**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc153730958)

[1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 5](#_Toc153730959)

[1.1 Анализ предметной области 5](#_Toc153730960)

[1.2 Постановка задачи 5](#_Toc153730961)

[2 РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 7](#_Toc153730962)

[2.1 Построение диаграммы «сущность-связь» в нотации П.Чена 7](#_Toc153730963)

[2.2 Построение модели основанной на ключах 9](#_Toc153730964)

[2.3 Построение полной атрибутивной модели 10](#_Toc153730965)

[3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 12](#_Toc153730966)

[3.1 Выбор аппаратной и программной платформы для реализации базы данных 12](#_Toc153730967)

[3.2 Реализация базы данных 13](#_Toc153730968)

[3.3 Тестирование базы данных 13](#_Toc153730969)

[3.4 Разграничения прав доступа 18](#_Toc153730970)

[4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 19](#_Toc153730971)

[4.1 Обоснования выбора языка программирования 19](#_Toc153730972)

[4.2 Разработка интерфейса пользователя 20](#_Toc153730973)

[ЗАКЛЮЧЕНИЕ 25](#_Toc153730974)

[СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ 26](#_Toc153730975)

[ПРИЛОЖЕНИЕ А 27](#_Toc153730976)

[SQL скрипт создания базы данных 27](#_Toc153730977)

[ПРИЛОЖЕНИЕ Б 29](#_Toc153730978)

[Код клиентского приложения 29](#_Toc153730979)

# **ВВЕДЕНИЕ**

На сегодняшний день использование баз данных является неотъемлемой частью практически любой информационной системы. С развитием баз данных тесно связано автоматизированное управление данными и рост числа пользователей, что приводит к появлению более мощных систем управления базами данных (СУБД), обеспечивающих высокую отказоустойчивость, защиту информации и производительность.

Целью данного курсового проекта является создание и реализация базы данных для издательского дома. Для достижения этой цели были выполнены следующие этапы:

Анализ предметной области, включающий определение пользовательских систем, функциональных требований, основных сущностей и их взаимосвязей.

Разработка логической схемы базы данных, включая построение ER-диаграммы, создание модели с использованием ключей и нормализацию отношений.

Реализация базы данных, включающая выбор и обоснование выбора СУБД, разработку физической схемы, создание и тестирование базы данных, а также разграничение прав доступа.

Исследование информационных параметров базы данных, включающее расчет значений некоторых параметров БД и приложения в целом.

Разработка клиентского приложения, включая обоснование выбора языка программирования, создание приложения, описание интерфейса пользователя и тестирование системы.

# **1 АНАЛИТИЧЕСКАЯ ЧАСТЬ**

## **1.1 Анализ предметной области**

Издательский дом – это организация, специализирующаяся на производстве и распространении печатной продукции, такой как книги, журналы, газеты и другие издания. Основными направлениями работы издательского дома являются: производство изданий; распространение литературной продукции; управление информацией о книгах и авторах.

При работе с базой данных издательского дома редактор должен иметь возможность решать следующие задачи:

* вести учет изданий;
* управлять информацией об авторах и их произведениях;
* контролировать распространение и продажи книг.

## **1.2 Постановка задачи**

Исходя из того, что пользователем приложения будут как администраторы издательского дома, так и заказчики, нужно разграничить доступ.

Определим следующие функциональные требования для приложения:

* ограничение доступа к возможностям заказчиков;
* хранение актуальной информации о количестве книг;
* возможность добавления, удаления и редактирования данных о книгах;
* возможность заказов;
* интуитивно понятный интерфейс.

Согласно вышеперечисленным функциональным требованиям, выделим сущности. Сущность – любой различимый объект, информацию о котором необходимо хранить в базе данных.

Для нашей задачи можно определить следующие сущности, которые в дальнейшем будут использоваться для построения внешней модели проектируемой базы данных: отзывы, книги, заказы, заказчики, писатели.

Каждой из перечисленной сущности соответствует свой набор атрибутов:

* Отзывы (ID отзыва, ID книги, дата, оценка);
* Книги (ID книги, ID автора, количество, название, жанр, цена, год);
* Писатели (ID писателя, дата рождения, фамилия, имя);
* Заказы (ID заказа, ID книги, ID заказчика, количество книг, статус, дата);
* Заказчики (ID заказчика, фамилия, имя, адрес).

Таким образом, сформировав некоторые требования к разработке логической базы данных, которую необходимо будет реализовать в виде клиентского приложения, можно переходить к проектированию логической модели БД.

# **2 РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ**

## **2.1 Построение диаграммы «сущность-связь» в нотации П.Чена**

После проведения анализа предметной области и выделения основных сущностей возникла возможность создать диаграмму "сущность-связь" в нотации П.Чена. Однако перед началом работы над диаграммой необходимо определить отношения между выделенными сущностями.

Была разработана сетевая структура, основанная на ключевых элементах предметной области. Разработанная структура представлена на рисунке 1.

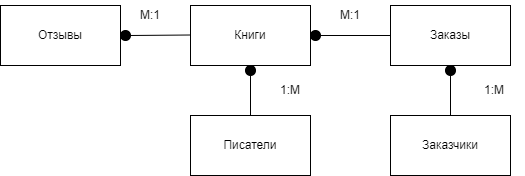


Рисунок 1 – Сетевая структура

Нормализация не может быть произведена в простой сетевой структуре, поэтому необходимо привести сетевую модель к иерархической.

Иерархическая (древовидная) структура не позволяет у одного элемента иметь несколько родителей. Поэтому, чтобы преобразовать простую сетевую структуру в древовидную, необходимо ввести избыточность, путем дублирования связей, которые являются потомками. Полученная древовидная структура представлена на рисунке 2.

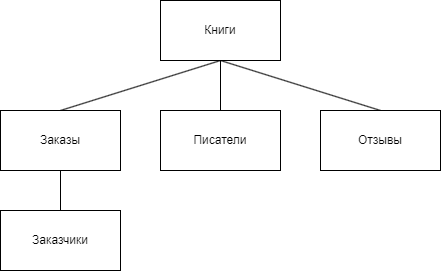


Рисунок 2 – Древовидная структура

Далее на основе прошлых таблиц была синтезирована ER-диаграмма (рисунок 3), отображающая взаимосвязь объектов, а также основные элементы взаимодействующих объектов.



Рисунок 3 – ER диаграмма в нотации П.Чена

## **2.2 Построение модели основанной на ключах**

Основной целью ключевой модели данных является обеспечение всестороннего обзора структур данных и ключевых элементов, необходимых для поддержки определенной области. Эта модель создает контекст, в котором можно разработать подробные модели, адаптированные к конкретным требованиям.

Модель, основанная на ключах, строится на основе ERD (сущность-связь диаграммы), однако она представляет больше деталей, таких как ключевые поля и внешние ключи.

Разработанная модель представлена на рисунке 4.

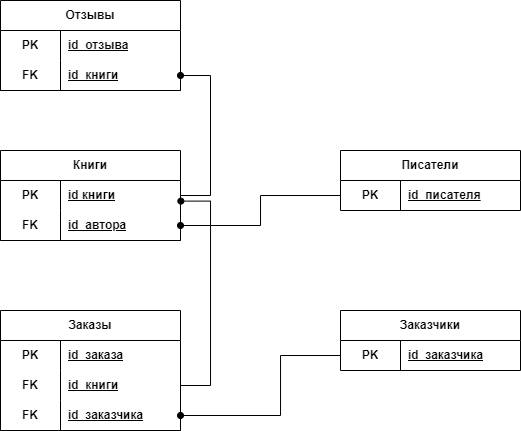


Рисунок 4 – Модель, основанная на ключах

## **2.3 Построение полной атрибутивной модели**

Достижение полной атрибутивной модели достигается путем нормализации отношений до третьей или четвертой нормальной формы. Цель нормализации базы данных заключается в устранении избыточности. Этот процесс включает последовательное приведение схемы базы данных к нормальным формам, каждая из которых устанавливает более строгие требования по сравнению с предыдущей.

После выделения всех сущностей мы проводим проверку на атомарность. Каждая сущность подтверждает свою атомарность, что свидетельствует о том, что наша база данных находится в первой нормальной форме.

Далее мы рассматриваем базу данных на соответствие второй нормальной форме, которая требует, чтобы каждый не ключевой атрибут полностью зависел от первичного ключа. Каждая сущность содержит уникальный первичный ключ, от которого полностью зависят все не ключевые атрибуты, что удовлетворяет условиям второй нормальной формы.

После этого мы анализируем базу данных на соответствие третьей нормальной форме, которая требует отсутствия транзитивных зависимостей. Во всех созданных сущностях такие зависимости отсутствуют, что подтверждает, что наша база данных находится в третьей нормальной форме.

В ходе нормализации была построена полная атрибутивная модель в нотации IDEF1X, которая представлена на рисунке 5.

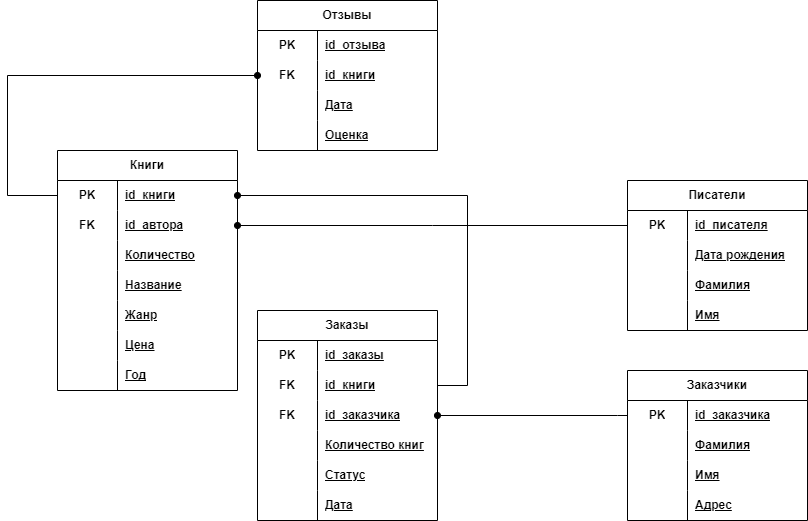


Рисунок 5 – Полная атрибутивная модель

# **3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ**

## **3.1 Выбор аппаратной и программной платформы для реализации базы данных**

Выбор системы управления базами данных представляет собой сложную многопараметрическую задачу и является одним из ключевых этапов при создании приложений баз данных. Выбранная СУБД должна соответствовать как текущим, так и будущим потребностям системы. На данный момент существует множество различных СУБД, однако в данной курсовой работе была выбрана MySQL по следующим причинам:

1. Быстродействие: MySQL является одной из наиболее быстрых баз данных, доступных на рынке.
2. Простота использования: MySQL обладает высокой производительностью и относительной простотой в использовании.
3. Возможности доступа: Сервер MySQL позволяет неограниченному числу пользователей одновременно работать с базой данных.
4. Аппаратная совместимость: MySQL хорошо работает на большинстве операционных систем.
5. Бесплатное использование: MySQL полностью бесплатна, её исходный код открыт, что позволяет разработчикам создавать собственные проекты на основе этой системы соблюдая условия лицензии.

## **3.2 Реализация базы данных**

Так как у нас уже имеется логическая модель базы данных в третьей нормальной форме, а также выбрана СУБД, которая её будет реализовать, преобразуем логическую модель в физическую. На рисунке 6 представлена физическая модель нашей базы данных.

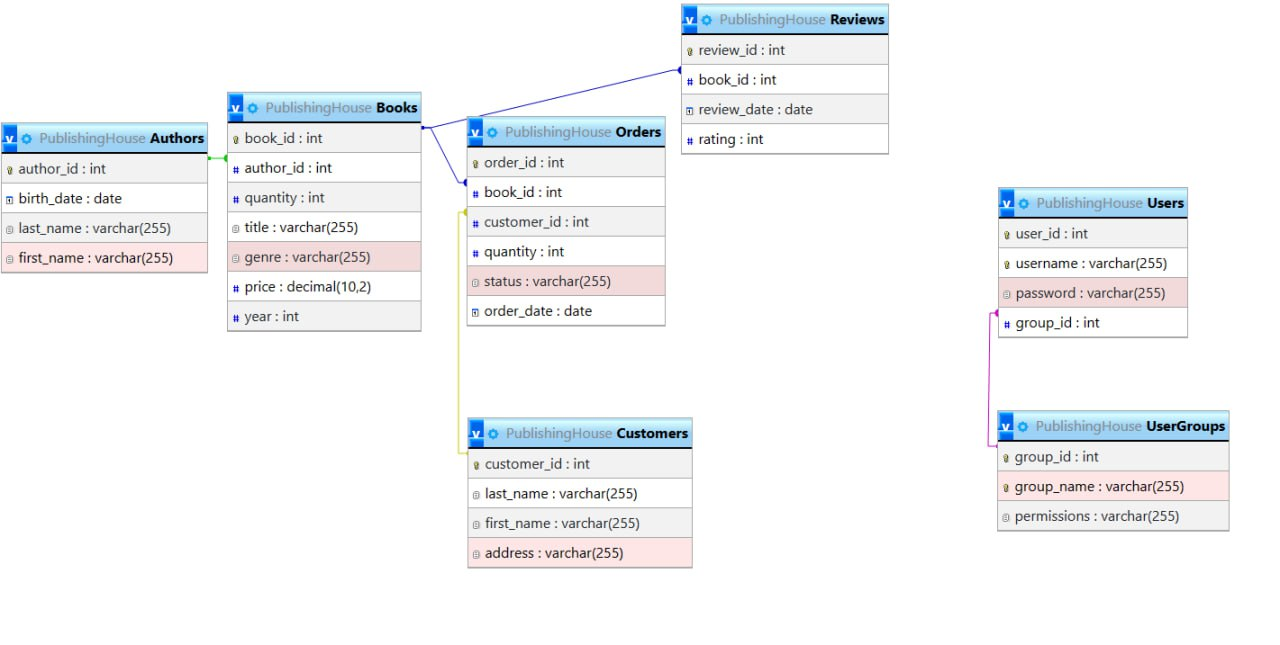


Рисунок 6 – Физическая модель базы данных

## **3.3 Тестирование базы данных**

После создания таблиц и заполнения основными данными необходимо протестировать полученную базу данных. Для этого построим не менее 12 разноплановых запросов и удостоверимся что все разработанные запросы выполняются правильно.

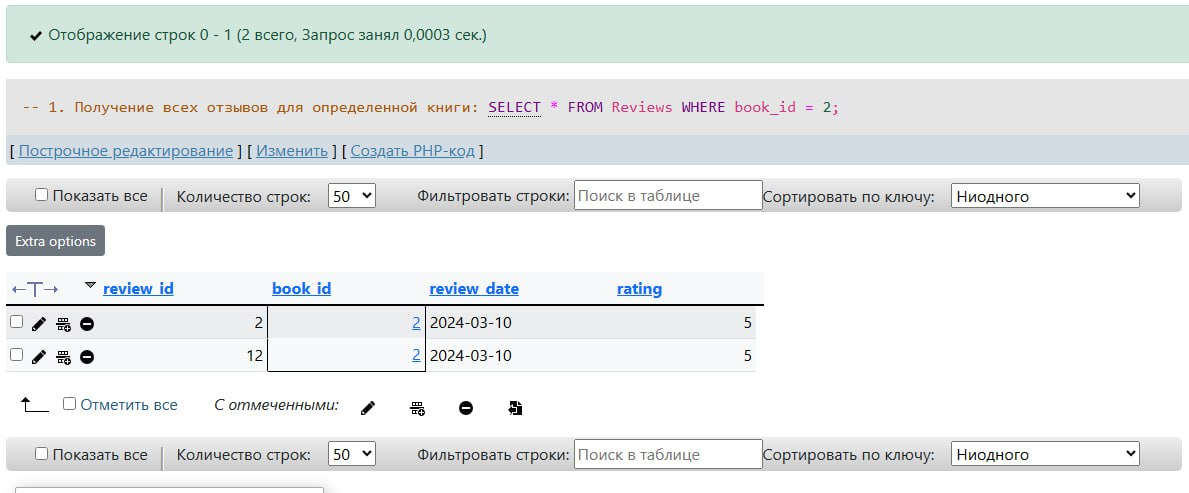


Рисунок 7 – Результат выполнения запроса

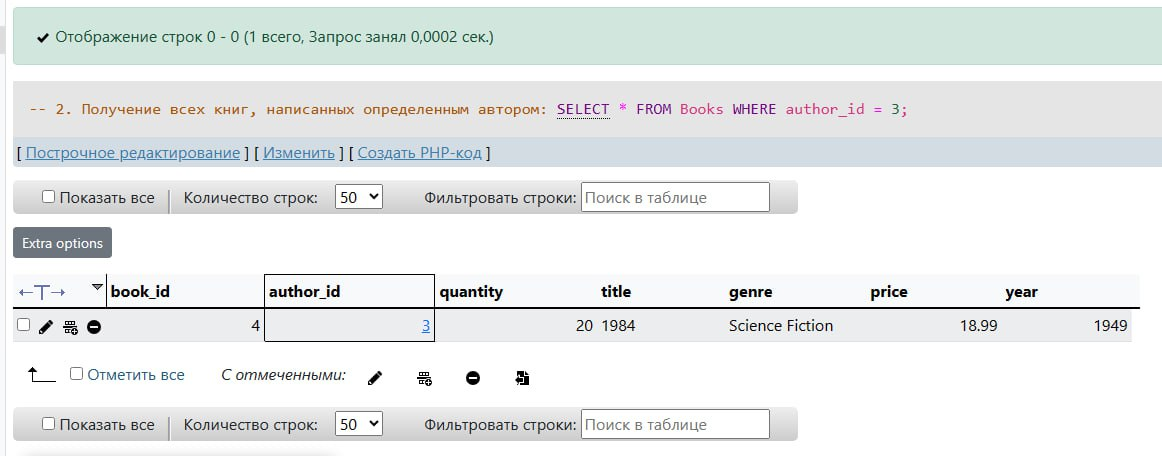


Рисунок 8 – Результат выполнения запроса



Рисунок 9 – Результат выполнения запроса

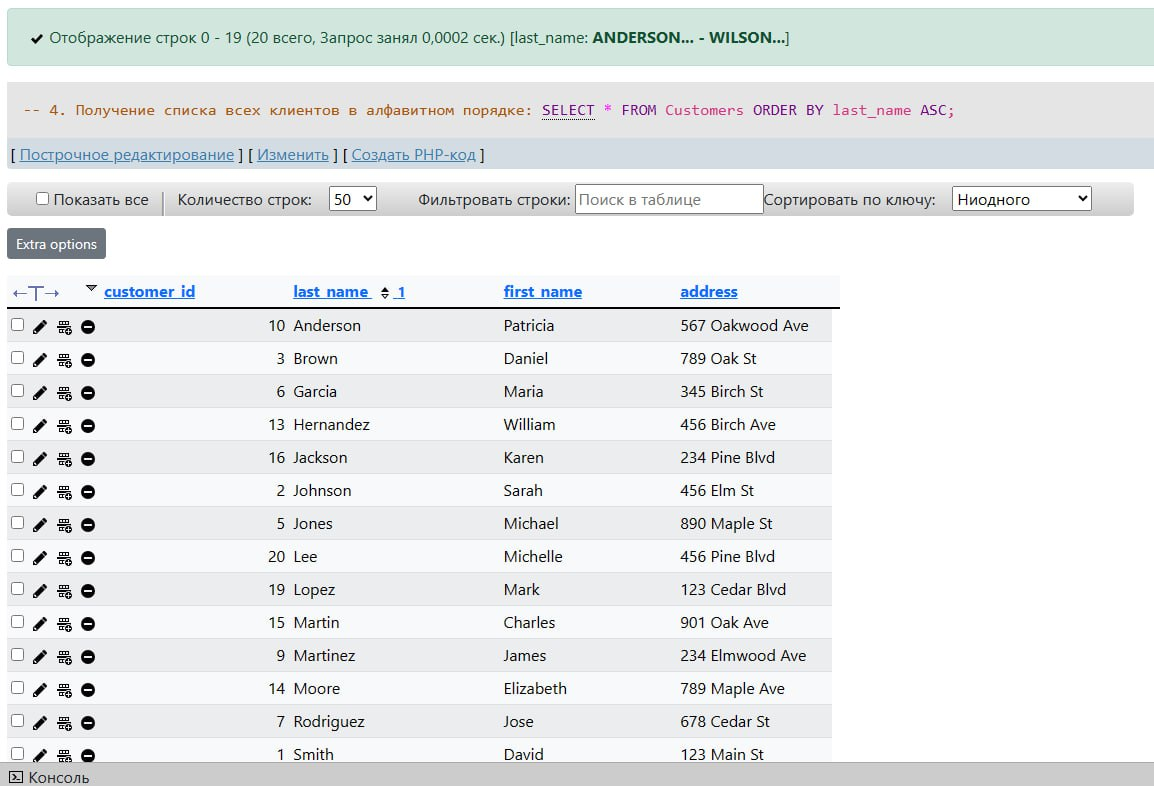


Рисунок 10 – Результат выполнения запроса



Рисунок 11 – Результат выполнения запроса

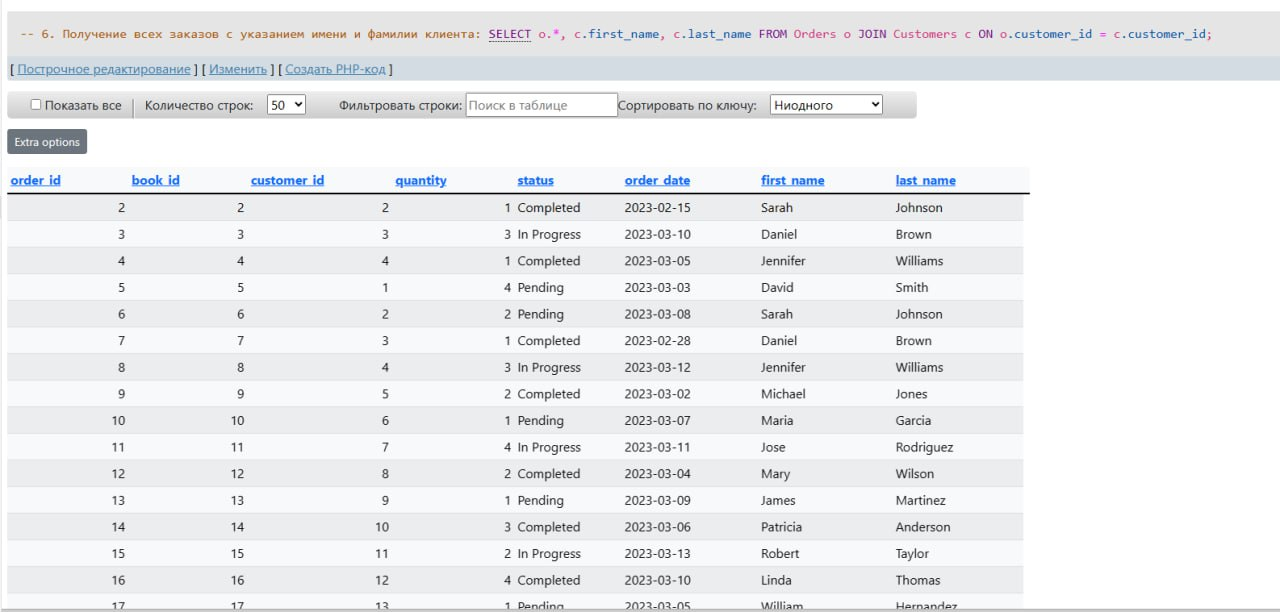


Рисунок 12 – Результат выполнения запроса

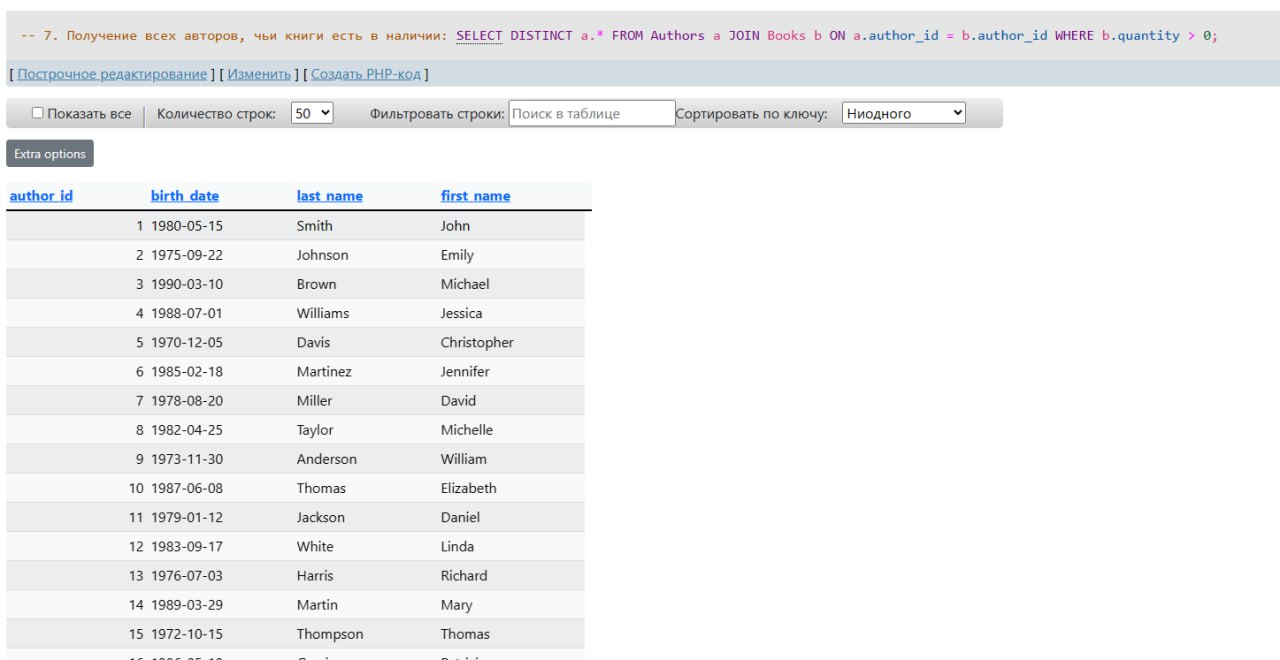


Рисунок 13 – Результат выполнения запроса



Рисунок 14 – Результат выполнения запроса

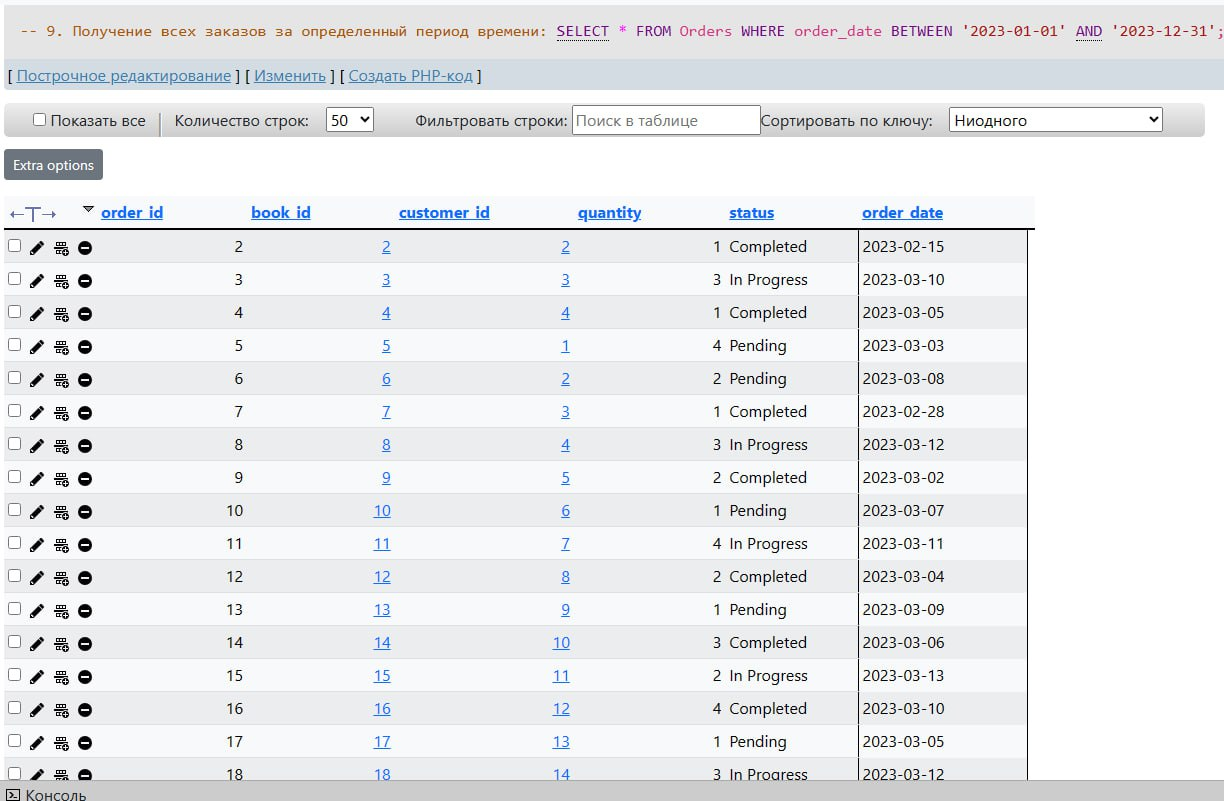


Рисунок 15 – Результат выполнения запроса



Рисунок 16 – Результат выполнения запроса

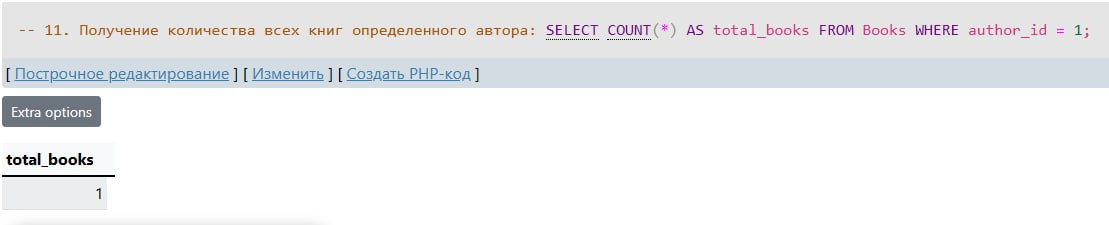


Рисунок 17 – Результат выполнения запроса

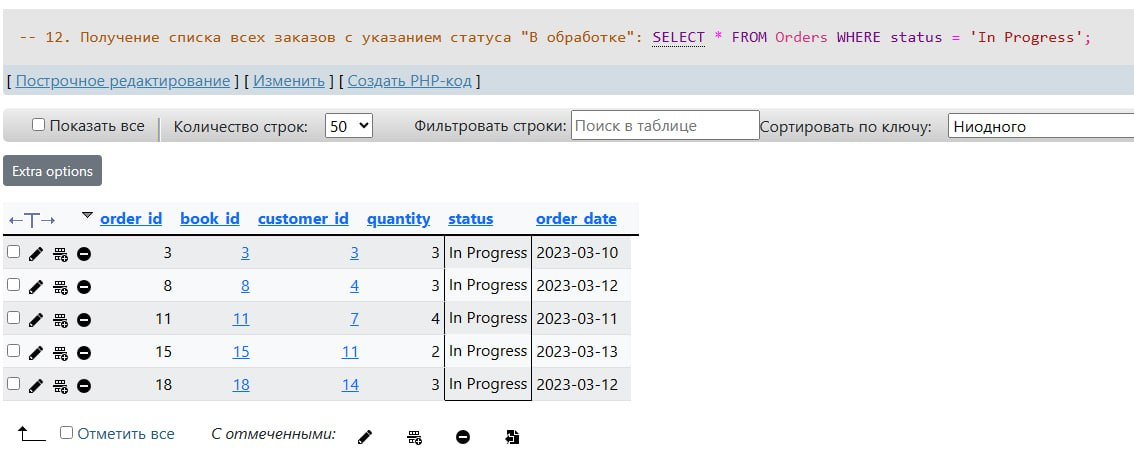


Рисунок 18 – Результат выполнения запроса

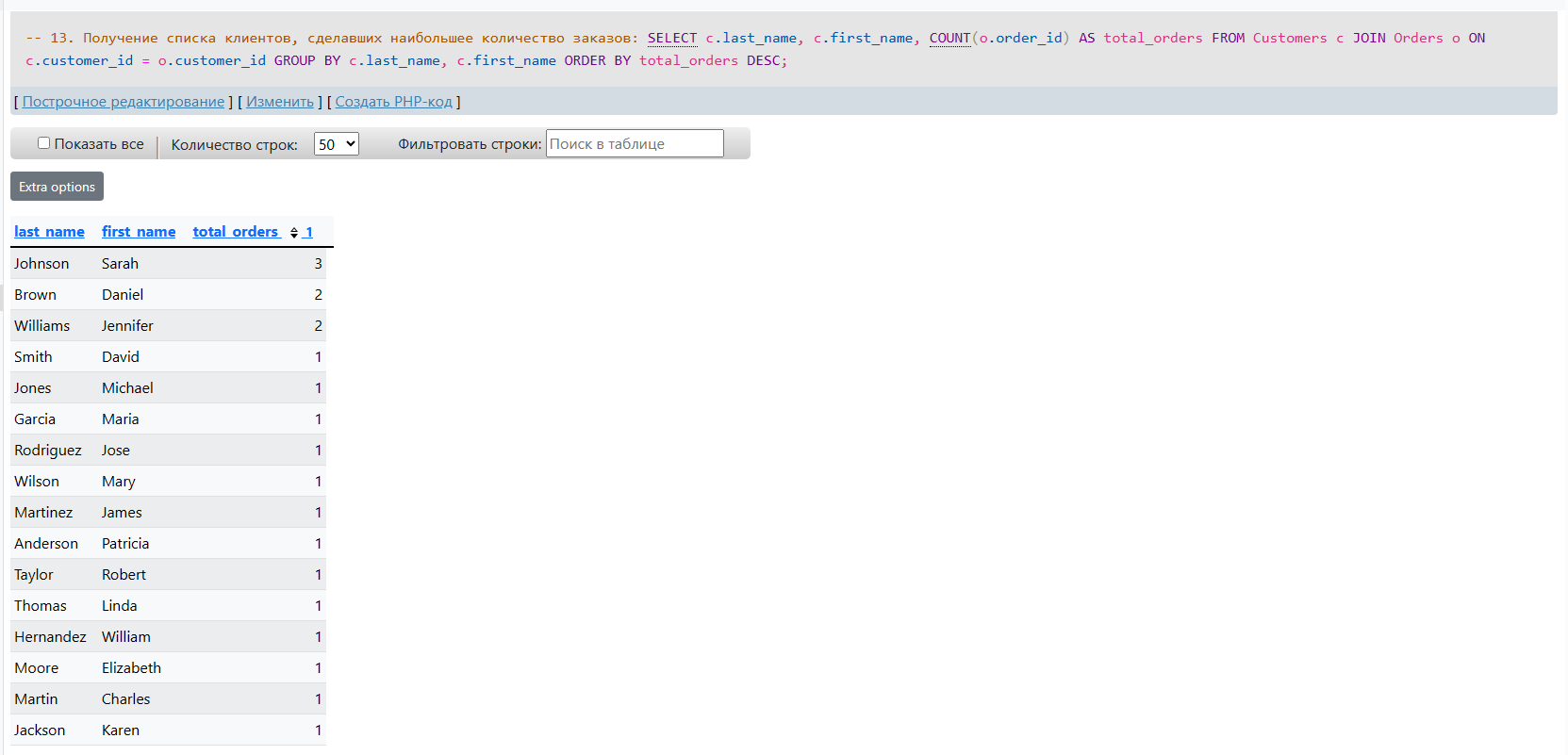
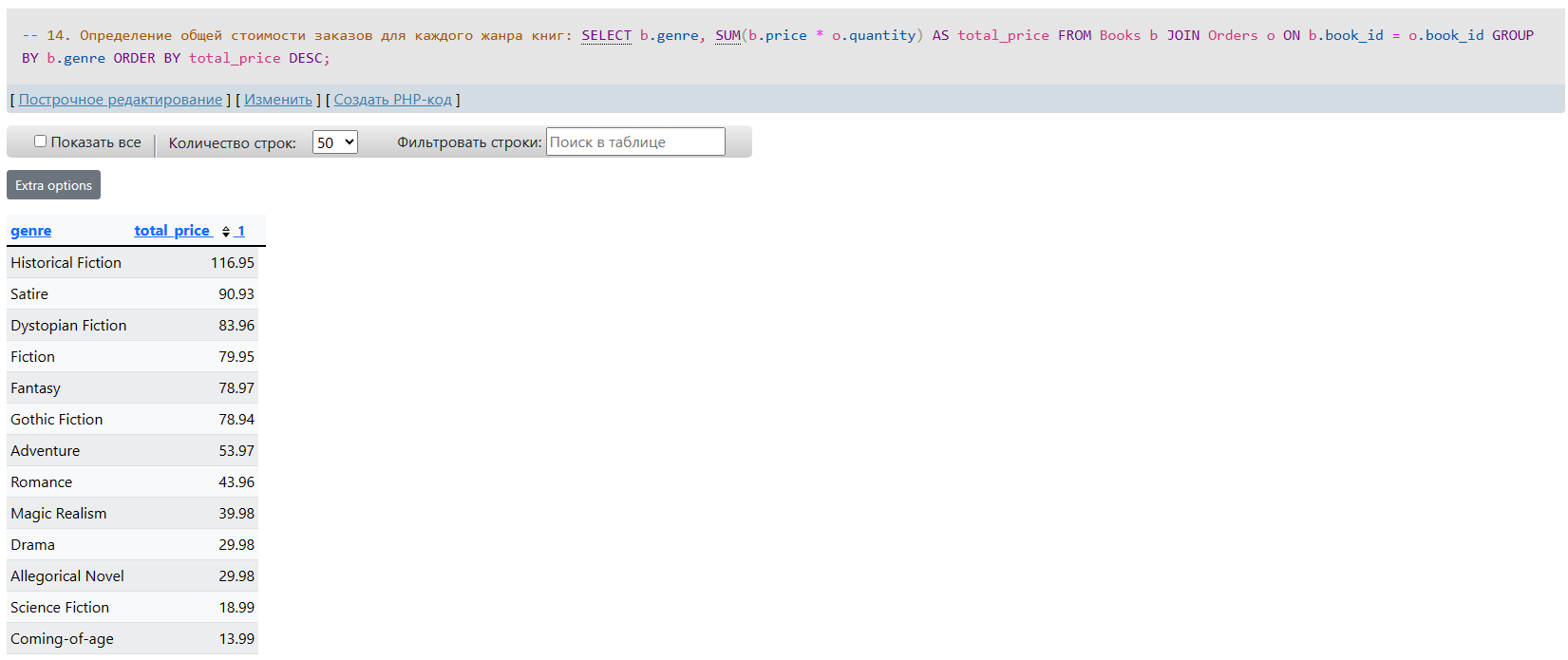
  
Рисунок 19 – Результат выполнения запроса  
  


Рисунок 20 – Результат выполнения запроса

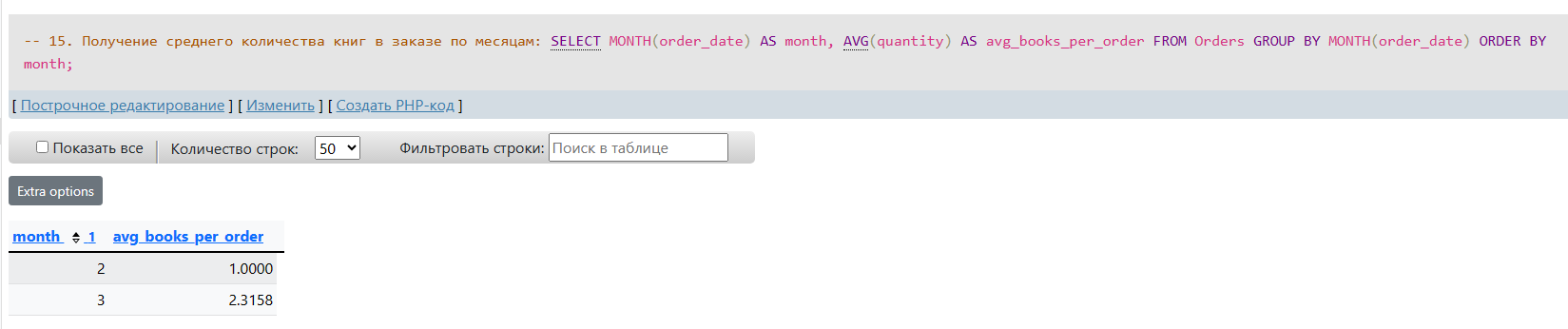


Рисунок 21 – Результат выполнения запроса

## **3.4 Разграничения прав доступа**

Для работы с проектируемой базой данных было выделено 2 пользователя, имеющие различные уровни доступа – это администратор и заказчик.

Администратор управляет базой данных.

Заказчик может посмотреть каталог книг, оформить заказ, просмотреть свои заказы.

# **4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ**

## **4.1 Обоснования выбора языка программирования**

В качестве языка программирования был выбран C#. C# является объектно-ориентированным языком программирования, разработанным компанией Microsoft, и используется для создания разнообразных приложений, включая приложения для платформы Windows, включая Windows Forms.

C# обладает всеми необходимыми средствами для разработки Windows Forms приложений. Он предлагает широкий набор классов и библиотек, которые упрощают создание графического пользовательского интерфейса, обработку событий и взаимодействие с компонентами операционной системы Windows.

Одним из преимуществ C# является его интеграция с платформой .NET, которая предоставляет обширный набор классов и функциональности для разработки приложений. C# также обладает сильной типизацией и безопасностью типов, что способствует созданию надежных и эффективных приложений.

Основным способом создания пользовательского интерфейса в C# для Windows Forms является использование визуального дизайнера, который позволяет размещать и настраивать элементы управления (кнопки, текстовые поля, списки и др.) на форме и связывать их с кодом для обработки событий и реализации функциональности приложения.

## 

## **4.2 Разработка интерфейса пользователя**

На рисунке 19 представлено окно авторизации.

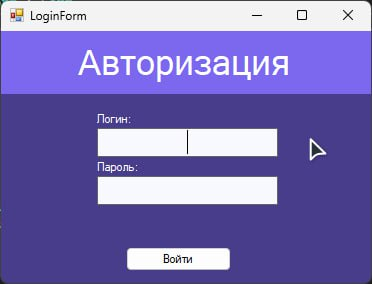


Рисунок 22 – Окно авторизации

Если данные были введены некорректно, то приложение предотвратит попытку входа и выведет соответствующее сообщение пользователю

Далее пользователь попадает на главную страницу приложения, где он может посмотреть каталог книг (рисунок 20).



Рисунок 23 – Главная страница

Также в этом окне заказчик может заказать книги. В программе предусмотрена ситуация, когда пользователь может попытаться купить больше единиц, товара, чем есть на складе (рисунок 22).

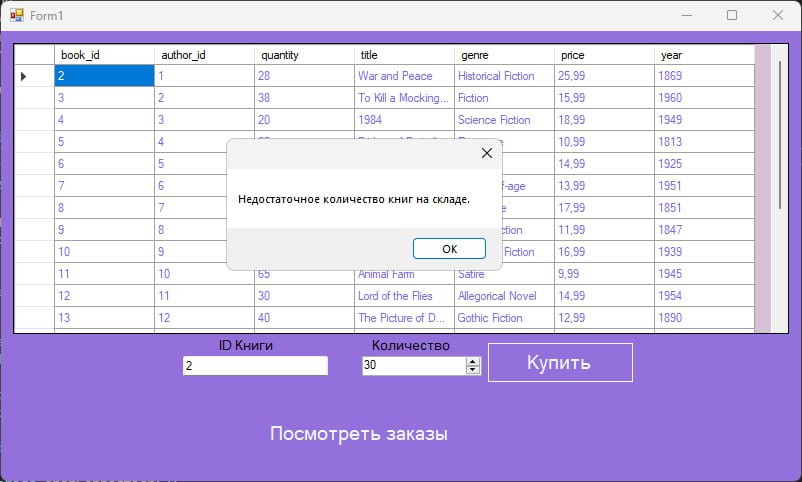


Рисунок 24 – Попытка заказать 30 книг с ID 2

На рисунке 23 изображена успешная покупка книг. Как видно, программа сразу же обновила количество книг в таблице.

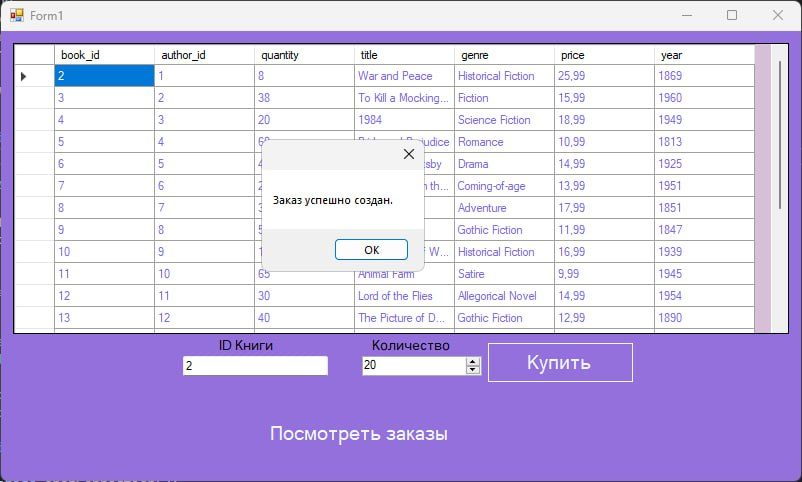


Рисунок 25 – Создание заказа

Отсюда пользователь также может перейти в окно с заказами, где на экран будут выведены все его заказы (рисунок 24).

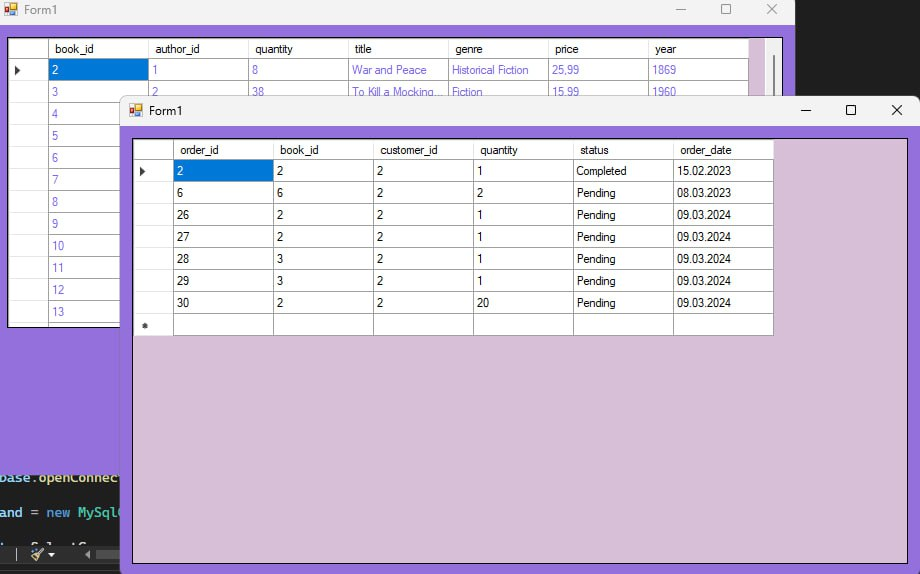


Рисунок 26 – Список заказов

Вернемся обратно в окно авторизации и зайдем с аккаунта администратора. Первым делом мы попадаем в меню администратора (рисунок 25). Здесь можно прочитать и удалить отзывы, добавить и удалить книги, добавить и удалить покупателей.

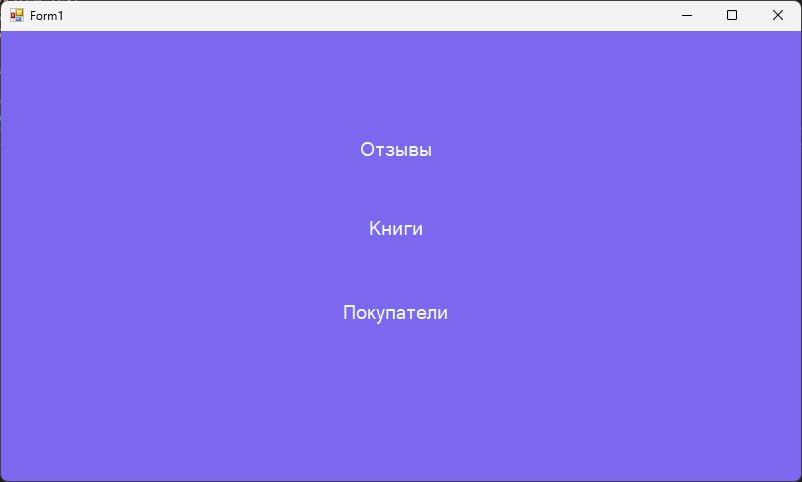


Рисунок 27 – Панель выбора

На примере вкладки «Книги» рассмотрим работу меню администратора. Здесь администратор может добавлять (рисунок 30-31) и удалять книги. При этом, если книга уже добавлена, то при повторном ее добавлении будет меняться только количество товара. Также при удалении и добавлении таблица будет сразу же обновляться.



Рисунок 28 – Добавление книги



Рисунок 29 – Обновление таблицы после закрытие оповещения

# **ЗАКЛЮЧЕНИЕ**

В ходе данного курсового проекта была разработана реляционная база данных, посвященная теме "База данных издательского дома". Написанное на языке C# приложение позволяет взаимодействовать с базой данных, а также был создан простой и удобный интерфейс для работы с ней.

Произведена нормализация логической модели базы данных до третьей нормальной формы, что привело к уменьшению избыточности данных и обеспечило их целостность. Была также разработана физическая модель базы данных.

В процессе выполнения курсового проекта были закреплены навыки проектирования логической модели базы данных, проведения нормализации данных, создания физической модели и разработки программного обеспечения.

В результате проекта был создан программный продукт для использования издательскими домами. Данный продукт представляет собой систему учета изданий и авторов, обеспечивая эффективное управление издательскими процессами.

# **СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ**

1. Линн Бейли, Майкл Моррисон. Изучаем MySQL. - Москва: Эксмо, 2010. - 768 с.

2. Гарсиа-Молина Г. Системы баз данных. Полный курс: Пер. с англ./Г.Гарсиа-Молина, Д.Д.Ульман, Д.Уидом. – М.: Издательский дом «Вильямс», 2004. – 1088с.

3. Дейт К.Дж. Введение в системы баз данных, 7-е издание: Пер. с англ./ К.Дж.Дейт. – М.: Издательский дом «Вильямс», 2002. – 1072с.

4. Ульман Д. Основы систем баз данных/Д.Ульман. – М.: Финансы и статистика, 1983. – 334 с.

5. Кузнецов С.Д. Основы баз данных; Бином. Лаборатория знаний, Интернет-университет информационных технологий – Москва, 2010. – 488 с.

# **ПРИЛОЖЕНИЕ А**

## **SQL скрипт создания базы данных**

CREATE TABLE Reviews (

review\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

review\_date DATE,

rating INT

);

CREATE TABLE Books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

author\_id INT,

quantity INT,

title VARCHAR(255),

genre VARCHAR(255),

price DECIMAL(10,2),

year INT

);

CREATE TABLE Orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

customer\_id INT,

quantity INT,

status VARCHAR(255),

order\_date DATE

);

CREATE TABLE Customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

last\_name VARCHAR(255),

first\_name VARCHAR(255),

address VARCHAR(255)

);

CREATE TABLE Authors (

author\_id INT AUTO\_INCREMENT PRIMARY KEY,

birth\_date DATE,

last\_name VARCHAR(255),

first\_name VARCHAR(255)

);

ALTER TABLE Reviews

ADD CONSTRAINT FK\_Review\_Book FOREIGN KEY (book\_id) REFERENCES Books(book\_id) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Books

ADD CONSTRAINT FK\_Book\_Author FOREIGN KEY (author\_id) REFERENCES Authors(author\_id) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Orders

ADD CONSTRAINT FK\_Order\_Book FOREIGN KEY (book\_id) REFERENCES Books(book\_id) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT FK\_Order\_Customer FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id) ON DELETE CASCADE ON UPDATE CASCADE;

# **ПРИЛОЖЕНИЕ Б**

## **Код клиентского приложения**

Листинг 1 – Код программы

using MySql.Data.MySqlClient;

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;

namespace WindowsForms\_SQL\_course\_work

{

public partial class AdminBooksForm : Form

{

public AdminBooksForm()

{

InitializeComponent();

LoadBookList();

}

private void LoadBookList()

{

DataTable table1 = new DataTable();

PublishingHouseDB database = new PublishingHouseDB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command;

string mainform\_query = "SELECT \* FROM Books;";

database.openConnection();

command = new MySqlCommand(mainform\_query, database.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table1);

reviewsList.DataSource = table1;

}

private void label9\_Click(object sender, EventArgs e)

{

}

private void reviewID\_TextChanged(object sender, EventArgs e)

{

}

private void label6\_Click(object sender, EventArgs e)

{

}

private void numericUpDown2\_ValueChanged(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

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

{

try

{

int bookid = int.Parse(bookID.Text);

string bookname = bookName.Text;

int bookauthor = int.Parse(bookAuthor.Text);

string bookgenre = bookGenre.Text;

int bookamount = int.Parse(bookAmount.Text);

decimal bookprice = decimal.Parse(bookPrice.Text);

int bookyear = int.Parse(bookYear.Text);

using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=PublishingHouse"))

{

connection.Open();

string checkQuery = "SELECT COUNT(\*) FROM Books WHERE book\_id = @bookId";

using (MySqlCommand command = new MySqlCommand(checkQuery, connection))

{

command.Parameters.AddWithValue("@bookId", bookid);

int existingCount = Convert.ToInt32(command.ExecuteScalar());

Sheleruch, [09.03.2024 10:51]

if (existingCount > 0)

{

string updateQuery = "UPDATE Books SET quantity = quantity + @bookAmount WHERE book\_id = @bookId";

using (MySqlCommand updateCommand = new MySqlCommand(updateQuery, connection))

{

updateCommand.Parameters.AddWithValue("@bookId", bookid);

updateCommand.Parameters.AddWithValue("@bookAmount", bookamount);

updateCommand.ExecuteNonQuery();

}

}

else

{

string insertQuery = "INSERT INTO Books (book\_id, author\_id, title, genre, quantity, price, year) VALUES (@bookId, @bookAuthor, @bookName, @bookGenre, @bookAmount, @bookPrice, @bookYear)";

using (MySqlCommand insertCommand = new MySqlCommand(insertQuery, connection))

{

insertCommand.Parameters.AddWithValue("@bookId", bookid);

insertCommand.Parameters.AddWithValue("@bookAuthor", bookauthor);

insertCommand.Parameters.AddWithValue("@bookName", bookname);

insertCommand.Parameters.AddWithValue("@bookGenre", bookgenre);

insertCommand.Parameters.AddWithValue("@bookAmount", bookamount);

insertCommand.Parameters.AddWithValue("@bookPrice", bookprice);

insertCommand.Parameters.AddWithValue("@bookYear", bookyear);

insertCommand.ExecuteNonQuery();

}

}

}

MessageBox.Show("Книга успешно добавлена.");

LoadBookList();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при добавлении книги: " + ex.Message);

LoadBookList();

}

}

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

{

try

{

int bookId = int.Parse(bookID.Text);

using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=PublishingHouse"))

{

connection.Open();

string deleteQuery = "DELETE FROM Books WHERE book\_id = @bookId";

using (MySqlCommand command = new MySqlCommand(deleteQuery, connection))

{

command.Parameters.AddWithValue("@bookId", bookId);

command.ExecuteNonQuery();

}

}

MessageBox.Show("Книга успешно удалена.");

LoadBookList();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при удалении книги: " + ex.Message);

}

}

}

}

Sheleruch, [09.03.2024 10:53]

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;

namespace WindowsForms\_SQL\_course\_work

{

public partial class AdminForm : Form

{

public AdminForm()

{

InitializeComponent();

}

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

{

AdminReviewsForm reviewsForm = new AdminReviewsForm();

reviewsForm.Show();

}

private void AdminForm\_FormClosing(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

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

{

AdminBooksForm booksForm = new AdminBooksForm();

booksForm.Show();

}

}

}

Sheleruch, [09.03.2024 10:53]

using MySql.Data.MySqlClient;

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;

namespace WindowsForms\_SQL\_course\_work

{

public partial class AdminReviewsForm : Form

{

public AdminReviewsForm()

{

InitializeComponent();

LoadReviewList();

}

private void reviewID\_TextChanged(object sender, EventArgs e)

{

}

private void LoadReviewList()

{

DataTable table1 = new DataTable();

PublishingHouseDB database = new PublishingHouseDB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command;

string mainform\_query = "SELECT \* FROM reviews;";

database.openConnection();

command = new MySqlCommand(mainform\_query, database.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table1);

reviewsList.DataSource = table1;

}

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

{

try

{

// Получаем ID отзыва из выделенной строки в DataGridView

int reviewId = int.Parse(reviewID.Text);

// Удаляем отзыв из таблицы Review

using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=PublishingHouse"))

{

connection.Open();

string query = "DELETE FROM Reviews WHERE review\_id = @reviewId";

using (MySqlCommand command = new MySqlCommand(query, connection))

{

command.Parameters.AddWithValue("@reviewId", reviewId);

command.ExecuteNonQuery();

}

}

// Обновляем DataGridView

LoadReviewList();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при удалении отзыва: " + ex.Message);

}

}

}

}

Sheleruch, [09.03.2024 10:54]

using MySql.Data.MySqlClient;

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;

namespace WindowsForms\_SQL\_course\_work

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

String loginUser = loginBox.Text;

String passwordUser = passBox.Text;

PublishingHouseDB database = new PublishingHouseDB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM Users WHERE username = @uL AND password = @uP", database.getConnection());

command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = loginUser;

command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = passwordUser;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MainForm mainForm = new MainForm(table.Rows[0]["username"].ToString());

this.Hide();

if (loginUser == "admin")

{

AdminForm adminForm = new AdminForm();

adminForm.Show();

MessageBox.Show("Авторизован");

}

else

{

mainForm.Show();

}

}

else

MessageBox.Show("Ошибка авторизации");

}

}

}

Sheleruch, [09.03.2024 10:54]

using MySql.Data.MySqlClient;

using MySqlX.XDevAPI.Relational;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsForms\_SQL\_course\_work

{

public partial class MainForm : Form

{

public string log { get; set; }

public int userID { get; set; }

public MainForm(string log)

{

InitializeComponent();

this.log = log;

LoadBookList();

}

private void LoadBookList()

{

DataTable table1 = new DataTable();

PublishingHouseDB database = new PublishingHouseDB();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command;

string mainform\_query = "SELECT \* FROM books;";

database.openConnection();

command = new MySqlCommand(mainform\_query, database.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table1);

bookList.DataSource = table1;

}

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

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

{

// Получаем ID книги из текстового поля

int book;

if (!int.TryParse(bookID.Text, out book))

{

MessageBox.Show("Введите корректный ID книги.");

return;

}

// Получаем количество книг из NumericUpDown

int amount = (int)booksAmount.Value;

// Создаем экземпляр сервиса заказов

OrderService orderService = new OrderService("server=localhost;port=3306;username=root;password=;database=PublishingHouse");

// Проверяем наличие книг на складе

if (!orderService.CheckAvailableQuantity(book, amount))

{

MessageBox.Show("Недостаточное количество книг на складе.");

return;

}

// Создаем заказ

try

{

orderService.CreateOrder(book, amount, log);

LoadBookList();

MessageBox.Show("Заказ успешно создан.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при создании заказа: " + ex.Message);

}

}

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

{

OrdersForm ordersForm = new OrdersForm(log);

ordersForm.Show();

}

private void label9\_Click(object sender, EventArgs e)

{

}

}

}

Sheleruch, [09.03.2024 10:54]

using System;

using MySql.Data.MySqlClient;

public class OrderService

{

private string connectionString;

public OrderService(string connectionString)

{

this.connectionString = connectionString;

}

public bool CheckAvailableQuantity(int bookId, int amount)

{

try

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

string query = "SELECT quantity FROM Books WHERE book\_id = @bookId";

using (MySqlCommand command = new MySqlCommand(query, connection))

{

command.Parameters.AddWithValue("@bookId", bookId);

connection.Open();

int availableQuantity = Convert.ToInt32(command.ExecuteScalar());

return availableQuantity >= amount;

}

}

}

catch (Exception ex)

{

Console.WriteLine("Ошибка при проверке наличия книг на складе: " + ex.Message);

return false;

}

}

public void CreateOrder(int bookId, int amount, string log)

{

int userID = -1;

try

{

using (MySqlConnection connection = new MySqlConnection(connectionString))

{

string query = "SELECT user\_id FROM USERS WHERE username = @username";

using (MySqlCommand command = new MySqlCommand(query, connection))

{

command.Parameters.AddWithValue("@username", log);

connection.Open();

object result = command.ExecuteScalar();

if (result != null)

{

userID = Convert.ToInt32(result);

}

}

string orderQuery = "INSERT INTO Orders (book\_id, customer\_id, quantity, status, order\_date) VALUES (@bookId, @customer\_id, @quantity, 'Pending', CURRENT\_TIMESTAMP)";

using (MySqlCommand command = new MySqlCommand(orderQuery, connection))

{

command.Parameters.AddWithValue("@bookId", bookId);

command.Parameters.AddWithValue("@quantity", amount);

command.Parameters.AddWithValue("@customer\_id", userID);

command.ExecuteNonQuery();

}

string updateBooksQuery = "UPDATE Books SET quantity = quantity - @amount WHERE book\_id = @bookId";

using (MySqlCommand command = new MySqlCommand(updateBooksQuery, connection))

{

command.Parameters.AddWithValue("@bookId", bookId);

command.Parameters.AddWithValue("@amount", amount);

command.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

Console.WriteLine("Ошибка при создании заказа: " + ex.Message);

}

}

}

Sheleruch, [09.03.2024 10:54]

using MySql.Data.MySqlClient;

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;

namespace WindowsForms\_SQL\_course\_work

{

public partial class OrdersForm : Form

{

public string log { set; get; }

public OrdersForm(string log)

{

InitializeComponent();

this.log = log;

LoadOrdersList(log);

}

private void LoadOrdersList(string username)

{

try

{

int userId = GetUserIdByUsername(username);

if (userId == -1)

{

MessageBox.Show("Пользователь с таким именем не найден.");

return;

}

string query = "SELECT \* FROM orders WHERE customer\_id = @userId";

using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=PublishingHouse"))

{

connection.Open();

using (MySqlCommand command = new MySqlCommand(query, connection))

{

command.Parameters.AddWithValue("@userId", userId);

using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))

{

DataTable table = new DataTable();

adapter.Fill(table);

ordersList.DataSource = table;

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке заказов: " + ex.Message);

}

}

public int GetUserIdByUsername(string username)

{

int userId = -1; // Значение по умолчанию, если пользователь не найден

try

{

using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=PublishingHouse"))

{

string query = "SELECT user\_id FROM users WHERE username = @username";

using (MySqlCommand command = new MySqlCommand(query, connection))

{

command.Parameters.AddWithValue("@username", username);

connection.Open();

object result = command.ExecuteScalar();

if (result != null)

{

userId = Convert.ToInt32(result);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при получении user\_id: " + ex.Message);

}

return userId;

}

private void ordersList\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

}

Sheleruch, [09.03.2024 10:54]

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsForms\_SQL\_course\_work

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LoginForm());

}

}

}

Sheleruch, [09.03.2024 10:54]

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WindowsForms\_SQL\_course\_work

{

class PublishingHouseDB

{

MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=PublishingHouse");

public void openConnection()

{

if(connection.State == System.Data.ConnectionState.Closed)

{

connection.Open();

}

}

public void closeConnection()

{

if (connection.State == System.Data.ConnectionState.Open)

{

connection.Close();

}

}

public MySqlConnection getConnection()

{

return connection;

}

}

}