

}

private void materialComboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void materialButton1\_Click\_1(object sender, EventArgs e)

{

filmsBindingSource.Filter = string.Empty;

roomsBindingSource.Filter = string.Empty;

ticketsBindingSource.Filter = string.Empty;

customersBindingSource.Filter = string.Empty;

customersBindingSource.RemoveFilter();

ticketsBindingSource.RemoveFilter();

roomsBindingSource.RemoveFilter();

filmsBindingSource.RemoveFilter();

materialTextBox3.Clear();

materialTextBox5.Clear();

}

private void materialButton2\_Click(object sender, EventArgs e)

{

string columnName = materialTextBox3.Text;

string filterValue = materialTextBox5.Text;

string a = materialComboBox2.SelectedItem.ToString();

if (materialSwitch5.Checked == false)

{

switch (a)

{

case "Films": filmsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "')"; break;

case "Товары": customersBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "')"; break;

case "Tickets": ticketsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "')"; break;

case "Rooms": roomsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "')"; break;

default: MessageBox.Show("Таблица не найдена", "Ошибка"); break;

}

}

else

{

if (materialSwitch1.Checked == false)

{

switch (a)

{

case "Films": filmsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "') and " + materialTextBox4.Text + " =('" + materialTextBox2.Text + "')"; break;

case "Customers": customersBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "') and " + materialTextBox4.Text + " =('" + materialTextBox2.Text + "')"; break;

case "Tickets": ticketsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "') and " + materialTextBox4.Text + " =('" + materialTextBox2.Text + "')"; break;

case "Rooms": roomsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "') and " + materialTextBox4.Text + " =('" + materialTextBox2.Text + "')"; break;

default: MessageBox.Show("Таблица не найдена", "Ошибка"); break;

}

}

else

{

switch (a)

{

case "Films": filmsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "') or " + materialTextBox4.Text + " =('" + materialTextBox2.Text + "')"; break;

case "Customers": customersBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "') or " + materialTextBox4.Text + " =('" + materialTextBox2.Text + "')"; break;

case "Tickets": ticketsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "') or " + materialTextBox4.Text + " =('" + materialTextBox2.Text + "')"; break;

case "Rooms": roomsBindingSource.Filter = materialTextBox3.Text + " = ('" + materialTextBox5.Text + "') or " + materialTextBox4.Text + " =('" + materialTextBox2.Text + "')"; break;

default: MessageBox.Show("Таблица не найдена", "Ошибка"); break;

}

}

}

}

string filepath = "";

private void exportToExcel(DataGridView dataGridView, string filepath)

{

try

{

if (materialComboBox1.SelectedItem != null)

{

string tableName = materialComboBox1.SelectedItem.ToString();

ExcelPackage.LicenseContext = OfficeOpenXml.LicenseContext.NonCommercial;

using (ExcelPackage excelPackage = new ExcelPackage())

{

ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add(tableName);

for (int column = 0; column < dataGridView.ColumnCount; column++)

{

worksheet.Cells[1, column + 1].Value = dataGridView.Columns[column].HeaderText;

}

for (int row = 0; row < dataGridView.Rows.Count; row++)

{

for (int column = 0; column < dataGridView.ColumnCount; column++)

{

worksheet.Cells[row + 2, column + 1].Value = dataGridView.Rows[row].Cells[column].Value;

}

}

using (ExcelRange range = worksheet.Cells[1, 1, dataGridView.Rows.Count + 1, dataGridView.ColumnCount])

{

range.Style.HorizontalAlignment = ExcelHorizontalAlignment.Center;

range.AutoFitColumns();

}

FileInfo excelFile = new FileInfo(filepath);

excelPackage.SaveAs(excelFile);

}

MessageBox.Show($"Таблица {tableName} успешно экспортирована!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

MessageBox.Show("Выберите таблицу для экспорта.", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch

{

MessageBox.Show("При экспорте таблицы произошла ошибка", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void materialButton3\_Click(object sender, EventArgs e)

{

string table = materialComboBox1.Text;

switch (table)

{

case "Films":

filepath = $"{materialTextBox1.Text}.xlsx";

exportToExcel(dataGridView1, filepath);

break;

case "Customers":

filepath = $"{materialTextBox1.Text}.xlsx";

exportToExcel(dataGridView2, filepath);

break;

case "Tickets":

filepath = $"{materialTextBox1.Text}.xlsx";

exportToExcel(dataGridView4, filepath);

break;

case "Rooms":

filepath = $"{materialTextBox1.Text}.xlsx";

exportToExcel(dataGridView3, filepath);

break;

default:

MessageBox.Show("Такой таблицы не существует!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

break;

}

}

private void materialSwitch5\_CheckedChanged(object sender, EventArgs e)

{

if (materialSwitch5.Checked == true)

{

groupBox5.Visible = true;

}

else

{

groupBox5.Visible = false;

}

}

private void materialSwitch1\_CheckedChanged(object sender, EventArgs e)

{

}

private void tabPage2\_Click(object sender, EventArgs e)

{

}

}

}