



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ (ИУ)

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)

# **РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

## ***К КУРСОВОЙ РАБОТЕ***

### ***НА ТЕМУ:***

#### ***Разработка базы данных книжной поисковой системы***

Студент группы ИУ7-63Б

Руководитель курсовой работы

Светличная А. А.

Филиппов М. В.

2023 г.

## РЕФЕРАТ

Рассчетно-пояснительная записка 42 с., 21 рис., 3 табл., 13 источн., 1 прил.

КНИГИ, ПОИСК КНИГ, БАЗЫ ДАННЫХ, WEB-ПРИЛОЖЕНИЕ, MVC, C#

Объект разработки: база данных и Web-приложение для работы с ней.  
Объект исследования: кластеризованные и некластеризованные индексы.

Цель работы: разработка базы данных для книжной информационной системы.

В конструкторской части были выделены 5 сущностей базы данных (книга, автор, серия, пользователь, книжная полка) и 3 роли (гость, авторизованный пользователь, администратор).

В технологической части были выбрана архитектура проекта на основе MVC и следующие средства реализации: C# как язык программирования, Microsoft SQL Server как средство управления базами данных.

В исследовательской части был проведен эксперимент, в ходе которого была рассмотрена зависимость времени выполнения сортировки от количества строк в таблице и наличия индексов. Результаты данного эксперимента показали, что использование кластеризованного индекса уменьшает время выполнения запроса в среднем в 1,5 раза, а некластеризованного — в 1,3 раза.

Область применения результатов: дальнейшее расширение книжной поисковой системы до приложения с возможность прочтения или покупки книг в приложении.

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	4
<b>1 Аналитическая часть</b>	5
1.1 Анализ существующих решений	5
1.2 Модели баз данных	6
1.2.1 Дореляционные базы данных	6
1.2.2 Реляционные базы данных	9
1.2.3 Постреляционные базы данных	11
1.3 Формализация задачи	11
1.4 Формализация данных	12
1.5 Формализация ролей пользователя	13
<b>2 Конструкторская часть</b>	16
2.1 Проектирование базы данных	16
2.2 Ограничения целостности базы данных	19
2.3 Роли базы данных	20
2.4 Триггеры базы данных	21
<b>3 Технологическая часть</b>	24
3.1 Средства реализации	24
3.2 Реализация ролей	25
3.3 Реализация триггеров	27
3.4 Демонстрация работы приложения	28
<b>4 Исследовательская часть</b>	36
4.1 Описание эксперимента	36
4.2 Результаты эксперимента	37
<b>ЗАКЛЮЧЕНИЕ</b>	39
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	40
<b>ПРИЛОЖЕНИЕ А Презентация</b>	42

# ВВЕДЕНИЕ

В 2023 году существует большое количество различных видов досуга. Какая-то часть людей выбирает для себя просмотр видео и фильмов, другая же предпочитает компьютерные игры. Однако, несмотря на такое разнообразие современных видов времяпрепровождения, люди продолжают выбирать один из наиболее древних досугов — чтение книг.

Согласно данным Российской книжной палаты, в 2021 г. российскими издательствами было выпущено 108 460 названий книг и брошюр совокупным тиражом 389,5 млн. экземпляров [1]. Это позволяет сделать вывод, что люди действительно увлечены чтением и книгами в целом.

Однако в экономике существует такое понятие как «избыток выбора» — это феномен, который возникает в результате слишком большого количества равнозначных вариантов, доступных потребителю. При этом чем больше выбор, тем беспокойнее чувствует себя потребитель. Возможным решением данной проблемы будет уточнение деталей о том или ином продукте, что сделает варианты не равнозначными [2].

Принимая во внимание все те аспекты вопроса, которые были описаны выше, было принято решение создать информационную систему книг, позволяющую производить поиск произведений по тем или иным характеристикам, например, таким как жанр, год выпуска, рейтинг и т.д.

**Цель курсовой работы:** разработка базы данных для книжной информационной системы.

**Задачи курсовой работы:**

- анализ существующих решений;
- анализ существующих моделей баз данных и выбор подходящей;
- формализация задачи и определение необходимых ролей;
- проектирование и разработка базы данных;
- проектирование и разработка WEB-приложение;
- исследование характеристик разработанного программного обеспечения.

# 1 Аналитическая часть

## 1.1 Анализ существующих решений

Вопрос выбора подходящей под настроение или желание потребителя книги далеко не нов, по этой причине на рынке уже существуют какое-то количество решений, предоставляющих достаточно различный функционал.

- Zelluloza
- Knigopoisk
- Fantlab

Сформулируем критерии сравнения:

1. Возможность добавления книг в избранное.
2. Возможность поиска книг по комбинации параметров.
3. Предоставление информации о книге по единому шаблону.
4. Предоставление информации об авторе.
5. Предоставление информации о книжных сериях.

Таблица 1.1 – Сравнение существующих решений

Критерий	Zelluloza	Knigopoisk	Fantlab
1	-	+	-
2	+	+	-
3	-	-	+
4	-	+	-
5	-	-	-

Таким образом, ни одно из рассмотренных решений не удовлетворяет всем критериям сравнения. Что позволяет предположить, что потребность в решениях, связанных с этой темой, все еще сохраняется.

## 1.2 Модели баз данных

**База данных** — это самодокументированное собрание интегрированных записей, которая состоит из двух частей: данные и метаданные, с помощью которых данные интегрируются и управляются [3].

**Система управления базами данных** — это приложение, обеспечивающий хранение, обновление и поиск информации в базе данных [3].

Основные функции СУБД:

- управление данными во внешней памяти (на дисках);
- управление данными в оперативной памяти с использованием дискового кэша;
- журнализация изменений (сохранение истории), резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД (язык определения данных, язык манипулирования данными) [3].

Существуют следующие группы моделей баз данных, отличающиеся структурой организации данных:

- дореляционные;
- реляционные;
- постреляционные [3].

### 1.2.1 Дореляционные базы данных

К дореляционным моделям баз данных относятся:

- иерархическая;
- сетевая;
- основанная на инвертированных списках [4].

# Иерархическая модель баз данных

Иерархическая модель представляет данные в виде упорядоченного графа (дерева) и определяется в терминах: элемент, агрегат, запись, групповое отношение [4].

1. Элемент (атрибут, поле) — наименьшая единица структуры данных. Обычно каждому элементу при описании базы данных присваивается уникальное имя. По этому имени к нему обращаются при обработке.
2. Запись (группа) — именованная совокупность атрибутов. Использование записей позволяет за одно обращение к базе получить некоторую логически связанную совокупность данных. Именно записи изменяются, добавляются и удаляются. Тип записи определяется составом ее атрибутов. Экземпляр записи — конкретная запись с конкретным значением элементов
3. Групповое отношение — иерархическое отношение между записями двух типов. Родительская запись (владелец группового отношения) называется исходной записью, а дочерние записи (члены группового отношения) — подчиненными. Иерархическая база данных может хранить только такие древовидные структуры [4].

## **Преимущества** иерархической модели баз данных:

- простота структуры;
- эффективность доступа;
- поддержка иерархических связей;
- эффективность при больших объемах данных;
- выполнение широкого спектра узкопрофильных задач [5].

## **Недостатки** иерархической модели баз данных:

- дублирование данных;
- трудности с изменениями структуры;

- неэффективность для данных с отношением M:N;
- ограниченность однонаправленных отношений;
- проблемы с целостностью данных при изменении или удалении родительского элемента [6].

## **Сетевая модель баз данных**

Сетевая модель баз данных развивалась из иерархической и представляет информацию в виде сети, состоящей из записей, связей и узлов [4].

Она организует данные в структуру, где каждая запись может иметь несколько связей с другими записями, что создает сложные отношения между данными. При этом одна и та же запись может участвовать в различных связях, что обеспечивает более гибкую структуру базы данных [7].

Для доступа к данным в сетевой модели используется навигационный подход. Это означает, что для получения нужной информации нужно обходить сеть, перемещаясь от одной связанной записи к другой [7].

**Преимущества** сетевой модели баз данных:

- представление сложных отношения между данными;
- быстрый доступ;
- повторное использование данных [8].

**Недостатки** сетевой модели баз данных:

- сложность разработки;
- ограниченная поддержки инструментов и СУБД;
- сложность запросов [8].

## **Модель баз данных, основанная на инвертированных списках**

Организация доступа к данным на основе инвертированных списков используется практически во всех современных реляционных СУБД, но в реляционных СУБД пользователи не имеют непосредственного доступа к инвертированным спискам (индексам). База данных на инвертированных списках



похожа на реляционную базу данных, т. е. также состоит из таблиц отношений, однако ей присущи важные отличия:

- допускается сложная структура атрибутов (атрибуты не обязательно атомарны);
- строки таблиц (записи) упорядочены в некоторой последовательности, каждой строке присваивается уникальный номер (физическая упорядоченность строк всех таблиц может определяться и для всей базы данных);
- пользователям видны и хранимые таблицы, и пути доступа к ним;
- пользователь может управлять логическим порядком строк в каждой таблице с помощью специального инструмента — индексов (эти индексы автоматически поддерживаются системой и явно видны пользователям) [9].

Некоторые атрибуты могут быть объявлены поисковыми (ключевыми), для каждого из них создается индекс, который содержит упорядоченные значения ключей и указатели на соответствующие записи основной таблицы (инвертированный список). Если таблицу требуется упорядочить по нескольким ключам, то создается несколько индексов.

**Преимущества** модели, основанной на инвертированных списках: более быстрый поиск (особенно поиск уникальной записи по нескольким условиям), возможность хранения элементов данных со сложной структурой.

**Недостатки** модели, основанной на инвертированных списках: отсутствие строгого математического аппарата, отсутствие средств для описания ограничений целостности базы данных [9].

## 1.2.2 Реляционные базы данных

Реляционная модель баз данных — это структурированный способ организации данных в базе данных, в которой данные представлены в виде таблиц (реляций), состоящих из строк и столбцов. Реляционная модель использует набор формальных правил для описания, как данные могут быть сохранены, изменены и извлечены из базы данных [10].

В реляционной модели, каждая таблица представляет отдельный тип данных и имеет своё название. Каждый столбец в таблице представляет отдельное свойство данных, а каждая строка представляет отдельную запись в базе данных. Каждая запись имеет уникальный идентификатор, который называется ключом [10].

Реляционные базы данных обеспечивают набор свойств ACID:

- Атомарность (Atomicity) — транзакция считается атомарной, если она либо полностью выполняется, либо не выполняется вовсе. Никакая часть транзакции не может быть выполнена отдельно от других частей, и если происходит ошибка, то все изменения должны быть отменены (откат).
- Согласованность (Consistency) — транзакция должна гарантировать, что база данных находится в согласованном состоянии до и после ее выполнения. Это означает, что все ограничения на данные, такие как ограничения уникальности, должны быть соблюдены.
- Изолированность (Isolation) — транзакции должны выполняться в изолированном режиме, так что результаты выполнения одной транзакции не должны влиять на результаты выполнения других транзакций. Это означает, что транзакции должны работать с базой данных так, как будто они работают в изолированной среде.
- Устойчивость (Durability) — после завершения транзакции ее изменения должны быть сохранены в базе данных и должны оставаться неизменными даже в случае сбоя системы [10].

**Преимущества** реляционной модели баз данных:

- интуитивно понятная структура;
- простота изменения и настройки в соответствии с требованиями конкретного приложения;
- масштабируемость;
- контроль доступа к данным;
- использование стандарта SQL;

- возможность определения правил наложения ограничений на значения в столбцах;
- оптимизация запросов [11].

**Недостатки** реляционной модели баз данных:

- сложность проектирования;
- семантическая перегрузка;
- однородная структура данных;
- трудности организации рекурсивных запросов [12].

### 1.2.3 Постреляционные базы данных

Для постреляционных моделей данных снимается ограничение неделимости данных: появляется возможность многозначных полей. Набор подзначений становится самостоятельной таблицей, встроенной в главную. Информация может быть представлена с помощью следующих структур организации данных:

- хэш-таблица пар «ключ-значение»;
- документы, упорядоченные по группам, называемым коллекциями;
- граф — модель на основе узлов и ребер, представляющих взаимосвязанные данные.

**Преимущества** постреляционных баз данных: простота масштабирования, отсутствие ограничений на типы данных.

**Недостатки** постреляционных баз данных: несовместимость с запросами SQL, ограничение требований свойств ACID.

## 1.3 Формализация задачи

Для курсовой работы требуется разработать базу данных, которая будет использоваться в книжной поисковой системе. Она должна содержать информацию об авторах, книгах и книжных сериях. Также необходимо создать

Web-приложение, которое будет позволять пользователям работать с базой данных, осуществляя поиск по различным параметрам, а также создавать свои собственные «книжные полки» и взаимодействовать с ними. Приложение должно реализовывать регистрацию и авторизацию, которая переключает пользователей на различные роли, каждая из которых получает свой определённый набор действий. Необходимо реализовать функционал для редактирования содержимого базы данных: добавление и обновление книг, редактирование прав пользователей.

## 1.4 Формализация данных

Для книжной поисковой системы требуется база данных, которая будет содержать информацию о книгах, авторах, сериях, пользователях и «книжных полках» пользователя.

Таблица 1.2 – Сущности и их атрибуты разрабатываемой базы данных

Сущность	Атрибуты
Автор	Id, Имя, Год рождения, Год смерти, Страна, Жанр
Книга	Id, Название, Жанр, Год выпуска, Язык оригинала, Рейтинг
Серия	Id, Название, Жанр, Издательство, Год первой публикации, Средний рейтинг
Пользователь	Id, Логин, Пароль, Права
Книжная полка	Id, Название, Количество книг, Средний рейтинг

Также на рисунке 1.1 изображена ER-диаграмма системы в нотации Чена.

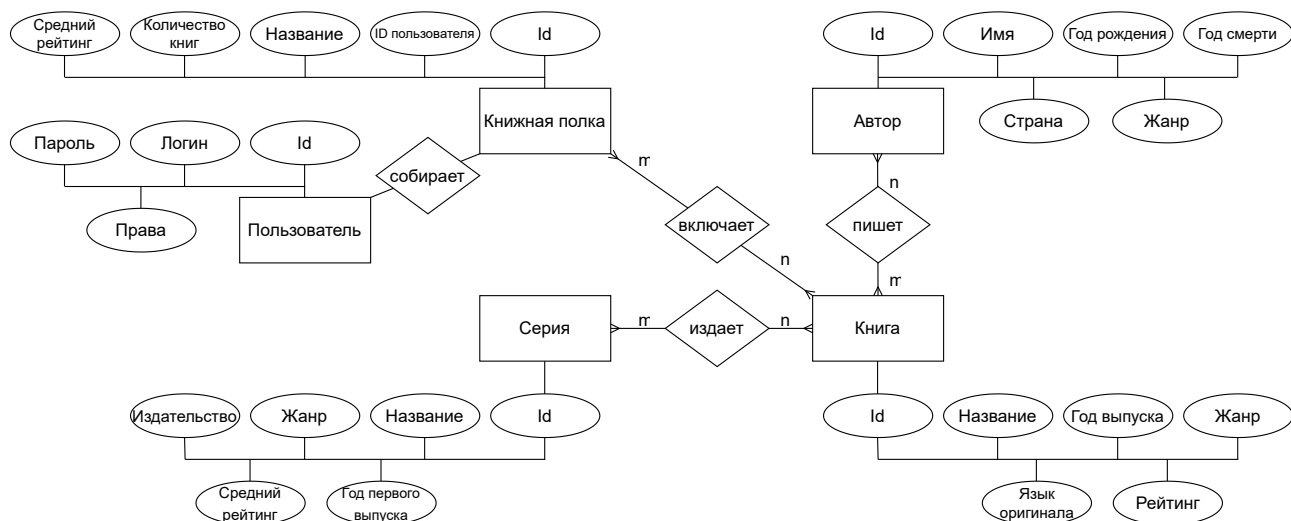


Рисунок 1.1 – ER-диаграмма в нотации Чена

## 1.5 Формализация ролей пользователя

Web-приложение предоставляет три уровня доступа для пользователей: гость, авторизованный пользователь и администратор. Каждая из этих ролей предоставляет различные права и функциональность в приложении.

**Гость** может просматривать базовую информацию о произведениях, авторах и книжных сериях, осуществляя простой поиск по названию или расширенный поиск по параметрам (для гостя это параметры, доступные для просмотра), расширенный поиск книг имеет дополнительные параметры: имя автора и книжной серии, которые учитываются, только если введены. Для гостя также доступна регистрация и авторизация, которые переводят пользователя в роль авторизованного.

**Авторизованный пользователь** получает два основных вектора возможностей: поиск и работа с личным кабинетом.

Авторизованный пользователь также может осуществлять простой и расширенный поиск, однако он получает большее количество информации и полей для поиска соответственно. На странице расширенного поиска книг авторизованный пользователь может добавить книгу к себе на «книжную полку».

В личном кабинете данный пользователь имеет возможность просмотреть свою «книжную полку», изменить пароль, удалить аккунт. На странице «книжной полки» доступна общая информация о количестве книг в избранном и среднем рейтинге этих книг, информация о названии, авторе, рейтинге

и состоянии (прочитано или нет) всех добавленных книг. Кроме того есть возможность отметить как прочитанное отдельную книгу или удалить её.

Авторизованный пользователь все еще имеет возможность зарегистрироваться или авторизоваться, но в таком случае он будет переключен на пользователя с соответствующим логином.

**Администратор** имеет право на все действия доступные для авторизованного пользователя и дополнительно возможность изменения информации в базе данных: добавление и редактирование книг, а также изменение прав зарегистрированных пользователей (переключение возможно только на авторизованного пользователя или администратора).

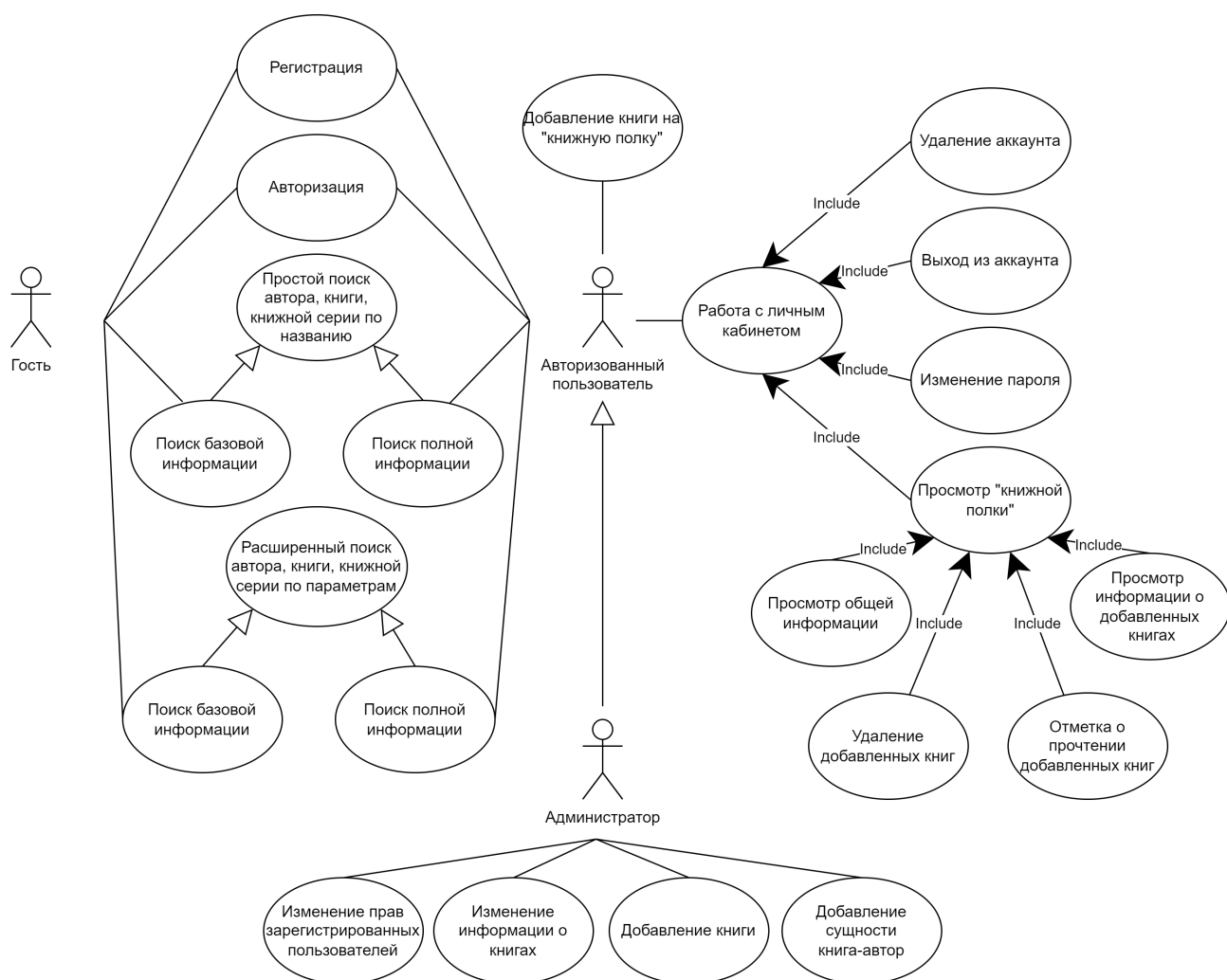


Рисунок 1.2 – Use-case диаграмма

## Вывод

В данном разделе были проанализованы аналоги поисковой системы книг, ни одно из рассмотренных решений не удовлетворило всем выдвину-

тым критериям сравнения. Также были формализованы поставленная задача, данные, а также роли пользователя. Были рассмотрены существующие модели баз данных. Для решения задачи была выбрана реляционная модель, поскольку в ходе анализа было выявлено наибольшее количество преимуществ, подходящих для данного проекта, именно у этой модели.

## 2 Конструкторская часть

### 2.1 Проектирование базы данных

На рисунке 2.1 изображена диаграмма разрабатываемой базы данных в соответствии с ER-диаграммой системы в нотации Чена на рисунке 1.1.

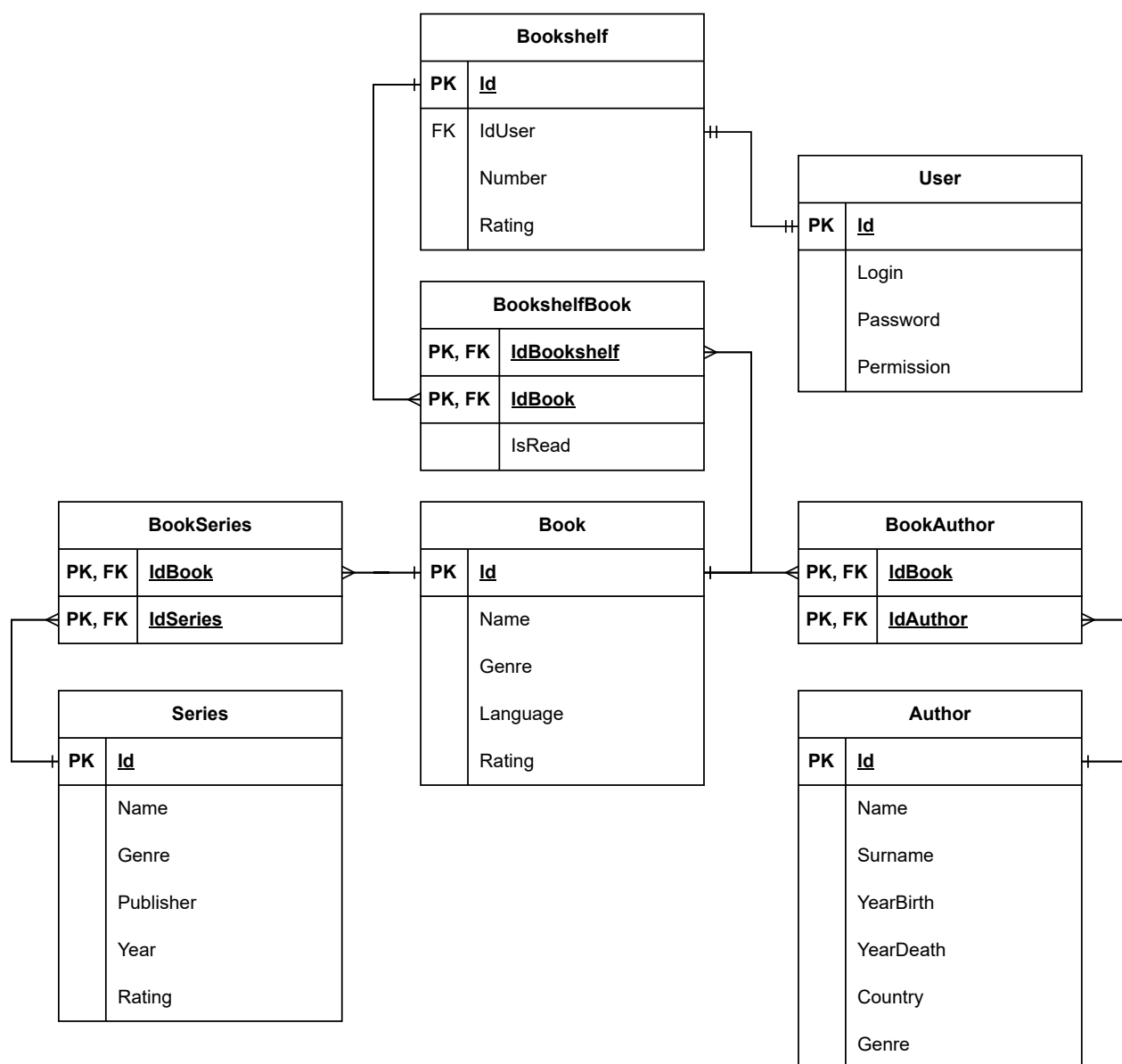


Рисунок 2.1 – Диаграмма разрабатываемой базы данных



Таблица **Author** содержит информацию об авторах и имеет следующие поля:

- Id — уникальный идентификатор автора в базе данных.
- Name — строковое значение, содержащее имя автора.
- YearBirth — целое числовое значение, указывающее год рождения автора.
- YearDeath — целое числовое значение, указывающее год смерти автора.
- Country — строковое значение, указывающее страну проживания автора.
- Genre — строковое значение, указывающее основной жанр произведений, написанных автором.

Таблица **Book** содержит информацию о книгах и имеет следующие поля:

- Id — уникальный идентификатор книги в базе данных.
- Name — строковое значение, описывающее название книги.
- Year — целое числовое значение, указывающее год выпуска книги.
- Genre — строковое значение, описывающее жанр книги.
- Language — строковое значение, указывающее язык, на котором была написана оригинальная версия книги.
- Rating — вещественное числовое значение, представляющее оценку книги по пятибалльной шкале.

Таблица **Series** содержит информацию о книжных сериях и имеет следующие поля:

- Id — уникальный идентификатор книжной серии в базе данных.
- Name — строковое значение, описывающее название книжной серии.

- Genre — строковое значение, указывающее основной жанр книжной серии.
- Publisher — строковое значение, указывающее издательство, выпускающее книги в рамках данной серии.
- Year — целое числовое значение, указывающее год публикации первой книги в серии.
- Rating — вещественное числовое значение, описывающее средний рейтинг книг в серии по пятибалльной шкале.

Таблица **User** содержит информацию о пользователях и имеет следующие поля:

- Id — уникальный идентификатор пользователя в базе данных.
- Login — строковое значение, указывающее логин пользователя для входа в систему.
- Password — строковое значение, содержащее зашифрованный пароль пользователя для входа в систему.
- Permission — строковое значение, указывающее права доступа пользователя.

Таблица **Bookshelf** содержит информацию о «книжных полках» пользователя и имеет следующие поля:

- Id — уникальный идентификатор «книжной полки» в базе данных.
- IdUser — уникальный идентификатор пользователя в базе данных, которому принадлежит «книжная полка».
- Number — целое числовое значение, указывающее количество книг на «книжной полке».
- Rating — вещественное числовое значение, указывающее средний рейтинг книг на «книжной полке» по пятибалльной шкале.

Таблицы, содержащие названия двух сущностей, являются таблицами-связками в случае, если какие-то сущности определяются связью «многие ко многим». Поля — идентификаторы сущностей, которые подлежат связыванию, а именно книга — автор (BookAuthor), книга — серия (BookSeries), книжная полка — книга (BookshelfBook).

Однако таблица BookshelfBook имеет дополнительное поле «IsRead», которое определяет является ли книга прочитанной пользователем, которому принадлежит «книжная полка», или нет.

## 2.2 Ограничения целостности базы данных

Для обеспечения целостности базы данных необходимо установить некоторые ограничения.

Таблица **Author**:

- Id — уникальное положительное число, первичный ключ.
- Name — не может отсутствовать.
- YearBirth — не может отсутствовать.
- Country — не может отсутствовать.
- Genre — не может отсутствовать.

Таблица **Book**:

- Id — уникальное положительное число, первичный ключ.
- Name — не может отсутствовать.
- Year — не может отсутствовать.
- Genre — не может отсутствовать.
- Language — не может отсутствовать.
- Rating — не может отсутствовать.

Таблица **Series**:

- Id — уникальное положительное число, первичный ключ.
- Name — не может отсутствовать .
- Genre — не может отсутствовать.
- Publisher — не может отсутствовать.
- Year — не может отсутствовать.
- Rating — не может отсутствовать.

#### Таблица **User**:

- Id — уникальное положительное число, первичный ключ.
- Login — уникальное значение, не может отсутствовать.
- Password — не может отсутствовать.
- Permission — не может отсутствовать.

#### Таблица **Bookshelf**:

- Id — уникальное положительное число, первичный ключ.
- IdUser — внешний ключ, не может отсутствовать.

Для всех таблиц-связок существуют ограничения первичного и внешнего ключа для каждого атрибута.

## 2.3 Роли базы данных

В аналитической части были определены три роли пользователей для взаимодействия с приложением: гость, авторизованный пользователь и администратор. Каждой из этих ролей необходимо создать соответствующую роль в базе данных с соответствующим названием.

**Гость** во время поиска получает не всю информацию, а лишь часть, поэтому необходимо создать ограничение на доступ только к некоторым полям:

- Author — Name, Genre, Country;
- Book — Name, Genre, Rating;
- Series — Name, Genre, Rating.

А также для реализации регистрации и авторизации необходимо предоставить возможность работы с такими сущностями как User и Bookshelf, однако для названных операций будет достаточно операций Select и Insert.

**Авторизованный пользователь**, в отличие от гостя, получает полный набор информации, а также имеет возможность запрашивать получение книг по имени автора или названию книжной серии, поэтому необходимо предусмотреть доступ не только к основным таблицам, но и к таблицам-связкам.

Кроме регистрации и авторизации этому виду пользователя доступно редактирование пароля и удаление аккаунта, а следовательно дополнительно необходимы операции Update и Insert соответственно.

у авторизованного пользователя также есть доступ к различному функционалу связанному с «книжной полкой», поэтому данная роль требует разрешения всех видов операций по отношению к таблицам Bookshelf и BookshelfBook.

**Администратор** является ролью, которая обладает наибольшими правами и возможностью редактирования информационных таблиц базы данных, в следствие этого она должна мочь совершать все виды операций по отношению ко всем таблицам, включая таблицы-связки.

## 2.4 Триггеры базы данных

В спроектированной базе данных содержится сущность (Bookshelf), атрибуты которой (Number, Rating) будут изменяться только в следствие обновления другой сущности (BookshelfBook), поэтому очевидным решением является добавление триггеров вставки и удаления из таблицы BookshelfBook, которые будут пересчитывать эти атрибуты автоматически.

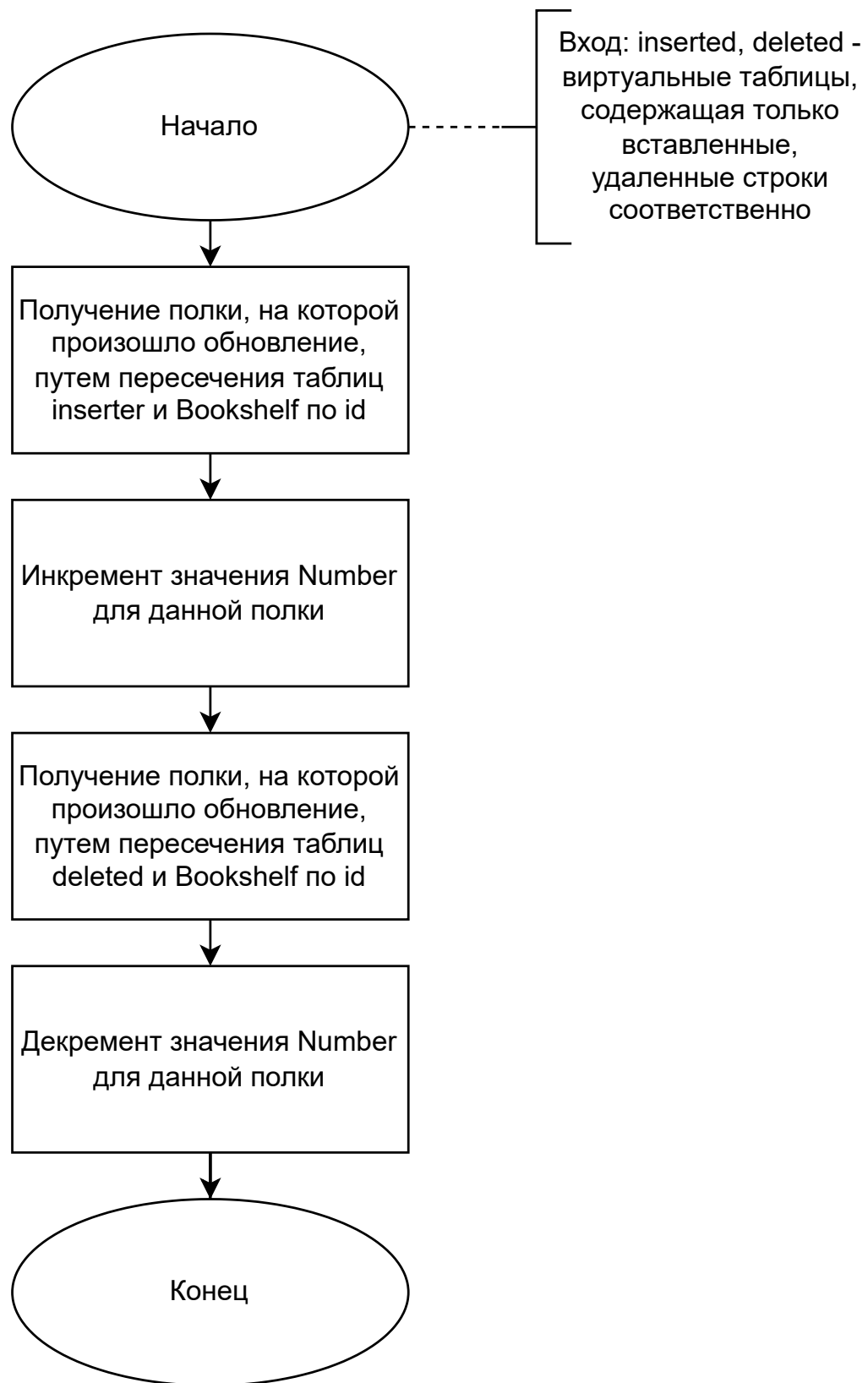


Рисунок 2.2 – Схема триггера обновления количества книг книжной полки после добавления или удаления книги

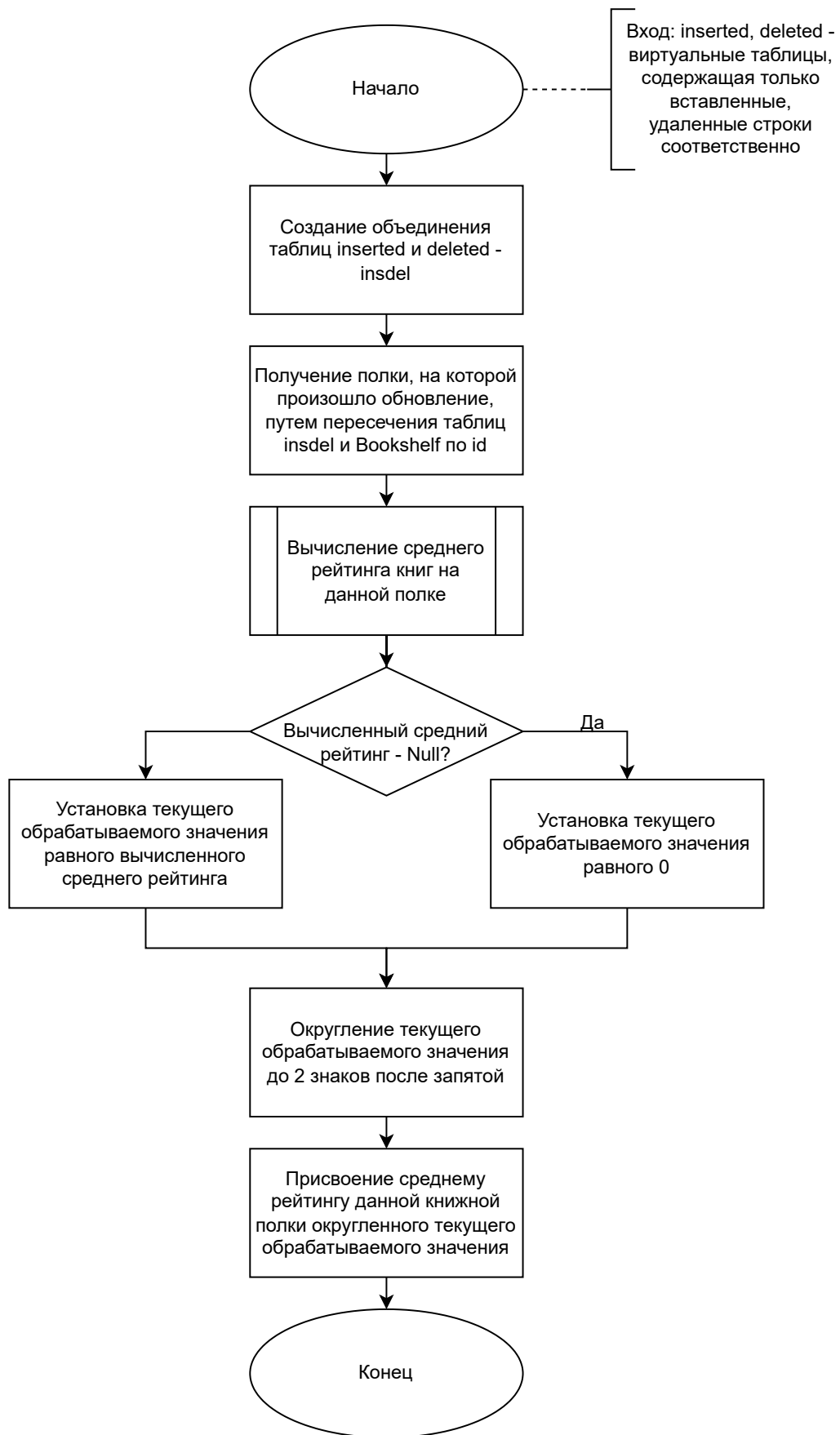


Рисунок 2.3 – Схема триггера обновления среднего рейтинга книжной полки после добавления или удаления книги

## 3 Технологическая часть

### 3.1 Средства реализации

Для написания данного проекта был выбран язык программирования C#, так как он обладает следующими преимуществами:

- поддержка принципов объектно-ориентированного программирования;
- предоставление мощного механизма для работы с многопоточностью и асинхронным программированием;
- тесная интеграция с различными продуктами и технологиями Microsoft, такими как Windows, Azure, SQL Server и другими;
- предоставление механизмов безопасности, таких как обработка исключений и управление памятью;
- применение для разработки разнообразных типов приложений, включая веб-приложения, настольные приложения, игры, мобильные приложения и другие.

В качестве архитектурной модели разрабатываемого приложения была выбрана MVC (MODEL, VIEW, CONTROLLER) в силу следующих достоинств:

- обеспечение явного разделения функциональности приложения на три компонента;
- легкая масштабируемость приложения, благодаря разделению ответственности;
- возможность переиспользования кода благодаря тому, что каждый компонент выполняет свою конкретную задачу;
- облегчение тестирования приложения, так как каждый компонент можно тестировать независимо от других;
- чистый и структурированный код, так как каждый компонент имеет четко определенную функцию и роль в приложении;



- уменьшение риска возникновения уязвимостей безопасности, так как каждый компонент выполняет определенные функции, и доступ к данным контролируется контроллером;

Из наиболее популярных реляционных СУБД, таких как MySQL, Oracle, PostgreSQL, Microsoft SQL Server и SQLite, была выбрана Microsoft SQL Server, так как она имеет следующие преимущества:

- широкий набор функциональных возможностей, таких как хранимые процедуры, функции, триггеры, представления и другие;
- тесная интеграция с другими продуктами и технологиями Microsoft, такими как .NET Framework, Visual Studio, Azure и другими;
- удобные инструменты для разработки баз данных и их управления, такие как SQL Server Management Studio и SQL Server Data Tools;
- механизмы безопасности для защиты данных от несанкционированного доступа, включая управление правами доступа;
- возможность работы без контейнеризации;

## 3.2 Реализация ролей

Роли, описанные в конструкторской части были реализованы следующим образом:

Листинг 3.1 – Роль «Гость»

```
1 CREATE LOGIN [guest] WITH PASSWORD = 'guest';
2 CREATE USER [guest] FOR LOGIN [guest];
3
4 GRANT SELECT ON Author (Name, Genre, Country) TO [guest];
5 GRANT SELECT ON Book (Name, Genre, Rating) TO [guest];
6 GRANT SELECT ON Series (Name, Genre, Rating) TO [guest];
7
8 GRANT SELECT, INSERT ON [User] TO [guest];
9
10 GRANT SELECT, INSERT ON Bookshelf TO [guest];
```

Листинг 3.2 – Роль «Авторизованный пользователь»

```
1 CREATE LOGIN [user] WITH PASSWORD = 'user';
2 CREATE USER [user] FOR LOGIN [user];
3
4 GRANT SELECT ON Author TO [user];
5 GRANT SELECT ON Book TO [user];
6 GRANT SELECT ON Series TO [user];
7
8 GRANT SELECT ON BookAuthor TO [user];
9 GRANT SELECT ON BookSeries TO [user];
10
11 GRANT SELECT, INSERT, UPDATE, DELETE ON [User] TO [user];
12
13 GRANT SELECT, INSERT, UPDATE, DELETE ON Bookshelf TO [user];
14
15 GRANT SELECT, INSERT, UPDATE, DELETE ON BookshelfBook TO [user
    ↪ ];
```

Листинг 3.3 – Роль «Администратор»

```
1 CREATE LOGIN [admin] WITH PASSWORD = 'admin';
2 CREATE USER [admin] FOR LOGIN [admin];
3 EXEC sp_addrolemember 'db_owner', 'admin';
```

### 3.3 Реализция триггеров

Триггеры, описанные в конструкторской части были реализованы следующим образом:

Листинг 3.4 – Триггер обновления количества книг на «книжной полке» после добавления или удаления книги

```
1 CREATE TRIGGER UpdateNumberBookshelf
2 ON Bookshelfbook
3 AFTER INSERT, DELETE
4 AS
5 BEGIN
6     UPDATE bs
7     SET Number = Number + 1
8     FROM Bookshelf bs
9     INNER JOIN inserted ins ON bs.Id = ins.IdBookshelf;
10
11    UPDATE bs
12    SET Number = Number - 1
13    FROM Bookshelf bs
14    INNER JOIN deleted del ON bs.Id = del.IdBookshelf;
15 END;
```

Листинг 3.5 – Триггер обновления среднего рейтинга «книжной полки» после добавления или удаления книги

```
1 CREATE TRIGGER UpdateRatingBookshelf
2 ON BookshelfBook
3 AFTER INSERT, DELETE
4 AS
5 BEGIN
6     UPDATE Bookshelf
7     SET Rating = ROUND((
8         SELECT COALESCE(AVG(b.Rating), 0)
9         FROM BookshelfBook bb
10        INNER JOIN Book b ON bb.IdBook = b.Id
11        WHERE bb.IdBookshelf = Bookshelf.Id
12    ), 2)
13 FROM Bookshelf
14 INNER JOIN (
15     SELECT IdBookshelf FROM inserted
16     UNION
17     SELECT IdBookshelf FROM deleted
18 ) insdel ON Bookshelf.Id = insdel.IdBookshelf;
19 END;
```

### 3.4 Демонстрация работы приложения

Как уже было сказано в аналитическом разделе, всю работу приложения можно разделить на две части: поиск и личный кабинет.

Для начала будет продемонстрирована работа с поисковыми строками приложения. Данное программное обеспечение предоставляет возможность осуществлять простой поиск только по названию, который происходит через одну поисковую строку для всех информационных сущностей базы данных, данная поисковая строка показана на рисунке 3.1.

**Поиск**

Найти

Рисунок 3.1 – Поисковая строка простого поиска

Однако результат поиска отличается в зависимости от роли, результат поиска книги для неавторизованного пользователя показан на рисунке 3.2, для авторизованного — на рисунке 3.3.

Результаты поиска		
Книги:		
Название	Основной жанр	Рейтинг
1984	Научная фантастика	4,38

Рисунок 3.2 – Результат простого поиска книги для роли «Гость»

Результаты поиска				
Книги:				
Название	Основной жанр	Язык оригинала	Год выпуска	Рейтинг
1984	Научная фантастика	Английский	1949	4,38

Добавить на книжную полку

Рисунок 3.3 – Результат простого поиска книги для роли «Авторизованный пользователь»

Также в приложении доступен расширенный поиск по параметрам, которому соответствует собственная поисковая форма для каждой информационной сущности базы данных. На рисунке 3.4 представлена поисковая форма для сущности «Книга», на рисунке 3.5 — результат такого поиска.

**Расширенный поиск книг**

Название

Основной жанр

Язык оригинала

Год выпуска

Параметры года выпуска: ☐ Больше ☐ Меньше ☐ Равно

Рейтинг

Параметры рейтинга: ☐ Больше ☐ Меньше ☐ Равно

Имя автора

Название книжной серии

**Найти**

Рисунок 3.4 – Поисковая форма книг в расширенном поиске

**Результаты поиска**

Книги:

Название	Основной жанр	Язык оригинала	Год выпуска	Рейтинг	Имя автора	
Евгений Онегин	Роман в стихах	Русский	1825	4,62	Александр Сергеевич Пушкин	<b>Добавить на книжную полку</b>
Дубровский	Повесть	Русский	1833	4,42	Александр Сергеевич Пушкин	<b>Добавить на книжную полку</b>
Капитанская дочка	Роман	Русский	1836	3,16	Александр Сергеевич Пушкин	<b>Добавить на книжную полку</b>
Руслан и Людмила	Поэма	Русский	1820	3,92	Александр Сергеевич Пушкин	<b>Добавить на книжную полку</b>

Рисунок 3.5 – Результат расширенного поиска книг

Чтобы изменить свою роль и расширить доступный функционал необходимо зарегистрироваться или авторизоваться, данная функция доступна всем трем ролям, однако имеет большой смысл только для роли «Гость». На рисунках 3.6 и 3.7 показаны соответствующие страницы.

### Регистрация

Зарегистрироваться

Уже есть аккаунт? Войти

Рисунок 3.6 – Регистрация

### Вход

Войти

Еще нет аккаунта? Зарегистрироваться

Рисунок 3.7 – Авторизация

В личном кабинете доступны различные функции для авторизованного пользователя и администратора. На рисунках 3.8 и 3.9 показаны панели личного кабинета для этих ролей соответственно.

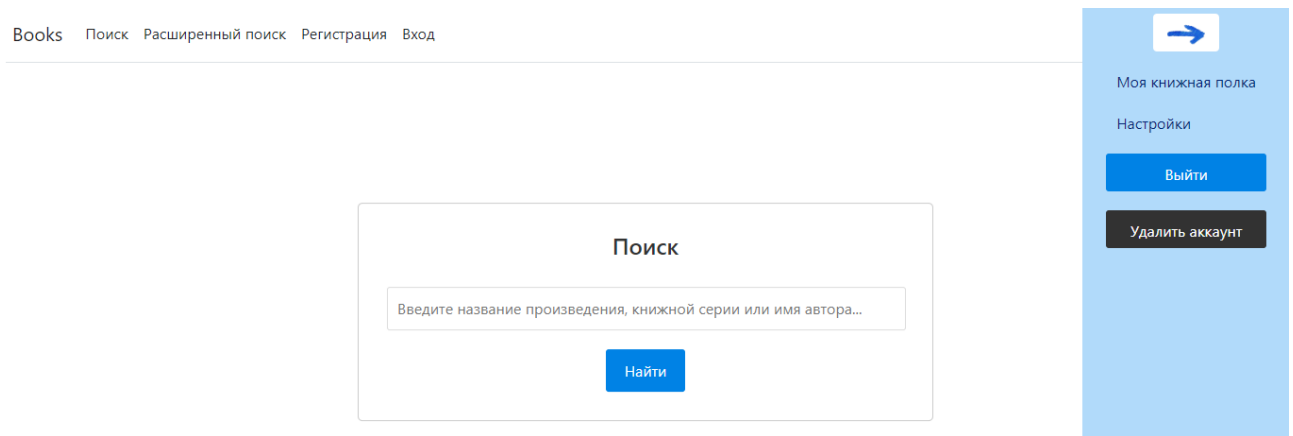


Рисунок 3.8 – Панель личного кабинета для роли «Авторизованный пользователь»

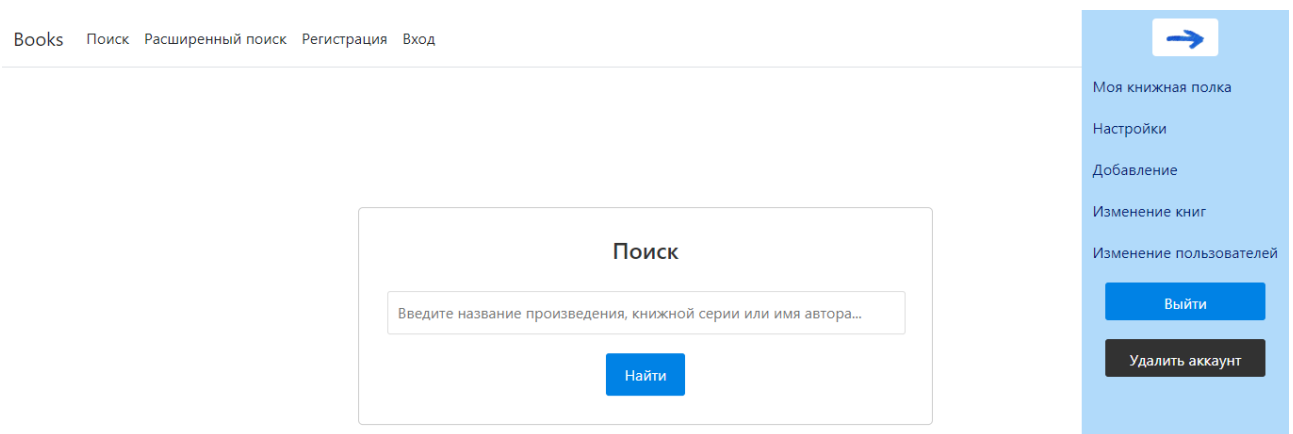


Рисунок 3.9 – Панель личного кабинета для роли «Администратор»

На рисунках 3.10 и 3.11 показаны страницы «книжной полки» и изменения пароля соответственно, данные старницы доступны обеим ролям из личного кабинета.



Количество книг: 4  
Средний рейтинг: 4,34

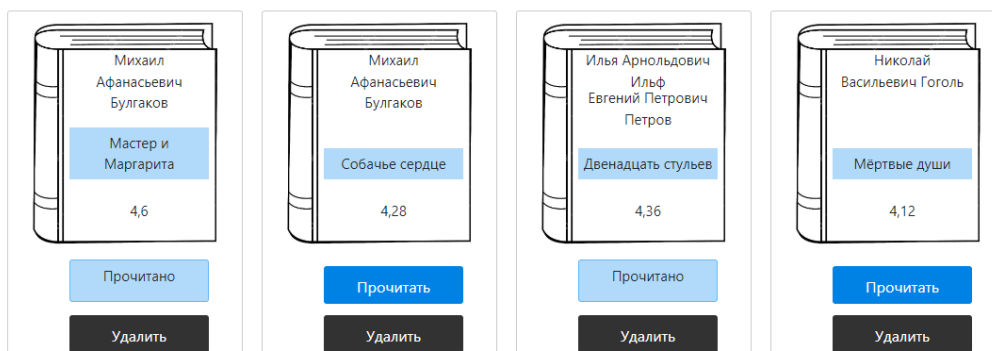


Рисунок 3.10 – «Книжная полка»

**Изменение пароля**

Рисунок 3.11 – Изменение пароля

Только администратору доступны страницы добавления и изменения, которые отображены на рисунках 3.12, 3.13 и 3.14.

## Добавление книги:

Название	Основной жанр	Язык оригинала	Год выпуска	Рейтинг	Добавить
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	

## Добавление автора:

Имя	Основной жанр	Год рождения	Год смерти	Страна проживания	Добавить
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	

## Добавление связи книга-автор:

ID книги	ID автора	Добавить
<input type="text"/>	<input type="text"/>	

Рисунок 3.12 – Добавление новых записей в базу данных

## Книги:

ID	Название	Основной жанр	Язык оригинала	Год выпуска	Рейтинг	
1	Мастер и Маргарита	<input type="text" value="Роман"/>	<input type="text" value="Русский"/>	<input type="text" value="1928"/>	<input type="text" value="4,6"/>	<input type="button" value="Изменить"/>
2	Собаачье сердце	<input type="text" value="Роман"/>	<input type="text" value="Русский"/>	<input type="text" value="1925"/>	<input type="text" value="4,28"/>	<input type="button" value="Изменить"/>
3	Двенадцать стульев	<input type="text" value="Роман"/>	<input type="text" value="Русский"/>	<input type="text" value="1928"/>	<input type="text" value="4,36"/>	<input type="button" value="Изменить"/>
4	Мёртвые души	<input type="text" value="Роман"/>	<input type="text" value="Русский"/>	<input type="text" value="1842"/>	<input type="text" value="4,12"/>	<input type="button" value="Изменить"/>
5	Граф Монте-Кристо	<input type="text" value="Роман"/>	<input type="text" value="Французский"/>	<input type="text" value="1844"/>	<input type="text" value="4,54"/>	<input type="button" value="Изменить"/>
6	Золотой теленок	<input type="text" value="Роман"/>	<input type="text" value="Русский"/>	<input type="text" value="1931"/>	<input type="text" value="3,96"/>	<input type="button" value="Изменить"/>
7	Три товарища	<input type="text" value="Роман"/>	<input type="text" value="Немецкий"/>	<input type="text" value="1936"/>	<input type="text" value="4,16"/>	<input type="button" value="Изменить"/>

Рисунок 3.13 – Изменение информации о существующих книгах в базе данных

Пользователи:

ID	Логин	Права доступа	
2	saaa20u748	Администратор	Изменить
3	asdfgh	Пользователь	Изменить
4	qwerty	Пользователь	Изменить
5	master12	Пользователь	Изменить

Пользователь

Администратор

Пользователь

Рисунок 3.14 – Изменение прав зарегистрированных пользователей в базе данных

## 4 Исследовательская часть

### 4.1 Описание эксперимента

Индексы в базах данных — это структуры данных, создаваемые для улучшения производительности операций в таблицах базы данных. Они играют важную роль в оптимизации запросов и ускорении доступа к данным [13].

Преимущества использования индексов:

- ускорение поиска;
- поддержка сортировки;
- оптимизация фильтрации;
- улучшение уникальности;
- улучшение производительности соединений;
- поддержка внешних ключей [13].

Недостатки использования индексов:

- дополнительная занимаемая память;
- трудоемкое создание и обслуживание;
- замедление операций модификации [13].

В ходе выполнения курсовой работы было решено провести эксперимент, в ходе которого будет рассмотрена зависимость времени выполнения сортировки от количества строк в таблице и наличия индексов. Так как ключевой сущностью для данного проекта является Book, а атрибутом, по которому наиболее часто сортируют данные, — Rating, то именно они и были выбраны для проведения эксперимента. В листинге 4.1 показано создание кластеризованного и некластеризованного индекса для таблицы NewBook, которая заполняется определенным количеством записей из основной таблицы Book.

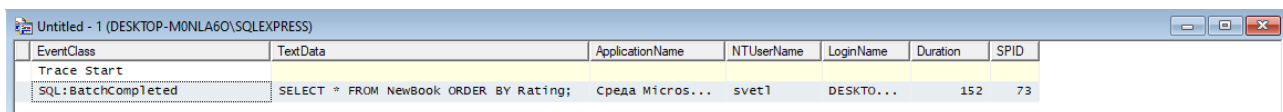
Листинг 4.1 – Создание кластеризованного и некластеризованного индекса для эксперимента

```
1 CREATE CLUSTERED INDEX book_rating_ind ON NewBook(Rating);
2 CREATE NONCLUSTERED INDEX book_rating_ind ON NewBook(Rating);
```

Замеры времени выполнялись при помощи специальных механизмов, представляемых системой управления базами данных, однако наиболее распространенный вариант замера, отображенный в листинге 4.2, не подошел в силу необходимости замерять сравнительно быстрые запросы, поэтому было решено использовать такой инструмент, как SQL Server Profiler. Пример информации, предоставляемой данным инструментом после выполнения запроса (в данном случае применена настройка, необходимая для данного эксперимента) показан на рисунке 4.1.

Листинг 4.2 – Стандартный способ измерения времени выполнения запроса

```
1 SET STATISTICS TIME ON;
2 ...
3 SET STATISTICS TIME OFF;
```



The screenshot shows the SQL Server Profiler interface with a trace event selected. The event details are as follows:

EventClass	TextData	ApplicationName	NTUserName	LoginName	Duration	SPID
Trace Start						
SQL:BatchCompleted	SELECT * FROM NewBook ORDER BY Rating;	Среда Micros...	svet1	DESKTO...	152	73

Рисунок 4.1 – Пример работы SQL Server Profiler

Тестирование программного обеспечения было проведено на устройстве со следующими техническими характеристиками:

- 1) операционная система Windows-10, 64-bit;
- 2) оперативная память 8 ГБ;
- 3) процессор Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, 2304 МГц, ядер 2, логических процессоров 4.

## 4.2 Результаты эксперимента

Ниже приведены результаты проведенного эксперимента, который был описан выше.

Таблица 4.1 – Зависимость времени выполнения запроса от числа записей в таблице без индексирования и с различными типами индексирования

Количество записей в таблице	Время выполнения запроса без индекса, мс	Время выполнения запроса с кластеризованным индексом, мс	Время выполнения запроса с некластеризованным индексом, мс
10	2	1	1
100	15	7	10
500	76	56	67
1000	99	63	89
5000	145	130	136

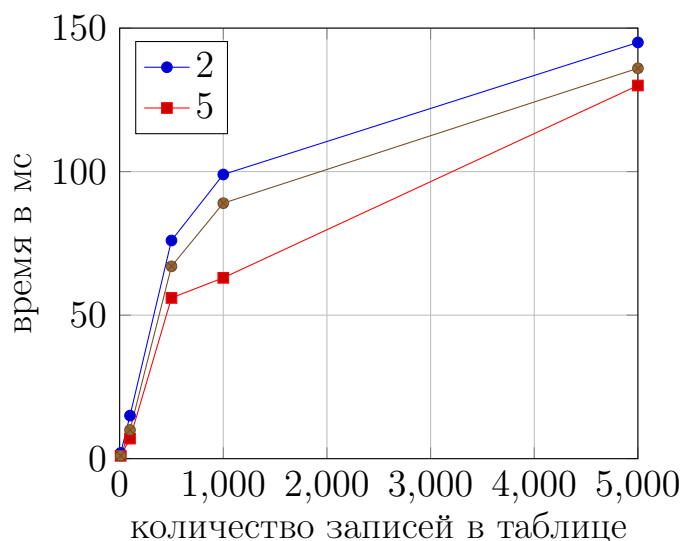


Рисунок 4.2 – График зависимости времени выполнения запроса от числа записей в таблице без индексирования и с различными типами индексирования

## Вывод

Благодаря данным, полученным в ходе выполнения эксперимента, можно сделать вывод, что использование кластеризованного индекса уменьшает время выполнения запроса в среднем в 1,5 раза, а некластеризованного — в 1,3 раза.

## ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была достигнута поставленная **цель**: разработана база данных для книжной информационной системы.

Все **задачи** курсовой работы выполнены:

- проведен анализ существующих решений;
- проанализированы существующие модели баз данных и выбрана подходящая;
- формализованы задачи и определены необходимые роли;
- спроектирована и разработана база данных;
- спроектировано и разработано WEB-приложение;
- проведено исследование характеристик разработанного программного обеспечения.

А также проведен эксперимент, целью которого являлось рассмотрение зависимости времени выполнения сортировки от количества строк в таблице и наличия индексов. Результаты данного эксперимента показали, что использование кластеризованного индекса уменьшает время выполнения запроса в среднем в 1,5 раза, а некластеризованного — в 1,3 раза.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ленский Б. В. Воропаев А. Н. Соловьева Е. В. Зорина С. Ю. Столяров А. А. Казакова А. М. Книжный рынок России. Состояние, тенденции и перспективы развития: Отраслевой доклад: Москва: Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации, 2021.
2. Психологические факторы, влияющие на принятие решения в UX или как сократить когнитивную нагрузку [Электронный ресурс]. URL: <https://goo.su/1aZxeRs> (дата обращения: 27.07.2023).
3. Гаврилова Ю. М. Конспект лекций по базам данных // Московский государственный университет им. Н. Э. Баумана. 2022.
4. Дореляционные модели представления данных [Электронный ресурс]. URL: <https://pandia.ru/text/78/445/11014.php> (дата обращения: 27.07.2023).
5. Иерархическая база данных: что это такое, пример модели БД организации [Электронный ресурс]. URL: <https://goo.su/hrYmEP> (дата обращения: 27.07.2023).
6. В чем состоят преимущества и недостатки иерархической модели данных [Электронный ресурс]. URL: [https://studbooks.net/2263476/informatika/sostoyat\\_preimuschestva\\_nedostatki\\_ierarhicheskoy\\_modeli\\_dannyh](https://studbooks.net/2263476/informatika/sostoyat_preimuschestva_nedostatki_ierarhicheskoy_modeli_dannyh) (дата обращения: 27.07.2023).
7. Сетевые базы данных [Электронный ресурс]. URL: <https://mega-obzor.ru/setevye-bazy-dannyh.html> (дата обращения: 27.07.2023).
8. Сетевая модель данных, ее достоинства и недостатки [Электронный ресурс]. URL: <https://studfile.net/preview/16467496/page:8/> (дата обращения: 27.07.2023).
9. Модели баз данных: учебное пособие / О.Е. Аврунев, В.М. Стасышин. – Новосибирск: Изд-во НГТУ, 2018. – 124 с.



10. Что такое реляционная база данных? [Электронный ресурс]. URL: <https://aws.amazon.com/ru/relational-database/> (дата обращения: 27.07.2023).
11. Что такое реляционная база данных? [Электронный ресурс]. URL: <https://appmaster.io/ru/blog/cto-takoe-reliatsionnaia-baza-dannykh> (дата обращения: 27.07.2023).
12. Лекции по базам данных [Электронный ресурс]. URL: <https://studfile.net/preview/7028484/> (дата обращения: 27.07.2023).
13. Использование и плюсы и минусы индексов [Электронный ресурс]. URL: [https://russianblogs.com/article/42532439756/#google\\_vignette](https://russianblogs.com/article/42532439756/#google_vignette) (дата обращения: 22.08.2023).

# ПРИЛОЖЕНИЕ А

## Презентация