Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**«БАЗА ДАННЫХ КНИЖНОГО МАГАЗИНА»**

БГУИР КП 1-40 01 01 020 ПЗ

Студент Панкратьев Е.С.

Руководитель Марина И.М.

Минск 2025

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2025г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Панкратьеву Егору Сергеевичу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *«База данных книжного магазина»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

2. Срок сдачи законченной работы *12.05.2025г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Среда проектирования DataGrip, MySQL, SQL.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ предметной области \_*\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_

*2 Анализ и разработка функциональных требований\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Инфологическая модель предметной области\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Подробное описание бизнес-логики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*5 Тестирование базы данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Марина И.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к – 15 % готовности работы;

раздел 2 к – 30 % готовности работы;

разделы 3,4 к – 60 % готовности работы;

раздел 5 к – 90 % готовности работы;

оформление пояснительной записки и графического материала к – 100 % готовности работы.

Защита курсового проекта с\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Марина И.М.*

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_Панкратьев Е.С. 16.02.2025г.*

*(дата и подпись студента)*

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc197715601)

[1 Анализ Предметной области 7](#_Toc197715602)

[1.1 Анализ существующих аналогов 7](#_Toc197715603)

[1.1.1 Интернет-магазин «Ozon» 7](#_Toc197715604)

[1.1.2 Интернет-магазин «Лабиринт» 8](#_Toc197715605)

[1.2 Постановка задачи 9](#_Toc197715606)

[1.2.1 Назначение разработки 9](#_Toc197715607)

[1.2.2 Перечень функциональных требований 9](#_Toc197715608)

[1.2.3 Входные данные 9](#_Toc197715609)

[1.2.4 Выходные данные 10](#_Toc197715610)

[1.2.5 Требования к составу и параметрам технических средств 10](#_Toc197715611)

[1.2.6 Требования к информационной и программной совместимости 10](#_Toc197715612)

[2 Анализ и разработка функциональных требований 11](#_Toc197715613)

[2.1 Используемые технологии при разработке 11](#_Toc197715614)

[2.2 Функциональные требования к базе данных 12](#_Toc197715615)

[2.3 Ключевые особенности разрабатываемой базы данных 13](#_Toc197715616)

[2.4 Алгоритм обработки данных и методы взаимодействия 13](#_Toc197715617)

[3 Инфологическая модель предметной области 14](#_Toc197715618)

[3.1 Модель базы данных 14](#_Toc197715619)

[3.2 Словесное описание модели 15](#_Toc197715620)

[4 Подробное описание бизнес-логики 18](#_Toc197715621)

[4.1 Хранимые процедуры и функции 18](#_Toc197715622)

[4.1.1 Обновление рейтинга книги 18](#_Toc197715623)

[4.1.2 Получение общего количества экземпляров книги 18](#_Toc197715624)

[4.1.3 Оформление заказа пользователя 18](#_Toc197715625)

[4.1.4 Подсчёт количества книг автора 19](#_Toc197715626)

[4.2 Триггеры 19](#_Toc197715627)

[4.2.1 Автоматический пересчёт рейтинга книги 20](#_Toc197715628)

[4.2.2 Агрегация количества в корзине 20](#_Toc197715629)

[4.2.3 Валидация цены и года для книг 21](#_Toc197715630)

[4.2.4 Валидация значения рейтинга 21](#_Toc197715631)

[4.2.5 Проверка целостности дат автора 22](#_Toc197715632)

[4.2.6 Проверка контрактных дат сотрудников 22](#_Toc197715633)

[4.2.7 Валидация процента скидки 23](#_Toc197715634)

[4.2.8 Контроль остатков на складе 23](#_Toc197715635)

[4.2.9 Проверка позиций заказа 24](#_Toc197715636)

[4.2.10 Валидация и закрытие счёта 24](#_Toc197715637)

[4.2.11 Расчёт итоговой суммы в счёте 25](#_Toc197715638)

[4.3 Представления 25](#_Toc197715639)

[4.3.1 Популярные книги по количеству заказов 25](#_Toc197715640)

[4.3.2 Детали заказов пользователей 26](#_Toc197715641)

[4.3.3 Полная информация о книге 26](#_Toc197715642)

[4.3.4 Количество проданных экземпляров по авторам 27](#_Toc197715643)

[4.3.5 Остатки книг по складам 27](#_Toc197715644)

[4.3.6 Активность пользователей (число заказов и общая сумма) 27](#_Toc197715645)

[4.3.7 Средний рейтинг и число отзывов по книгам 28](#_Toc197715646)

[4.3.8 Продажи по месяцам (количество заказов и выручка) 28](#_Toc197715647)

[5 Тестирование базы данных 29](#_Toc197715648)

[Заключение 32](#_Toc197715649)

[Список использованной литературы 33](#_Toc197715650)

[Приложение А 34](#_Toc197715651)

Введение

Книги остаются важнейшим источником знаний, культурного наследия и вдохновения, несмотря на стремительное развитие цифровых технологий. Сегодня книжные магазины активно переходят в онлайн-среду, предоставляя пользователям удобный доступ к огромному ассортименту литературы, расширенным возможностям поиска, рекомендациям, отзывам и быстрой доставке. Эффективная работа таких интернет-магазинов невозможна без надёжной информационной системы, в центре которой находится грамотно спроектированная база данных.

База данных книжного магазина должна учитывать большое количество разнообразных сущностей и связей между ними. Это не только сами книги, но и авторы, жанры, языки, серии, издательства, а также данные о пользователях, их предпочтениях, заказах, корзинах, списках желаемого, отзывах и рейтингах. Кроме того, необходимо обеспечить хранение информации о сотрудниках магазина, поставщиках, складах, остатках продукции, скидках, оплате и доставке. Такая структура требует продуманного подхода к проектированию базы данных с учётом нормализации, обеспечения целостности данных и логики бизнес-процессов.

Особое внимание должно быть уделено функциональности базы данных: поддержка фильтрации книг по различным критериям, отслеживание истории заказов, учёт складских остатков, применение скидок, реализация системы рецензий и оценок. Это требует высокой гибкости и масштабируемости структуры, чтобы база могла развиваться вместе с расширением бизнеса и пользовательских потребностей.

Целью данной курсовой работы является разработка базы данных для книжного интернет-магазина на платформе MySQL, которая позволит эффективно хранить и обрабатывать информацию о товарах, клиентах, заказах и операциях, обеспечивая стабильную работу информационной системы и удобство для конечного пользователя.

# Анализ Предметной области

## Анализ существующих аналогов

### Интернет-магазин «Ozon»

Ozon – один из крупнейших маркетплейсов с каталогом более 1 млн наименований книг и прочих товаров [1].

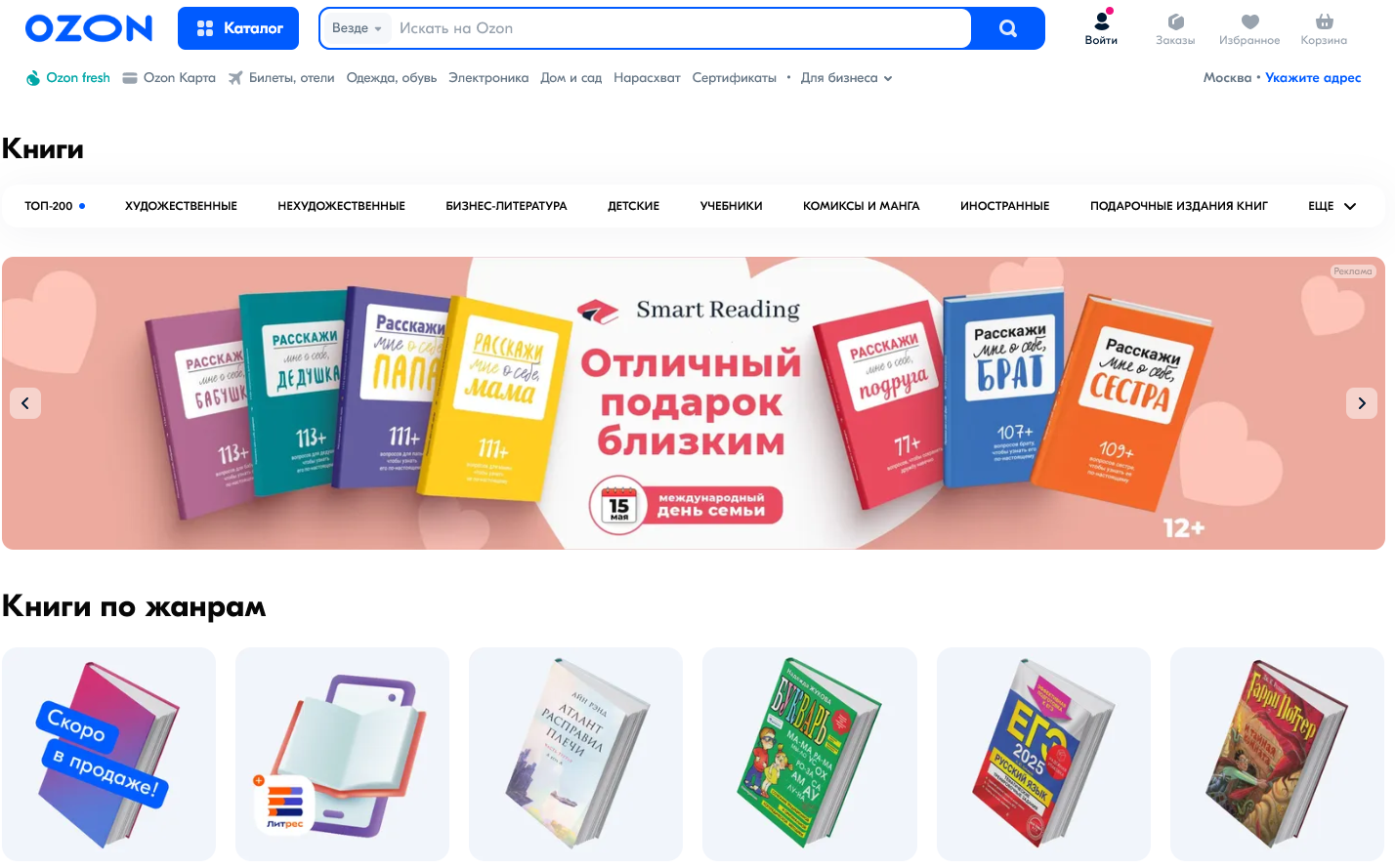


Рисунок 1.1.1 – Интерфейс интернет-магазина «Ozon»

Ozon предлагает обширный каталог книг с возможностью точной фильтрации по жанру, автору, формату (электронная/бумажная книга), языку и ценовому диапазону.

Строка поиска поддерживает поиск по части названия и автодополнение, что ускоряет нахождение нужного издания даже при неточном вводе.

В карточке товара выводятся блоки «Похожие товары», «С этим товаром покупают» и персонализированные рекомендации на основе истории просмотров.

Главная и категории насыщены рекламными баннерами и акциями, что при медленном соединении может замедлять загрузку страниц и отвлекать внимание пользователя.

При сортировке доступны опции по популярности, рейтингу, цене и новизне, что позволяет удовлетворить потребности как спонтанных, так и целенаправленных покупателей.

### Интернет-магазин «Лабиринт»

Ещё одним важным примером является интернет-магазин «Лабиринт» [2].

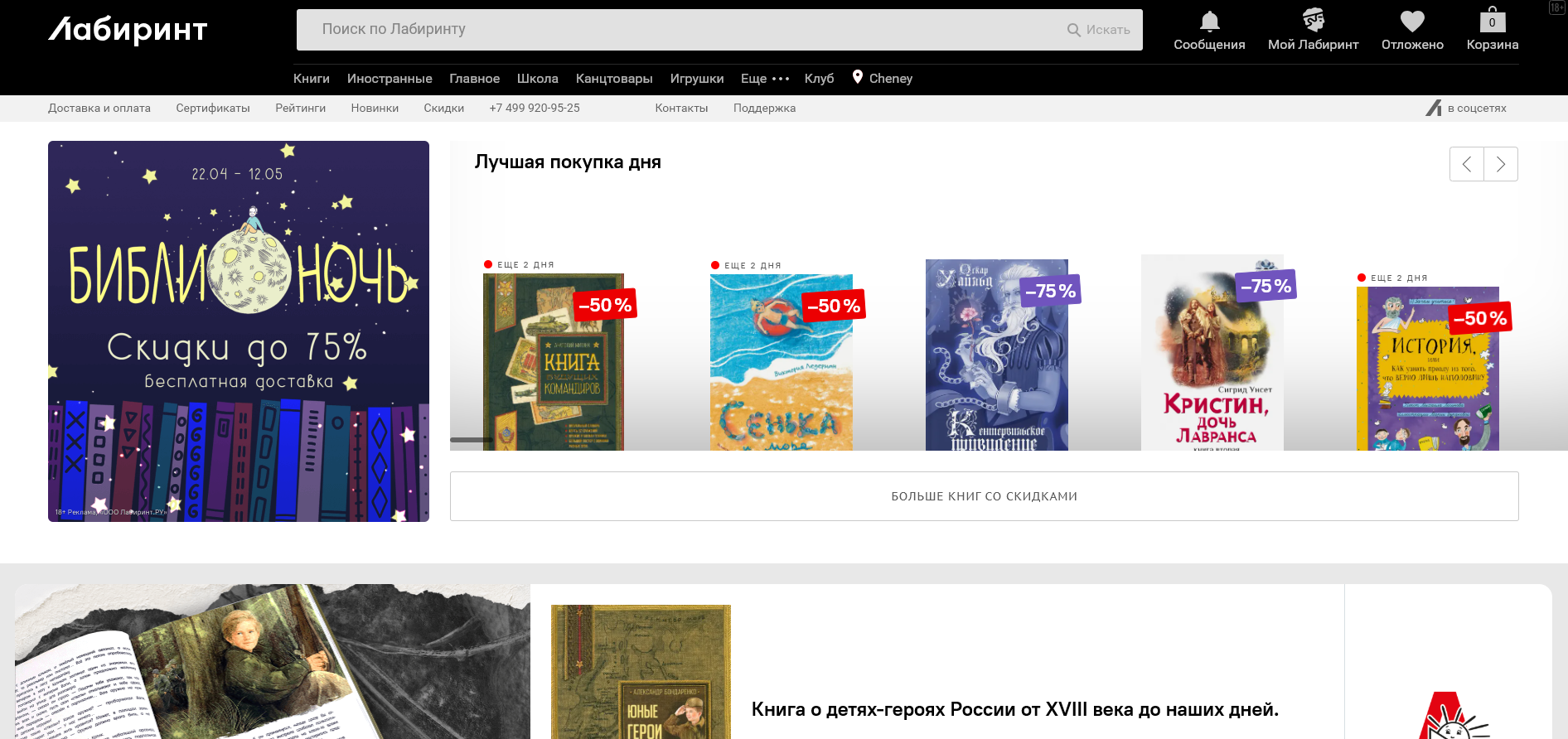


Рисунок 1.1.2 – Интерфейс интернет-магазина «Лабиринт»

«Лабиринт» характеризуется лаконичным горизонтальным меню, раскрывающимся в дерево подкатегорий по наведению, что упрощает навигацию по основным разделам (художественная, учебная, детская литература и др.).

Каталог оснащён многоуровневой фильтрацией: жанр, формат, возрастная категория, издатель и ценовой диапазон. Отдельный блок демонстрирует редакционные и пользовательские подборки.

В карточке товара доступны отзывы читателей, рейтинги, подробное описание и блоки рекомендаций «Читайте также» и «С этим товаром покупают».

Пользователи отмечают, что интерфейс «Лабиринта» быстро откликается на действия и не перегружен рекламой, однако локальная персонализация рекомендаций реализована не столь глубоко, как на крупных маркетплейсах.

При незагруженной фильтрации иногда показывает книги, которых нет в наличии, что может сбивать с толку пользователя при выборе издания для немедленной покупки.

## Постановка задачи

### Назначение разработки

В рамках данной курсовой работы планируется разработать базу данных, достаточную для построения программного средства – книжного интернет-магазина. Для выполнения поставленной задачи необходимо:

* разработать физическую модель базы данных для реляционной базы данных MySQL 8.4;
* протестировать базу данных на корректность и полноту**.**

### Перечень функциональных требований

Система должна реализовывать следующие основные функции:

– хранение информации о книгах, их жанрах, авторах, издателях и сериях;

– поддержка многоязычности книг;

– хранение информации о пользователях, сотрудниках, ролях и правах доступа;

– поддержка корзины покупателя, заказов, платежей и выставления счетов;

– хранение отзывов, рейтингов, комментариев и списков желаемого (wishlist);

– управление запасами товаров, поставками от поставщиков и заявками на пополнение;

– ведение истории изменения статусов заказов и поставок.

### Входные данные

В процессе проектирования требований были определены следующие входные данные для системы:

1. Пользователи:

– Email, пароль, ФИО, дата рождения, телефон, пол;

– Роли и пользовательские предпочтения.

1. Книги и связанная информация:

– Название, описание, цена, год издания, издательство, язык, жанры, теги;

– Авторы, национальности, пол, биография;

– Серии книг, тома.

1. Система заказов:

– Информация о корзинах, заказах, статусах, способах оплаты, скидках;

– Адреса доставки и данные счетов.

1. Складской и закупочный учет:

– Склады, остатки книг, поставщики, заявки на пополнение, статусы поставок.

1. Обратная связь:

– Отзывы, оценки, комментарии к отзывам.

### Выходные данные

Выходными данными разрабатываемой базы данных являются:

– результат SQL-запросов (поиск книг, пользователи, заказы, остатки и т.д.);

– автоматические действия процедур и триггеров (например, обновление рейтингов книг, фиксация даты подписания контракта);

– отчёты и представления для анализа: например, популярные книги, активные пользователи, статистика заказов.

### Требования к составу и параметрам технических средств

Программное средство, реализованное с использованием разрабатываемой базой данных, должно функционировать на персональных компьютерах со следующими характеристиками:

* СУБД система должна работать на MySQL 8.4;
* процессор Intel Core i5 3.3 Мгц и лучше;
* оперативная память 4 GB 1600 MHz DDR3 и лучше;
* накопитель HDD или SSD объемом 64 GB и больше;
* операционные системы – поддержка работы в средах Windows, Linux и macOS, что обеспечивает гибкость развертывания.

### Требования к информационной и программной совместимости

Система должна обеспечивать:

* возможность подключения через различные драйверы (MySQL Connector, ODBC, JDBC);
* обеспечение совместимости с актуальными версиями СУБД (MySQL 8.0+, MariaDB 10.6+). Поддержка стандартных функций и синтаксиса, без использования устаревших или специфичных для конкретной версии возможностей;
* **экспорт и импорт данных.** Возможность выгрузки данных в стандартные форматы.

# Анализ и разработка функциональных требований

## Используемые технологии при разработке

В данном проекте в качестве основной системы управления базами данных используется MySQL 8.4 — надёжная и широко распространённая реляционная СУБД с открытым исходным кодом. MySQL обеспечивает высокую производительность, масштабируемость и стабильность, что делает её отличным выбором для построения комплексной информационной системы книжного интернет-магазина. Благодаря поддержке транзакций, хранимых процедур и триггеров MySQL позволяет реализовать сложную бизнес-логику, необходимую для таких операций, как оформление заказов, управление складскими запасами, обработка платежей и регистрация пользователей [3].

MySQL 8.4 предлагает следующие ключевые возможности, активно применяемые в проекте:

транзакционность — обеспечивает целостность данных при работе с заказами, оплатой и запасами книг;

хранимые процедуры и триггеры — автоматизируют рутинные процессы, такие как логирование, обновление остатков на складе и применение скидок;

оптимизация SQL-запросов и представления — ускоряют выполнение аналитических запросов и формирование отчетов;

поддержка внешних ключей — позволяет выстраивать сложные связи между сущностями и обеспечивает целостность данных на уровне базы.

Для администрирования и проектирования базы данных используется DataGrip — универсальный и мощный инструмент, предназначенная для работы с различными СУБД, включая MySQL. В отличие от специализированных инструментов, DataGrip предоставляет расширенные средства анализа, написания и отладки SQL-запросов, а также ряд возможностей, полезных при командной разработке:

* визуальное построение структуры БД и ER-диаграмм;
* интеллектуальная подсветка синтаксиса, автодополнение и проверка SQL-запросов;
* удобная навигация между таблицами, внешними ключами и представлениями;
* инструменты для управления миграциями и фиксации изменений в схеме базы данных.

Использование связки MySQL и DataGrip обеспечивает высокую гибкость и продуктивность в разработке, а также способствует удобной поддержке и масштабированию информационной системы в будущем.

## Функциональные требования к базе данных

Разрабатываемая база данных для информационной системы книжного интернет-магазина должна обеспечивать реализацию следующих ключевых функций:

* хранение информации о книгах. Необходимо хранение данных о книгах: название, авторы, жанры, издательство, год издания, ISBN, аннотация, количество страниц, цена, остаток на складе и рейтинг. Структура базы должна обеспечивать быстрый доступ к информации для отображения в каталоге, поиска и фильтрации;
* учёт пользователей. База данных должна поддерживать регистрацию и авторизацию пользователей, хранить информацию об аккаунтах (имя, логин, хеш пароля, email, дата регистрации и др.), а также обеспечивать связь с их действиями: заказы, отзывы, корзина, история просмотров и избранное;
* управление заказами и корзиной. Система должна обеспечивать формирование заказов с указанием состава (список книг), способа оплаты, статуса доставки и даты оформления. Также необходимо реализовать хранение содержимого «корзины» пользователя до момента подтверждения заказа;
* отзывы и рейтинги. Поддерживается хранение отзывов и оценок пользователей по конкретным книгам. Требуется связка отзывов с пользователями и книгами, а также реализация механизма подсчета средней оценки книги для отображения рейтингов;
* поиск и фильтрация. База должна поддерживать гибкий поиск книг по названию, автору, жанру, издательству и другим характеристикам. Фильтрация по цене, году издания, рейтингу и наличию также должна быть реализована. Для повышения производительности применяются индексы по основным поисковым полям;
* администрирование и управление каталогом. Администраторы должны иметь возможность добавлять, редактировать и удалять записи о книгах, управлять пользователями, обрабатывать заказы и модерировать отзывы;
* целостность и безопасность данных. Используются ограничения целостности (внешние ключи, проверки), хранимые процедуры и триггеры для контроля корректности операций. Важные данные, такие как пароли, должны храниться в зашифрованном виде. Реализуется контроль доступа на уровне прав пользователей;
* аналитика и отчётность. Система должна поддерживать сбор и хранение статистических данных: количество продаж, популярные книги, активность пользователей. Это необходимо для формирования отчётов, анализа поведения покупателей и управления ассортиментом.

## Ключевые особенности разрабатываемой базы данных

Ключевыми особенностями разрабатываемой базы данных являются:

* высокая производительность — обеспечение быстрой обработки запросов за счёт оптимизации структуры данных и использования индексирования;
* масштабируемость — гибкая модульная архитектура, позволяющая безболезненно расширять систему новыми сущностями и функционалом по мере роста требований;
* целостность данных — механизмы автоматической проверки и поддержания согласованности информации при выполнении операций добавления, обновления и удаления;
* автоматизация бизнес-логики — встроенные средства для выполнения ключевых операций (пересчёт агрегированных показателей, обновление связанных данных) без вмешательства приложений;
* поддержка аналитики — возможности по сбору и подготовке агрегированных данных для отчётов и аналитических выборок.

## Алгоритм обработки данных и методы взаимодействия

Обработка данных в базе данных книжного интернет-магазина реализуется с учётом требований к надёжности, согласованности и автоматизации:

* автоматическая обработка событий. Используются триггеры для автоматического выполнения операций при изменении данных, таких как пересчёт агрегированных значений или обновление связанных записей;
* валидация данных на уровне СУБД. Реализована проверка целостности и корректности информации перед выполнением транзакций. Это позволяет предотвратить логические ошибки и обеспечить выполнение бизнес-правил;
* инкапсуляция логики в процедурах. Бизнес-операции выполняются через хранимые процедуры, что упрощает повторное использование и централизованное управление логикой взаимодействия;
* поддержка аналитики и отчётности. Для анализа данных используются представления, объединяющие информацию из разных таблиц. Это позволяет формировать отчёты без прямой нагрузки на основную структуру данных и упрощает работу внешних сервисов и аналитических инструментов.

# Инфологическая модель предметной области

## Модель базы данных

База данных спроектирована в DataGrip, что позволило объединить инфологическую и даталогическую модели в единую схему [4]. В результате полученная модель охватывает ключевые элементы предметной области — сведения о книгах и их характеристиках, каталог, покупки и корзину, а также информацию о пользователях, их отзывах и рейтингах.

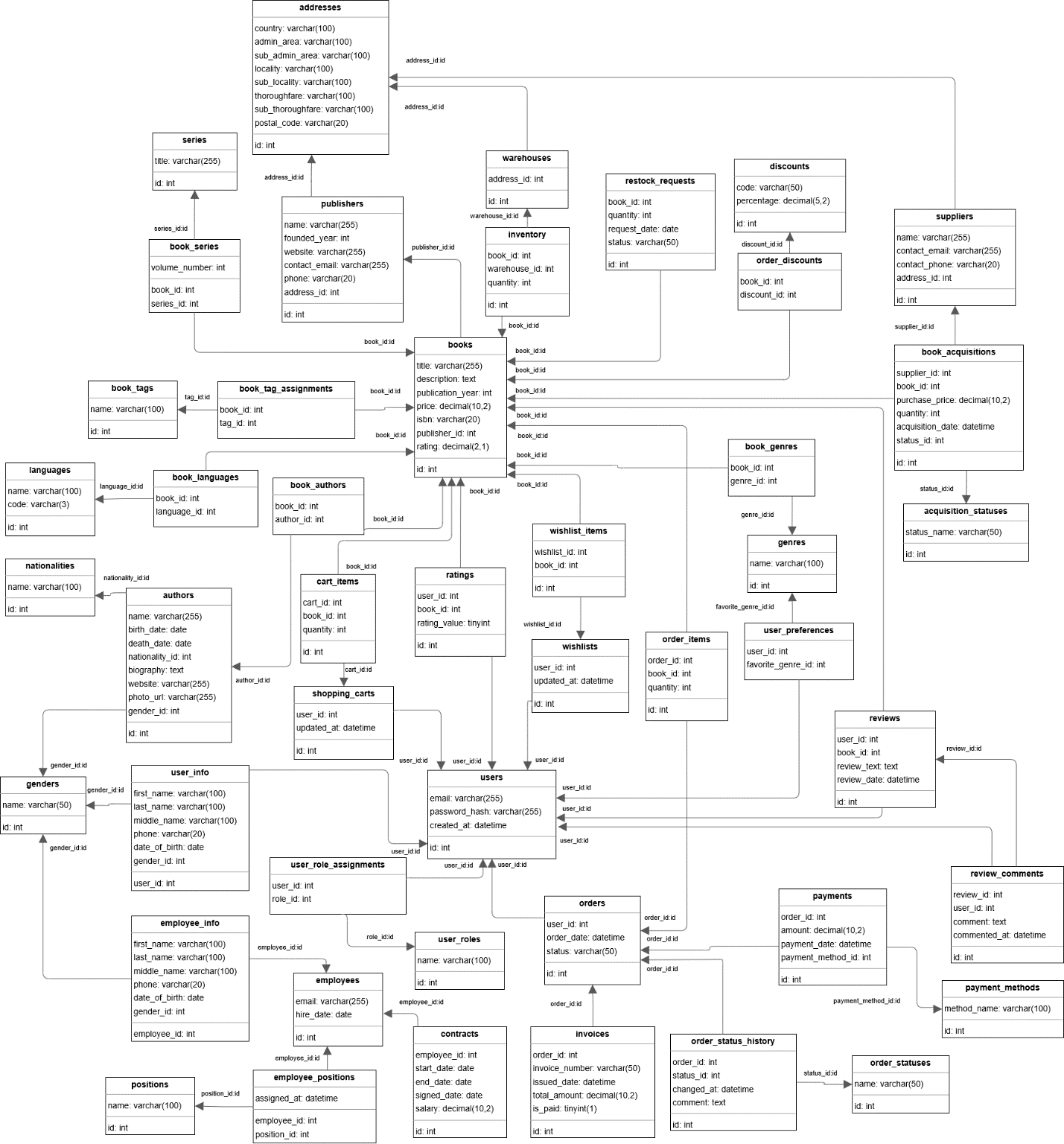


Рисунок 3.1 – Даталогическая модель базы данных

## Словесное описание модели

Для разработки базы данных были выбраны следующие сущности с определенными полями:

* languages справочник языков. Поля: уникальный идентификатор (id), название языка (name), ISO-код (code);
* genres справочник жанров. Поля: уникальный идентификатор (id), название жанра (name);
* addresses адреса. Поля: уникальный идентификатор (id), страна (country), область/регион (admin\_area), район (sub\_admin\_area), город (locality), микрорайон/квартал (sub\_locality), улица (thoroughfare), номер дома (sub\_thoroughfare), почтовый индекс (postal\_code);
* publishers издатели. Поля: уникальный идентификатор (id), название (name), год основания (founded\_year), сайт (website), email (contact\_email), телефон (phone), ссылка на адрес (address\_id);
* series серии книг. Поля: уникальный идентификатор (id), название серии (title);
* books книги. Поля: уникальный идентификатор (id), название (title), описание (description), год публикации (publication\_year), цена (price), ISBN (isbn), ссылка на издателя (publisher\_id), средний рейтинг (rating);
* book\_languages связь книг и языков. Поля: id книги (book\_id), id языка (language\_id);
* nationalities справочник национальностей. Поля: уникальный идентификатор (id), название (name);
* genders справочник полов. Поля: уникальный идентификатор (id), название (name);
* authors авторы. Поля: уникальный идентификатор (id), имя (name), дата рождения (birth\_date), дата смерти (death\_date), ссылка на национальность (nationality\_id), биография (biography), сайт (website), ссылка на фото (photo\_url), ссылка на пол (gender\_id);
* book\_authors связь книг и авторов. Поля: id книги (book\_id), id автора (author\_id);
* book\_genres связь книг и жанров. Поля: id книги (book\_id), id жанра (genre\_id);
* book\_series связь книг и серий. Поля: id книги (book\_id), id серии (series\_id), номер тома в серии (volume\_number);
* book\_tags справочник тегов. Поля: уникальный идентификатор (id), название тега (name);
* book\_tag\_assignments присвоенные теги книгам. Поля: id книги (book\_id), id тега (tag\_id);
* users пользователи. Поля: уникальный идентификатор (id), email (email), хеш пароля (password\_hash), дата создания (created\_at);
* user\_roles справочник ролей. Поля: уникальный идентификатор (id), название роли (name);
* user\_role\_assignments связь пользователей и ролей. Поля: id пользователя (user\_id), id роли (role\_id);
* user\_info доп. информация о пользователях. Поля: id пользователя (user\_id), имя (first\_name), фамилия (last\_name), отчество (middle\_name), телефон (phone), дата рождения (date\_of\_birth), ссылка на пол (gender\_id);
* user\_preferences предпочтения пользователей. Поля: id пользователя (user\_id), id любимого жанра (favorite\_genre\_id);
* employees сотрудники. Поля: уникальный идентификатор (id), email (email), дата приёма на работу (hire\_date);
* employee\_info доп. информация о сотрудниках. Поля: id сотрудника (employee\_id), имя (first\_name), фамилия (last\_name), отчество (middle\_name), телефон (phone), дата рождения (date\_of\_birth), ссылка на пол (gender\_id);
* positions должности. Поля: уникальный идентификатор (id), название должности (name);
* employee\_positions связь сотрудников и должностей. Поля: id сотрудника (employee\_id), id должности (position\_id), дата назначения (assigned\_at);
* contracts контракты сотрудников. Поля: уникальный идентификатор (id), id сотрудника (employee\_id), дата начала (start\_date), дата окончания (end\_date), дата подписания (signed\_date), зарплата (salary);
* shopping\_carts корзины. Поля: уникальный идентификатор (id), id пользователя (user\_id), дата обновления (updated\_at);
* cart\_items товары в корзине. Поля: уникальный идентификатор (id), id корзины (cart\_id), id книги (book\_id), количество (quantity);
* orders заказы. Поля: уникальный идентификатор (id), id пользователя (user\_id), дата заказа (order\_date), статус (status);
* order\_statuses справочник статусов заказа. Поля: уникальный идентификатор (id), название статуса (name);
* order\_status\_history история статусов заказа. Поля: уникальный идентификатор (id), id заказа (order\_id), id статуса (status\_id), дата изменения (changed\_at), комментарий (comment);
* order\_items позиции заказа. Поля: уникальный идентификатор (id), id заказа (order\_id), id книги (book\_id), количество (quantity);
* payment\_methods способы оплаты. Поля: уникальный идентификатор (id), название метода (method\_name);
* payments платежи. Поля: уникальный идентификатор (id), id заказа (order\_id), сумма (amount), дата платежа (payment\_date), id способа оплаты (payment\_method\_id);
* invoices счета. Поля: уникальный идентификатор (id), id заказа (order\_id), номер счета (invoice\_number), дата выставления (issued\_date), общая сумма (total\_amount), признак оплаты (is\_paid);
* discounts скидки. Поля: уникальный идентификатор (id), код скидки (code), процент (percentage);
* order\_discounts применённые скидки к книгам. Поля: id книги (book\_id), id скидки (discount\_id);
* warehouses склады. Поля: уникальный идентификатор (id), ссылка на адрес (address\_id);
* inventory остатки на складе. Поля: уникальный идентификатор (id), id книги (book\_id), id склада (warehouse\_id), количество (quantity);
* suppliers поставщики. Поля: уникальный идентификатор (id), название (name), email (contact\_email), телефон (contact\_phone), ссылка на адрес (address\_id);
* restock\_requests запросы на пополнение. Поля: уникальный идентификатор (id), id книги (book\_id), количество (quantity), дата запроса (request\_date), статус (status);
* acquisition\_statuses статусы закупок. Поля: уникальный идентификатор (id), название статуса (status\_name);
* book\_acquisitions закупки книг. Поля: уникальный идентификатор (id), id поставщика (supplier\_id), id книги (book\_id), цена закупки (purchase\_price), количество (quantity), дата закупки (acquisition\_date), id статуса (status\_id);
* reviews отзывы пользователей. Поля: уникальный идентификатор (id), id пользователя (user\_id), id книги (book\_id), текст отзыва (review\_text), дата отзыва (review\_date);
* review\_comments комментарии к отзывам. Поля: уникальный идентификатор (id), id отзыва (review\_id), id пользователя (user\_id), текст комментария (comment), дата комментария (commented\_at);
* ratings оценки книг. Поля: уникальный идентификатор (id), id пользователя (user\_id), id книги (book\_id), значение оценки (rating\_value);
* wishlists избранные списки. Поля: уникальный идентификатор (id), id пользователя (user\_id), дата обновления (updated\_at);
* wishlist\_items элементы избранного. Поля: уникальный идентификатор (id), id списка (wishlist\_id), id книги (book\_id).

# Подробное описание бизнес-логики

## Хранимые процедуры и функции

Хранимая процедура – это подпрограмма, которая может быть параметризована и предназначена для выполнения множества операций с данными и структурами базы данных. Она сохраняется в самой базе данных и доступна как для вызова из других процедур и триггеров, так и для прямого исполнения [5].

### Обновление рейтинга книги

update\_book\_rating — процедура для обновления среднего рейтинга книги на основе оценок из таблицы ratings:

CREATE PROCEDURE update\_book\_rating(IN bookId INT)

BEGIN

DECLARE avgRating DECIMAL(2,1);

SELECT ROUND(AVG(rating\_value), 1) INTO avgRating

FROM ratings

WHERE book\_id = bookId;

UPDATE books

SET rating = avgRating

WHERE id = bookId;

END;

### Получение общего количества экземпляров книги

get\_book\_stock — функция возвращает общее количество доступных экземпляров книги на складе:

CREATE FUNCTION get\_book\_stock(bookId INT)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE total\_stock INT;

SELECT SUM(quantity) INTO total\_stock

FROM inventory

WHERE book\_id = bookId;

RETURN IFNULL(total\_stock, 0);

END;

### Оформление заказа пользователя

place\_order — процедура, оформляющая заказ по содержимому корзины пользователя и создающая историю статуса:

CREATE PROCEDURE place\_order(IN userId INT)

BEGIN

DECLARE cartId INT;

DECLARE newOrderId INT;

SELECT id INTO cartId

FROM shopping\_carts

WHERE user\_id = userId;

INSERT INTO orders(user\_id)

VALUES (userId);

SET newOrderId = LAST\_INSERT\_ID();

INSERT INTO order\_items(order\_id, book\_id, quantity)

SELECT newOrderId, book\_id, quantity

FROM cart\_items

WHERE cart\_id = cartId;

DELETE FROM cart\_items

WHERE cart\_id = cartId;

INSERT INTO order\_status\_history(order\_id, status\_id)

VALUES (

newOrderId,

(SELECT id FROM order\_statuses WHERE name = 'Создан' LIMIT 1)

);

END;

### Подсчёт количества книг автора

get\_author\_book\_count — функция возвращает количество книг, написанных указанным автором:

CREATE FUNCTION get\_author\_book\_count(authorId INT)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE book\_count INT;

SELECT COUNT(\*) INTO book\_count

FROM book\_authors

WHERE author\_id = authorId;

RETURN book\_count;

END;

## Триггеры

Триггер – это объект базы данных, содержащий набор действий, которые автоматически выполняются при возникновении определённого события. Он может срабатывать до, после или вместо операций вставки, обновления или удаления, помогая поддерживать целостность данных и автоматизировать повторяющиеся процессы [5].

### Автоматический пересчёт рейтинга книги

Пересчитывает и обновляет поле books.rating после любых изменений в таблице ratings (INSERT, UPDATE, DELETE).

CREATE TRIGGER trg\_after\_rating\_insert

AFTER INSERT ON ratings

FOR EACH ROW

BEGIN

CALL update\_book\_rating(NEW.book\_id);

END;

CREATE TRIGGER trg\_after\_rating\_update

AFTER UPDATE ON ratings

FOR EACH ROW

BEGIN

CALL update\_book\_rating(NEW.book\_id);

END;

CREATE TRIGGER trg\_after\_rating\_delete

AFTER DELETE ON ratings

FOR EACH ROW

BEGIN

CALL update\_book\_rating(OLD.book\_id);

END;

### Агрегация количества в корзине

При попытке вставить дублирующую запись в cart\_items увеличивает существующее quantity и отменяет вставку с ошибкой, чтобы не было двух строк на тот же товар.

CREATE TRIGGER trg\_update\_cart\_item

BEFORE INSERT ON cart\_items

FOR EACH ROW

BEGIN

DECLARE existing\_quantity INT;

SELECT quantity

INTO existing\_quantity

FROM cart\_items

WHERE cart\_id = NEW.cart\_id

AND book\_id = NEW.book\_id

LIMIT 1;

IF existing\_quantity IS NOT NULL THEN

UPDATE cart\_items

SET quantity = quantity + NEW.quantity

WHERE cart\_id = NEW.cart\_id

AND book\_id = NEW.book\_id;

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Дублирующая запись: количество обновлено.';

END IF;

END;

### Валидация цены и года для книг

Запрещает вставку и обновление книги с отрицательной ценой или годом публикации в будущем.

CREATE TRIGGER trg\_books\_before\_insert

BEFORE INSERT ON books

FOR EACH ROW

BEGIN

IF NEW.price < 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Цена книги не может быть отрицательной.';

END IF;

IF NEW.publication\_year IS NOT NULL

AND NEW.publication\_year > YEAR(CURDATE()) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Год публикации не может быть в будущем.';

END IF;

END;

CREATE TRIGGER trg\_books\_before\_update

BEFORE UPDATE ON books

FOR EACH ROW

BEGIN

IF NEW.price < 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Цена книги не может быть отрицательной.';

END IF;

IF NEW.publication\_year IS NOT NULL

AND NEW.publication\_year > YEAR(CURDATE()) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Год публикации не может быть в будущем.';

END IF;

END;

### Валидация значения рейтинга

Не допускает записи оценок вне диапазона 1–5.

CREATE TRIGGER trg\_ratings\_before\_insert

BEFORE INSERT ON ratings

FOR EACH ROW

BEGIN

IF NEW.rating\_value < 1 OR NEW.rating\_value > 5 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Значение рейтинга должно быть от 1 до 5.';

END IF;

END;

CREATE TRIGGER trg\_ratings\_before\_update

BEFORE UPDATE ON ratings

FOR EACH ROW

BEGIN

IF NEW.rating\_value < 1 OR NEW.rating\_value > 5 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Значение рейтинга должно быть от 1 до 5.';

END IF;

END;

### Проверка целостности дат автора

Запрещает ситуацию, когда дата рождения позже даты смерти.

CREATE TRIGGER trg\_authors\_before\_insert

BEFORE INSERT ON authors

FOR EACH ROW

BEGIN

IF NEW.birth\_date IS NOT NULL

AND NEW.death\_date IS NOT NULL

AND NEW.birth\_date > NEW.death\_date THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Дата рождения не может быть позже даты смерти.';

END IF;

END;

CREATE TRIGGER trg\_authors\_before\_update

BEFORE UPDATE ON authors

FOR EACH ROW

BEGIN

IF NEW.birth\_date IS NOT NULL

AND NEW.death\_date IS NOT NULL

AND NEW.birth\_date > NEW.death\_date THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Дата рождения не может быть позже даты смерти.';

END IF;

END;

### Проверка контрактных дат сотрудников

Гарантирует, что начало контракта предшествует его завершению.

CREATE TRIGGER trg\_contracts\_before\_insert

BEFORE INSERT ON contracts

FOR EACH ROW

BEGIN

IF NEW.start\_date >= NEW.end\_date THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Дата начала контракта должна быть раньше даты окончания.';

END IF;

END;

CREATE TRIGGER trg\_contracts\_before\_update

BEFORE UPDATE ON contracts

FOR EACH ROW

BEGIN

IF NEW.start\_date >= NEW.end\_date THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Дата начала контракта должна быть раньше даты окончания.';

END IF;

END;

### Валидация процента скидки

Не допускает отрицательных скидок или более 100 %.

CREATE TRIGGER trg\_discounts\_before\_insert

BEFORE INSERT ON discounts

FOR EACH ROW

BEGIN

IF NEW.percentage < 0 OR NEW.percentage > 100 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Процент скидки должен быть от 0 до 100.';

END IF;

END;

CREATE TRIGGER trg\_discounts\_before\_update

BEFORE UPDATE ON discounts

FOR EACH ROW

BEGIN

IF NEW.percentage < 0 OR NEW.percentage > 100 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Процент скидки должен быть от 0 до 100.';

END IF;

END;

### Контроль остатков на складе

Запрещает запись отрицательного количества, автоматически пополняет inventory при новых поступлениях (book\_acquisitions).

CREATE TRIGGER trg\_inventory\_before\_insert

BEFORE INSERT ON inventory

FOR EACH ROW

BEGIN

IF NEW.quantity < 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Количество на складе не может быть отрицательным.';

END IF;

END;

CREATE TRIGGER trg\_inventory\_before\_update

BEFORE UPDATE ON inventory

FOR EACH ROW

BEGIN

IF NEW.quantity < 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Количество на складе не может быть отрицательным.';

END IF;

END;

CREATE TRIGGER trg\_acquisitions\_after\_insert

AFTER INSERT ON book\_acquisitions

FOR EACH ROW

BEGIN

INSERT INTO inventory(book\_id, warehouse\_id, quantity)

VALUES (

NEW.book\_id,

(SELECT id FROM warehouses LIMIT 1),

NEW.quantity

)

ON DUPLICATE KEY

UPDATE quantity = quantity + NEW.quantity;

END;

### Проверка позиций заказа

Гарантирует, что quantity в order\_items всегда положительно.

CREATE TRIGGER trg\_order\_items\_before\_insert

BEFORE INSERT ON order\_items

FOR EACH ROW

BEGIN

IF NEW.quantity <= 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Количество в заказе должно быть положительным.';

END IF;

END;

CREATE TRIGGER trg\_order\_items\_before\_update

BEFORE UPDATE ON order\_items

FOR EACH ROW

BEGIN

IF NEW.quantity <= 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Количество в заказе должно быть положительным.';

END IF;

END;

### Валидация и закрытие счёта

Перед вставкой в payments проверяет, чтобы сумма не превышала остаток по счёту;

После вставки — подсчитывает все платежи и устанавливает invoices.is\_paid = TRUE, если счёт погашен.

CREATE TRIGGER trg\_payments\_before\_insert

BEFORE INSERT ON payments

FOR EACH ROW

BEGIN

DECLARE inv\_total DECIMAL(10,2);

SELECT total\_amount

INTO inv\_total

FROM invoices

WHERE order\_id = NEW.order\_id

AND is\_paid = FALSE

LIMIT 1;

IF NEW.amount > inv\_total THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Сумма платежа превышает сумму по счёту.';

END IF;

END;

CREATE TRIGGER trg\_payments\_after\_insert

AFTER INSERT ON payments

FOR EACH ROW

BEGIN

DECLARE paid\_sum DECIMAL(10,2);

SELECT IFNULL(SUM(amount),0)

INTO paid\_sum

FROM payments

WHERE order\_id = NEW.order\_id;

UPDATE invoices inv

JOIN orders o ON o.id = NEW.order\_id

ON inv.order\_id = o.id

SET inv.is\_paid = (paid\_sum >= inv.total\_amount)

WHERE inv.order\_id = NEW.order\_id;

END;

### Расчёт итоговой суммы в счёте

Перед созданием счета (invoices) автоматически вычисляет total\_amount с учётом количества, цены книги и скидок.

CREATE TRIGGER trg\_invoices\_before\_insert

BEFORE INSERT ON invoices

FOR EACH ROW

BEGIN

DECLARE v\_total DECIMAL(10,2);

SELECT IFNULL(SUM(

oi.quantity

\* b.price

\* (1 - IFNULL(d.percentage,0)/100)

),0)

INTO v\_total

FROM order\_items oi

JOIN books b ON oi.book\_id = b.id

LEFT JOIN order\_discounts od ON od.book\_id = oi.book\_id

LEFT JOIN discounts d ON d.id = od.discount\_id

WHERE oi.order\_id = NEW.order\_id;

SET NEW.total\_amount = v\_total;

END;

## Представления

Представление – это виртуальная таблица, основанная на результате выполнения SQL-запроса. Оно упрощает доступ к данным, скрывая сложные запросы, улучшает читаемость кода и может способствовать повышению производительности [5].

### Популярные книги по количеству заказов

Отображает список книг, отсортированных по убыванию количества заказов. Включает идентификатор книги, название и общее число заказов. Количество заказов рассчитывается по числу записей в order\_items, связанных с каждой книгой.

CREATE OR REPLACE VIEW popular\_books AS

SELECT

b.id,

b.title,

COUNT(oi.book\_id) AS total\_orders

FROM books b

JOIN order\_items oi ON b.id = oi.book\_id

GROUP BY b.id, b.title

ORDER BY total\_orders DESC;

### Детали заказов пользователей

Представление объединяет информацию о заказах: идентификатор заказа, email пользователя, наименование книги, количество экземпляров, дату заказа и статус. Полезно для анализа заказов на уровне пользователей.

CREATE OR REPLACE VIEW user\_order\_details AS

SELECT

o.id AS order\_id,

u.email,

b.title,

oi.quantity,

o.order\_date,

osh.status\_id

FROM orders o

JOIN users u ON o.user\_id = u.id

JOIN order\_items oi ON o.id = oi.order\_id

JOIN books b ON oi.book\_id = b.id

JOIN order\_status\_history osh ON o.id = osh.order\_id;

### Полная информация о книге

Объединяет данные о книге, её издателе, авторах и жанрах. Предназначено для использования в каталоге и карточке товара. Авторы и жанры выводятся в виде строк, разделённых запятыми.

CREATE OR REPLACE VIEW book\_full\_info AS

SELECT

b.id,

b.title,

b.description,

b.price,

b.rating,

p.name AS publisher,

GROUP\_CONCAT(DISTINCT a.name SEPARATOR ', ') AS authors,

GROUP\_CONCAT(DISTINCT g.name SEPARATOR ', ') AS genres

FROM books b

LEFT JOIN publishers p ON b.publisher\_id = p.id

LEFT JOIN book\_authors ba ON b.id = ba.book\_id

LEFT JOIN authors a ON ba.author\_id = a.id

LEFT JOIN book\_genres bg ON b.id = bg.book\_id

LEFT JOIN genres g ON bg.genre\_id = g.id

GROUP BY b.id;

### Количество проданных экземпляров по авторам

Отчёт по авторам, ранжированный по количеству проданных экземпляров их книг. Содержит идентификатор автора, имя и общее число продаж.

CREATE OR REPLACE VIEW author\_sales AS

SELECT

a.id AS author\_id,

a.name AS author\_name,

COUNT(oi.book\_id) AS total\_sold

FROM authors a

JOIN book\_authors ba ON a.id = ba.author\_id

JOIN order\_items oi ON ba.book\_id = oi.book\_id

GROUP BY a.id, a.name

ORDER BY total\_sold DESC;

### Остатки книг по складам

Позволяет отслеживать количество каждой книги на всех складах. Показывает склад, книгу и её остаток. Если книга отсутствует на складе — количество будет 0.

CREATE OR REPLACE VIEW warehouse\_inventory AS

SELECT

w.id AS warehouse\_id,

w.address\_id,

b.id AS book\_id,

b.title AS book\_title,

IFNULL(i.quantity, 0) AS quantity

FROM warehouses w

CROSS JOIN books b

LEFT JOIN inventory i ON i.warehouse\_id = w.id AND i.book\_id = b.id

ORDER BY w.id, b.title;

### Активность пользователей (число заказов и общая сумма)

Отражает количество заказов и общую сумму, потраченную каждым пользователем. Используется для оценки клиентской ценности.

CREATE OR REPLACE VIEW user\_activity AS

SELECT

u.id AS user\_id,

u.email AS user\_email,

COUNT(o.id) AS orders\_count,

ROUND(IFNULL(SUM(oi.quantity \* b.price), 2), 2) AS total\_spent

FROM users u

LEFT JOIN orders o ON u.id = o.user\_id

LEFT JOIN order\_items oi ON o.id = oi.order\_id

LEFT JOIN books b ON oi.book\_id = b.id

GROUP BY u.id, u.email

ORDER BY total\_spent DESC;

### Средний рейтинг и число отзывов по книгам

Показывает агрегированную статистику отзывов: средний рейтинг и количество отзывов для каждой книги. Используется для сортировки и отбора книг по популярности и качеству.

CREATE OR REPLACE VIEW book\_reviews\_summary AS

SELECT

b.id AS book\_id,

b.title AS book\_title,

ROUND(AVG(r.rating\_value), 1) AS avg\_rating,

COUNT(r.id) AS reviews\_count

FROM books b

LEFT JOIN ratings r ON b.id = r.book\_id

GROUP BY b.id, b.title

ORDER BY avg\_rating DESC, reviews\_count DESC;

### Продажи по месяцам (количество заказов и выручка)

Аналитическое представление по продажам в разрезе месяцев. Учитывает общее количество заказов и суммарную выручку.

CREATE OR REPLACE VIEW monthly\_sales AS

SELECT

DATE\_FORMAT(o.order\_date, '%Y-%m') AS month,

COUNT(DISTINCT o.id) AS orders\_count,

ROUND(SUM(oi.quantity \* b.price), 2) AS revenue

FROM orders o

JOIN order\_items oi ON o.id = oi.order\_id

JOIN books b ON oi.book\_id = b.id

GROUP BY month

ORDER BY month DESC;

# Тестирование базы данных

Тестирование базы данных — ключевой этап разработки, в ходе которого выявляются ошибки, проверяется корректность работы триггеров, хранимых процедур, функций и представлений, а также подтверждается соответствие системы требованиям и ожиданиям пользователей.

Таблица 5.1 – Тест-кейсы

|  |  |  |
| --- | --- | --- |
| Действия | Ожидаемый результат | Результат |
| Вставка корректной записи книги в таблицу books с существующим publisher\_id и language\_id | Запись добавлена. Установлено значение created\_at | Запись добавлена |
| Вставка книги с несуществующим language\_id | Ошибка нарушения внешнего ключа. Запись не добавляется | Ошибка сгенерирована, запись отклонена |
| Вставка пользователя с уникальным username и email | Запись добавлена | Запись добавлена |
| Вставка пользователя с уже существующим email | Нарушение уникальности. Запись не добавляется | Ошибка сгенерирована |
| Добавление записи в cart\_items с существующим cart\_id и book\_id | Запись добавлена | Запись добавлена |
| Добавление записи в cart\_items с несуществующим book\_id | Ошибка нарушения внешнего ключа. Запись не добавляется | Ошибка сгенерирована |
| Добавление оценки в ratings с рейтингом 5 от пользователя к книге | Запись добавлена. Обновлён books.rating | Запись добавлена, рейтинг обновлён |
| Добавление оценки в ratings с рейтингом 0 | Нарушение ограничения CHECK. Запись не добавляется | Ошибка сгенерирована |
| Добавление книги в список желаемого через wishlist\_items | Запись добавлена. Уникальность пары (wishlist\_id, book\_id) сохранена | Запись добавлена |

Продолжение таблицы 5.1.

|  |  |  |
| --- | --- | --- |
| Повторное добавление той же книги в тот же wishlist | Нарушение ограничения уникальности. Запись не добавляется | Ошибка сгенерирована |
| Добавление заказа с существующими user\_id и address\_id | Запись добавлена. Установлены дата и статус | Запись добавлена |
| Вызов процедуры place\_order(userId) при наличии книг в корзине | Создаётся заказ и связанные записи. Корзина очищается | Заказ успешно создан |
| Вызов place\_order(userId) при пустой корзине | Ошибка или ничего не происходит. В зависимости от реализации | Корректная обработка: заказ не создаётся |
| Вызов функции get\_book\_stock(bookId) для существующей книги | Возвращается общее количество книг на складах | Возвращаемое значение корректно |
| Вызов update\_book\_rating(bookId) после изменения оценок | Средний рейтинг в books.rating обновляется | Рейтинг успешно пересчитан |
| Попытка удаления книги, которая используется в order\_items | Удаление запрещено, если нет ON DELETE CASCADE | Ошибка внешнего ключа |
| Вставка записи в inventory с book\_id, warehouse\_id, количеством > 0 | Запись добавлена | Запись добавлена |
| Попытка добавления склада с location\_id, которого нет в locations | Нарушение внешнего ключа. Запись не добавляется | Ошибка сгенерирована |
| Вызов функции get\_author\_book\_count(authorId) для автора с книгами | Возвращается количество книг этого автора | Функция возвращает корректное значение |
| Вызов функции get\_author\_book\_count(authorId) для автора без книг | Возвращается 0 | Значение корректно |
| Чтение представления TopSellingBooks | Возвращается список книг с наибольшими продажами, отсортированный по количеству | Представление возвращает ожидаемые данные |

Продолжение таблицы 5.1.

|  |  |  |
| --- | --- | --- |
| Чтение представления LowStockBooks | Возвращаются книги, у которых остаток на складах < 5 | Представление возвращает ожидаемые данные |
| Удаление пользователя, имеющего orders и wishlist | Все связанные данные каскадно удаляются (если включён ON DELETE CASCADE) | Связанные данные удалены |
| Удаление записи из wishlists приводит к удалению записей из wishlist\_items | Каскадное удаление срабатывает | Записи в wishlist\_items удалены |

# Заключение

В рамках данной курсовой работы была успешно спроектирована и реализована реляционная база данных, предназначенная для поддержки работы современного онлайн-магазина книг. Разработка охватила все ключевые аспекты, обозначенные во введении: необходимость хранения структурированной информации о книгах, авторах, жанрах, издательствах, а также учетных данных пользователей, заказах, отзывах и взаимодействиях с системой.

Созданная схема включает более 40 взаимосвязанных таблиц, в которых реализованы ограничения целостности, каскадные действия и индексация, что обеспечивает высокую степень согласованности и надёжности данных. Разработанные хранимые процедуры и функции автоматизируют бизнес-логику системы — от оформления заказа до управления списками желаемого, — что делает систему не только устойчивой, но и функционально гибкой.

Представления, созданные в рамках проекта, позволяют получать агрегированные отчёты о продажах, популярности жанров и наличии товаров, облегчая аналитическую работу и принятие решений. Таким образом, база данных предоставляет прочный фундамент для реализации полнофункционального веб-приложения книжного магазина, способного удовлетворить как текущие, так и перспективные потребности пользователей и администраторов.

Достигнутые в ходе работы результаты демонстрируют, что разработанная база данных готова к практическому применению и может стать надёжной основой для масштабируемой информационной системы в сфере электронной коммерции.

Список использованной литературы

[1] Ozon [Электронный ресурс]. – Режим доступа: https://ozon.by/ – Дата доступа: 04.05.2025.

[2] Лабиринт [Электронный ресурс]. – Режим доступа: https://www.labirint.ru/ – Дата доступа: 04.05.2025.

[3] Куликов Святослав Святославович «Работа с MySQL, MSSQL Server и Oracle» (2021) – 602 с.

[4] Куликов Святослав Святославович «Реляционные базы данных в примерах» (2021) – 422 с.

[5] Линн Бейли «Изучаем SQL» (2012) – 592 с.

[6] MySQL Documentation [Электронный ресурс]. – Режим доступа <https://dev.mysql.com/doc/>.

[7] SQL tutorials [Электронный ресурс]. – Режим доступа https://www.w3schools.com/sql/.

[8] Сильбершатц, А., Корт, Х. Ф., Сударшан, С. «Database System Concepts», 7-е изд. (2019) – 976 с.

[9] Бурцев, В. М., Костин, А. Ю. «Реляционные базы данных. Теория и практика» (2017) – 450 с

[10] Радков, А. В. «Основы реляционных баз данных» (2018) – 350 с.

[11] O'Reilly, C. «Learning SQL» (2005) – 400 с.

Приложение А

(обязательное)

-- Server version 9.3.0

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8mb4 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Current Database: `bookstore`

--

CREATE DATABASE /\*!32312 IF NOT EXISTS\*/ `bookstore` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;

USE `bookstore`;

--

-- Table structure for table `acquisition\_statuses`

--

DROP TABLE IF EXISTS `acquisition\_statuses`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `acquisition\_statuses` (

`id` int NOT NULL AUTO\_INCREMENT,

`status\_name` varchar(50) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `status\_name` (`status\_name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `acquisition\_statuses`

--

LOCK TABLES `acquisition\_statuses` WRITE;

/\*!40000 ALTER TABLE `acquisition\_statuses` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `acquisition\_statuses` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `addresses`

--

DROP TABLE IF EXISTS `addresses`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `addresses` (

`id` int NOT NULL AUTO\_INCREMENT,

`country` varchar(100) NOT NULL,

`admin\_area` varchar(100) NOT NULL,

`sub\_admin\_area` varchar(100) NOT NULL,

`locality` varchar(100) NOT NULL,

`sub\_locality` varchar(100) NOT NULL,

`thoroughfare` varchar(100) NOT NULL,

`sub\_thoroughfare` varchar(100) NOT NULL,

`postal\_code` varchar(20) NOT NULL,

PRIMARY KEY (`id`),

KEY `idx\_addresses\_postal\_code` (`postal\_code`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `addresses`

--

LOCK TABLES `addresses` WRITE;

/\*!40000 ALTER TABLE `addresses` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `addresses` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `author\_sales`

--

DROP TABLE IF EXISTS `author\_sales`;

/\*!50001 DROP VIEW IF EXISTS `author\_sales`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `author\_sales` AS SELECT

1 AS `author\_id`,

1 AS `author\_name`,

1 AS `total\_sold`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `authors`

--

DROP TABLE IF EXISTS `authors`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `authors` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

`birth\_date` date DEFAULT NULL,

`death\_date` date DEFAULT NULL,

`nationality\_id` int DEFAULT NULL,

`biography` text,

`website` varchar(255) DEFAULT NULL,

`photo\_url` varchar(255) DEFAULT NULL,

`gender\_id` int DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `nationality\_id` (`nationality\_id`),

KEY `gender\_id` (`gender\_id`),

CONSTRAINT `authors\_ibfk\_1` FOREIGN KEY (`nationality\_id`) REFERENCES `nationalities` (`id`) ON DELETE SET NULL,

CONSTRAINT `authors\_ibfk\_2` FOREIGN KEY (`gender\_id`) REFERENCES `genders` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `authors`

--

LOCK TABLES `authors` WRITE;

/\*!40000 ALTER TABLE `authors` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `authors` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_authors\_before\_insert` BEFORE INSERT ON `authors` FOR EACH ROW BEGIN

IF NEW.birth\_date IS NOT NULL AND NEW.death\_date IS NOT NULL AND NEW.birth\_date > NEW.death\_date THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Дата рождения не может быть позже даты смерти.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_authors\_before\_update` BEFORE UPDATE ON `authors` FOR EACH ROW BEGIN

IF NEW.birth\_date IS NOT NULL AND NEW.death\_date IS NOT NULL AND NEW.birth\_date > NEW.death\_date THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Дата рождения не может быть позже даты смерти.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `book\_acquisitions`

--

DROP TABLE IF EXISTS `book\_acquisitions`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `book\_acquisitions` (

`id` int NOT NULL AUTO\_INCREMENT,

`supplier\_id` int NOT NULL,

`book\_id` int NOT NULL,

`purchase\_price` decimal(10,2) DEFAULT NULL,

`quantity` int NOT NULL,

`acquisition\_date` datetime DEFAULT CURRENT\_TIMESTAMP,

`status\_id` int DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `supplier\_id` (`supplier\_id`),

KEY `book\_id` (`book\_id`),

KEY `status\_id` (`status\_id`),

CONSTRAINT `book\_acquisitions\_ibfk\_1` FOREIGN KEY (`supplier\_id`) REFERENCES `suppliers` (`id`) ON DELETE RESTRICT,

CONSTRAINT `book\_acquisitions\_ibfk\_2` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE,

CONSTRAINT `book\_acquisitions\_ibfk\_3` FOREIGN KEY (`status\_id`) REFERENCES `acquisition\_statuses` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `book\_acquisitions`

--

LOCK TABLES `book\_acquisitions` WRITE;

/\*!40000 ALTER TABLE `book\_acquisitions` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `book\_acquisitions` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_acquisitions\_after\_insert` AFTER INSERT ON `book\_acquisitions` FOR EACH ROW BEGIN

-- Автоматически увеличиваем остаток товаров при поступлении

INSERT INTO inventory(book\_id, warehouse\_id, quantity)

VALUES(NEW.book\_id, (SELECT id FROM warehouses LIMIT 1), NEW.quantity)

ON DUPLICATE KEY UPDATE quantity = quantity + NEW.quantity;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `book\_authors`

--

DROP TABLE IF EXISTS `book\_authors`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `book\_authors` (

`book\_id` int NOT NULL,

`author\_id` int NOT NULL,

PRIMARY KEY (`book\_id`,`author\_id`),

KEY `idx\_book\_authors\_author\_id` (`author\_id`),

CONSTRAINT `book\_authors\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE,

CONSTRAINT `book\_authors\_ibfk\_2` FOREIGN KEY (`author\_id`) REFERENCES `authors` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `book\_authors`

--

LOCK TABLES `book\_authors` WRITE;

/\*!40000 ALTER TABLE `book\_authors` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `book\_authors` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `book\_full\_info`

--

DROP TABLE IF EXISTS `book\_full\_info`;

/\*!50001 DROP VIEW IF EXISTS `book\_full\_info`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `book\_full\_info` AS SELECT

1 AS `id`,

1 AS `title`,

1 AS `description`,

1 AS `price`,

1 AS `rating`,

1 AS `publisher`,

1 AS `authors`,

1 AS `genres`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `book\_genres`

--

DROP TABLE IF EXISTS `book\_genres`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `book\_genres` (

`book\_id` int NOT NULL,

`genre\_id` int NOT NULL,

PRIMARY KEY (`book\_id`,`genre\_id`),

KEY `idx\_book\_genres\_genre\_id` (`genre\_id`),

CONSTRAINT `book\_genres\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE,

CONSTRAINT `book\_genres\_ibfk\_2` FOREIGN KEY (`genre\_id`) REFERENCES `genres` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `book\_genres`

--

LOCK TABLES `book\_genres` WRITE;

/\*!40000 ALTER TABLE `book\_genres` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `book\_genres` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `book\_languages`

--

DROP TABLE IF EXISTS `book\_languages`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `book\_languages` (

`book\_id` int NOT NULL,

`language\_id` int NOT NULL,

PRIMARY KEY (`book\_id`,`language\_id`),

KEY `idx\_book\_languages\_language\_id` (`language\_id`),

CONSTRAINT `book\_languages\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE,

CONSTRAINT `book\_languages\_ibfk\_2` FOREIGN KEY (`language\_id`) REFERENCES `languages` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `book\_languages`

--

LOCK TABLES `book\_languages` WRITE;

/\*!40000 ALTER TABLE `book\_languages` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `book\_languages` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `book\_reviews\_summary`

--

DROP TABLE IF EXISTS `book\_reviews\_summary`;

/\*!50001 DROP VIEW IF EXISTS `book\_reviews\_summary`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `book\_reviews\_summary` AS SELECT

1 AS `book\_id`,

1 AS `book\_title`,

1 AS `avg\_rating`,

1 AS `reviews\_count`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `book\_series`

--

DROP TABLE IF EXISTS `book\_series`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `book\_series` (

`book\_id` int NOT NULL,

`series\_id` int NOT NULL,

`volume\_number` int DEFAULT NULL,

PRIMARY KEY (`book\_id`,`series\_id`),

KEY `idx\_book\_series\_series\_id` (`series\_id`),

CONSTRAINT `book\_series\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE,

CONSTRAINT `book\_series\_ibfk\_2` FOREIGN KEY (`series\_id`) REFERENCES `series` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `book\_series`

--

LOCK TABLES `book\_series` WRITE;

/\*!40000 ALTER TABLE `book\_series` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `book\_series` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `book\_tag\_assignments`

--

DROP TABLE IF EXISTS `book\_tag\_assignments`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `book\_tag\_assignments` (

`book\_id` int NOT NULL,

`tag\_id` int NOT NULL,

PRIMARY KEY (`book\_id`,`tag\_id`),

KEY `idx\_book\_tag\_assignments\_tag\_id` (`tag\_id`),

CONSTRAINT `book\_tag\_assignments\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE,

CONSTRAINT `book\_tag\_assignments\_ibfk\_2` FOREIGN KEY (`tag\_id`) REFERENCES `book\_tags` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `book\_tag\_assignments`

--

LOCK TABLES `book\_tag\_assignments` WRITE;

/\*!40000 ALTER TABLE `book\_tag\_assignments` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `book\_tag\_assignments` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `book\_tags`

--

DROP TABLE IF EXISTS `book\_tags`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `book\_tags` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `book\_tags`

--

LOCK TABLES `book\_tags` WRITE;

/\*!40000 ALTER TABLE `book\_tags` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `book\_tags` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `books`

--

DROP TABLE IF EXISTS `books`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `books` (

`id` int NOT NULL AUTO\_INCREMENT,

`title` varchar(255) NOT NULL,

`description` text,

`publication\_year` int DEFAULT NULL,

`price` decimal(10,2) NOT NULL,

`isbn` varchar(20) DEFAULT NULL,

`publisher\_id` int DEFAULT NULL,

`rating` decimal(2,1) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `isbn` (`isbn`),

KEY `idx\_books\_publisher\_id` (`publisher\_id`),

KEY `idx\_books\_publication\_year` (`publication\_year`),

KEY `idx\_books\_price` (`price`),

CONSTRAINT `books\_ibfk\_1` FOREIGN KEY (`publisher\_id`) REFERENCES `publishers` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `books`

--

LOCK TABLES `books` WRITE;

/\*!40000 ALTER TABLE `books` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `books` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_books\_before\_insert` BEFORE INSERT ON `books` FOR EACH ROW BEGIN

-- Цена должна быть неотрицательной

IF NEW.price < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Цена книги не может быть отрицательной.';

END IF;

-- Год публикации не больше текущего

IF NEW.publication\_year IS NOT NULL AND NEW.publication\_year > YEAR(CURDATE()) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Год публикации не может быть в будущем.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_books\_before\_update` BEFORE UPDATE ON `books` FOR EACH ROW BEGIN

IF NEW.price < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Цена книги не может быть отрицательной.';

END IF;

IF NEW.publication\_year IS NOT NULL AND NEW.publication\_year > YEAR(CURDATE()) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Год публикации не может быть в будущем.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `cart\_items`

--

DROP TABLE IF EXISTS `cart\_items`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `cart\_items` (

`id` int NOT NULL AUTO\_INCREMENT,

`cart\_id` int DEFAULT NULL,

`book\_id` int DEFAULT NULL,

`quantity` int NOT NULL,

PRIMARY KEY (`id`),

KEY `cart\_id` (`cart\_id`),

KEY `book\_id` (`book\_id`),

CONSTRAINT `cart\_items\_ibfk\_1` FOREIGN KEY (`cart\_id`) REFERENCES `shopping\_carts` (`id`) ON DELETE CASCADE,

CONSTRAINT `cart\_items\_ibfk\_2` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `cart\_items`

--

LOCK TABLES `cart\_items` WRITE;

/\*!40000 ALTER TABLE `cart\_items` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `cart\_items` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_update\_cart\_item` BEFORE INSERT ON `cart\_items` FOR EACH ROW BEGIN

DECLARE existing\_quantity INT;

SELECT quantity INTO existing\_quantity

FROM cart\_items

WHERE cart\_id = NEW.cart\_id AND book\_id = NEW.book\_id

LIMIT 1;

IF existing\_quantity IS NOT NULL THEN

UPDATE cart\_items

SET quantity = quantity + NEW.quantity

WHERE cart\_id = NEW.cart\_id AND book\_id = NEW.book\_id;

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Дубликат записи предотвращён. Количество обновлено вручную.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `contracts`

--

DROP TABLE IF EXISTS `contracts`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `contracts` (

`id` int NOT NULL AUTO\_INCREMENT,

`employee\_id` int NOT NULL,

`start\_date` date NOT NULL,

`end\_date` date NOT NULL,

`signed\_date` date DEFAULT NULL,

`salary` decimal(10,2) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `employee\_id` (`employee\_id`),

CONSTRAINT `contracts\_ibfk\_1` FOREIGN KEY (`employee\_id`) REFERENCES `employees` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `contracts`

--

LOCK TABLES `contracts` WRITE;

/\*!40000 ALTER TABLE `contracts` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `contracts` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_contracts\_before\_insert` BEFORE INSERT ON `contracts` FOR EACH ROW BEGIN

IF NEW.start\_date >= NEW.end\_date THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Дата начала контракта должна быть раньше даты окончания.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_contracts\_before\_update` BEFORE UPDATE ON `contracts` FOR EACH ROW BEGIN

IF NEW.start\_date >= NEW.end\_date THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Дата начала контракта должна быть раньше даты окончания.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `discounts`

--

DROP TABLE IF EXISTS `discounts`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `discounts` (

`id` int NOT NULL AUTO\_INCREMENT,

`code` varchar(50) DEFAULT NULL,

`percentage` decimal(5,2) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `code` (`code`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `discounts`

--

LOCK TABLES `discounts` WRITE;

/\*!40000 ALTER TABLE `discounts` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `discounts` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_discounts\_before\_insert` BEFORE INSERT ON `discounts` FOR EACH ROW BEGIN

IF NEW.percentage < 0 OR NEW.percentage > 100 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Процент скидки должен быть от 0 до 100.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_discounts\_before\_update` BEFORE UPDATE ON `discounts` FOR EACH ROW BEGIN

IF NEW.percentage < 0 OR NEW.percentage > 100 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Процент скидки должен быть от 0 до 100.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `employee\_info`

--

DROP TABLE IF EXISTS `employee\_info`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `employee\_info` (

`employee\_id` int NOT NULL,

`first\_name` varchar(100) NOT NULL,

`last\_name` varchar(100) NOT NULL,

`middle\_name` varchar(100) DEFAULT NULL,

`phone` varchar(20) DEFAULT NULL,

`date\_of\_birth` date DEFAULT NULL,

`gender\_id` int DEFAULT NULL,

PRIMARY KEY (`employee\_id`),

KEY `idx\_employee\_info\_gender\_id` (`gender\_id`),

CONSTRAINT `employee\_info\_ibfk\_1` FOREIGN KEY (`employee\_id`) REFERENCES `employees` (`id`) ON DELETE CASCADE,

CONSTRAINT `employee\_info\_ibfk\_2` FOREIGN KEY (`gender\_id`) REFERENCES `genders` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `employee\_info`

--

LOCK TABLES `employee\_info` WRITE;

/\*!40000 ALTER TABLE `employee\_info` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `employee\_info` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `employee\_positions`

--

DROP TABLE IF EXISTS `employee\_positions`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `employee\_positions` (

`employee\_id` int NOT NULL,

`position\_id` int NOT NULL,

`assigned\_at` datetime DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`employee\_id`,`position\_id`),

KEY `idx\_employee\_positions\_position\_id` (`position\_id`),

CONSTRAINT `employee\_positions\_ibfk\_1` FOREIGN KEY (`employee\_id`) REFERENCES `employees` (`id`) ON DELETE CASCADE,

CONSTRAINT `employee\_positions\_ibfk\_2` FOREIGN KEY (`position\_id`) REFERENCES `positions` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `employee\_positions`

--

LOCK TABLES `employee\_positions` WRITE;

/\*!40000 ALTER TABLE `employee\_positions` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `employee\_positions` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `employees`

--

DROP TABLE IF EXISTS `employees`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `employees` (

`id` int NOT NULL AUTO\_INCREMENT,

`email` varchar(255) DEFAULT NULL,

`hire\_date` date NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `email` (`email`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `employees`

--

LOCK TABLES `employees` WRITE;

/\*!40000 ALTER TABLE `employees` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `employees` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `genders`

--

DROP TABLE IF EXISTS `genders`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `genders` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(50) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `genders`

--

LOCK TABLES `genders` WRITE;

/\*!40000 ALTER TABLE `genders` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `genders` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `genres`

--

DROP TABLE IF EXISTS `genres`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `genres` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `genres`

--

LOCK TABLES `genres` WRITE;

/\*!40000 ALTER TABLE `genres` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `genres` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `inventory`

--

DROP TABLE IF EXISTS `inventory`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `inventory` (

`id` int NOT NULL AUTO\_INCREMENT,

`book\_id` int DEFAULT NULL,

`warehouse\_id` int DEFAULT NULL,

`quantity` int DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `book\_id` (`book\_id`),

KEY `warehouse\_id` (`warehouse\_id`),

CONSTRAINT `inventory\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE,

CONSTRAINT `inventory\_ibfk\_2` FOREIGN KEY (`warehouse\_id`) REFERENCES `warehouses` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `inventory`

--

LOCK TABLES `inventory` WRITE;

/\*!40000 ALTER TABLE `inventory` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `inventory` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_inventory\_before\_insert` BEFORE INSERT ON `inventory` FOR EACH ROW BEGIN

IF NEW.quantity < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Количество на складе не может быть отрицательным.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_inventory\_before\_update` BEFORE UPDATE ON `inventory` FOR EACH ROW BEGIN

IF NEW.quantity < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Количество на складе не может быть отрицательным.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `invoices`

--

DROP TABLE IF EXISTS `invoices`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `invoices` (

`id` int NOT NULL AUTO\_INCREMENT,

`order\_id` int NOT NULL,

`invoice\_number` varchar(50) DEFAULT NULL,

`issued\_date` datetime DEFAULT CURRENT\_TIMESTAMP,

`total\_amount` decimal(10,2) DEFAULT NULL,

`is\_paid` tinyint(1) DEFAULT '0',

PRIMARY KEY (`id`),

UNIQUE KEY `invoice\_number` (`invoice\_number`),

KEY `idx\_invoices\_order\_id` (`order\_id`),

CONSTRAINT `invoices\_ibfk\_1` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `invoices`

--

LOCK TABLES `invoices` WRITE;

/\*!40000 ALTER TABLE `invoices` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `invoices` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_invoices\_before\_insert` BEFORE INSERT ON `invoices` FOR EACH ROW BEGIN

DECLARE v\_total DECIMAL(10,2);

SELECT IFNULL(SUM(

oi.quantity

\* b.price

\* (1 - IFNULL(d.percentage, 0) / 100)

), 0)

INTO v\_total

FROM order\_items oi

JOIN books b

ON oi.book\_id = b.id

LEFT JOIN order\_discounts od

ON od.book\_id = oi.book\_id

AND od.book\_id = oi.book\_id

LEFT JOIN discounts d

ON d.id = od.discount\_id

WHERE oi.order\_id = NEW.order\_id;

SET NEW.total\_amount = v\_total;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `languages`

--

DROP TABLE IF EXISTS `languages`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `languages` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

`code` varchar(3) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`),

UNIQUE KEY `code` (`code`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `languages`

--

LOCK TABLES `languages` WRITE;

/\*!40000 ALTER TABLE `languages` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `languages` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `monthly\_sales`

--

DROP TABLE IF EXISTS `monthly\_sales`;

/\*!50001 DROP VIEW IF EXISTS `monthly\_sales`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `monthly\_sales` AS SELECT

1 AS `month`,

1 AS `orders\_count`,

1 AS `revenue`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `nationalities`

--

DROP TABLE IF EXISTS `nationalities`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `nationalities` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `nationalities`

--

LOCK TABLES `nationalities` WRITE;

/\*!40000 ALTER TABLE `nationalities` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `nationalities` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `order\_discounts`

--

DROP TABLE IF EXISTS `order\_discounts`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `order\_discounts` (

`book\_id` int NOT NULL,

`discount\_id` int NOT NULL,

PRIMARY KEY (`book\_id`,`discount\_id`),

KEY `idx\_order\_discounts\_discount\_id` (`discount\_id`),

CONSTRAINT `order\_discounts\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE,

CONSTRAINT `order\_discounts\_ibfk\_2` FOREIGN KEY (`discount\_id`) REFERENCES `discounts` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `order\_discounts`

--

LOCK TABLES `order\_discounts` WRITE;

/\*!40000 ALTER TABLE `order\_discounts` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `order\_discounts` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `order\_items`

--

DROP TABLE IF EXISTS `order\_items`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `order\_items` (

`id` int NOT NULL AUTO\_INCREMENT,

`order\_id` int DEFAULT NULL,

`book\_id` int DEFAULT NULL,

`quantity` int DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `idx\_order\_items\_order\_id` (`order\_id`),

KEY `idx\_order\_items\_book\_id` (`book\_id`),

CONSTRAINT `order\_items\_ibfk\_1` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE,

CONSTRAINT `order\_items\_ibfk\_2` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `order\_items`

--

LOCK TABLES `order\_items` WRITE;

/\*!40000 ALTER TABLE `order\_items` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `order\_items` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_order\_items\_before\_insert` BEFORE INSERT ON `order\_items` FOR EACH ROW BEGIN

IF NEW.quantity <= 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Количество в заказе должно быть положительным.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_order\_items\_before\_update` BEFORE UPDATE ON `order\_items` FOR EACH ROW BEGIN

IF NEW.quantity <= 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Количество в заказе должно быть положительным.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `order\_status\_history`

--

DROP TABLE IF EXISTS `order\_status\_history`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `order\_status\_history` (

`id` int NOT NULL AUTO\_INCREMENT,

`order\_id` int NOT NULL,

`status\_id` int NOT NULL,

`changed\_at` datetime DEFAULT CURRENT\_TIMESTAMP,

`comment` text,

PRIMARY KEY (`id`),

KEY `idx\_order\_status\_history\_order\_id` (`order\_id`),

KEY `idx\_order\_status\_history\_status\_id` (`status\_id`),

CONSTRAINT `order\_status\_history\_ibfk\_1` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE,

CONSTRAINT `order\_status\_history\_ibfk\_2` FOREIGN KEY (`status\_id`) REFERENCES `order\_statuses` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `order\_status\_history`

--

LOCK TABLES `order\_status\_history` WRITE;

/\*!40000 ALTER TABLE `order\_status\_history` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `order\_status\_history` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `order\_statuses`

--

DROP TABLE IF EXISTS `order\_statuses`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `order\_statuses` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(50) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `order\_statuses`

--

LOCK TABLES `order\_statuses` WRITE;

/\*!40000 ALTER TABLE `order\_statuses` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `order\_statuses` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `orders`

--

DROP TABLE IF EXISTS `orders`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `orders` (

`id` int NOT NULL AUTO\_INCREMENT,

`user\_id` int DEFAULT NULL,

`order\_date` datetime DEFAULT CURRENT\_TIMESTAMP,

`status` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `idx\_orders\_user\_id` (`user\_id`),

CONSTRAINT `orders\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `orders`

--

LOCK TABLES `orders` WRITE;

/\*!40000 ALTER TABLE `orders` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `orders` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_orders\_set\_default\_status` BEFORE INSERT ON `orders` FOR EACH ROW BEGIN

IF NEW.status IS NULL OR NEW.status = '' THEN

SET NEW.status = 'Создан';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `payment\_methods`

--

DROP TABLE IF EXISTS `payment\_methods`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `payment\_methods` (

`id` int NOT NULL AUTO\_INCREMENT,

`method\_name` varchar(100) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `payment\_methods`

--

LOCK TABLES `payment\_methods` WRITE;

/\*!40000 ALTER TABLE `payment\_methods` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `payment\_methods` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `payments`

--

DROP TABLE IF EXISTS `payments`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `payments` (

`id` int NOT NULL AUTO\_INCREMENT,

`order\_id` int DEFAULT NULL,

`amount` decimal(10,2) DEFAULT NULL,

`payment\_date` datetime DEFAULT NULL,

`payment\_method\_id` int DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `idx\_payments\_order\_id` (`order\_id`),

KEY `idx\_payments\_method\_id` (`payment\_method\_id`),

CONSTRAINT `payments\_ibfk\_1` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE,

CONSTRAINT `payments\_ibfk\_2` FOREIGN KEY (`payment\_method\_id`) REFERENCES `payment\_methods` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `payments`

--

LOCK TABLES `payments` WRITE;

/\*!40000 ALTER TABLE `payments` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `payments` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_payments\_before\_insert` BEFORE INSERT ON `payments` FOR EACH ROW BEGIN

DECLARE inv\_total DECIMAL(10,2);

SELECT total\_amount INTO inv\_total

FROM invoices

WHERE order\_id = NEW.order\_id

AND is\_paid = FALSE

LIMIT 1;

IF NEW.amount > inv\_total THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Сумма платежа превышает сумму по счёту.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_payments\_after\_insert` AFTER INSERT ON `payments` FOR EACH ROW BEGIN

DECLARE paid\_sum DECIMAL(10,2);

SELECT IFNULL(SUM(amount), 0) INTO paid\_sum

FROM payments

WHERE order\_id = NEW.order\_id;

UPDATE invoices i

JOIN orders o ON o.id = NEW.order\_id

JOIN invoices inv ON inv.order\_id = o.id

SET inv.is\_paid = (paid\_sum >= inv.total\_amount)

WHERE inv.order\_id = NEW.order\_id;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Temporary view structure for view `popular\_books`

--

DROP TABLE IF EXISTS `popular\_books`;

/\*!50001 DROP VIEW IF EXISTS `popular\_books`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `popular\_books` AS SELECT

1 AS `id`,

1 AS `title`,

1 AS `total\_orders`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `positions`

--

DROP TABLE IF EXISTS `positions`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `positions` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `positions`

--

LOCK TABLES `positions` WRITE;

/\*!40000 ALTER TABLE `positions` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `positions` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `publishers`

--

DROP TABLE IF EXISTS `publishers`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `publishers` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

`founded\_year` int DEFAULT NULL,

`website` varchar(255) DEFAULT NULL,

`contact\_email` varchar(255) DEFAULT NULL,

`phone` varchar(20) DEFAULT NULL,

`address\_id` int NOT NULL,

PRIMARY KEY (`id`),

KEY `idx\_publishers\_address\_id` (`address\_id`),

CONSTRAINT `publishers\_ibfk\_1` FOREIGN KEY (`address\_id`) REFERENCES `addresses` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `publishers`

--

LOCK TABLES `publishers` WRITE;

/\*!40000 ALTER TABLE `publishers` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `publishers` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `ratings`

--

DROP TABLE IF EXISTS `ratings`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `ratings` (

`id` int NOT NULL AUTO\_INCREMENT,

`user\_id` int DEFAULT NULL,

`book\_id` int DEFAULT NULL,

`rating\_value` tinyint DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `user\_id` (`user\_id`),

KEY `book\_id` (`book\_id`),

CONSTRAINT `ratings\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE,

CONSTRAINT `ratings\_ibfk\_2` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `ratings`

--

LOCK TABLES `ratings` WRITE;

/\*!40000 ALTER TABLE `ratings` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `ratings` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_ratings\_before\_insert` BEFORE INSERT ON `ratings` FOR EACH ROW BEGIN

IF NEW.rating\_value < 1 OR NEW.rating\_value > 5 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Значение рейтинга должно быть от 1 до 5.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_after\_rating\_insert` AFTER INSERT ON `ratings` FOR EACH ROW BEGIN

CALL update\_book\_rating(NEW.book\_id);

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_ratings\_before\_update` BEFORE UPDATE ON `ratings` FOR EACH ROW BEGIN

IF NEW.rating\_value < 1 OR NEW.rating\_value > 5 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Значение рейтинга должно быть от 1 до 5.';

END IF;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_after\_rating\_update` AFTER UPDATE ON `ratings` FOR EACH ROW BEGIN

CALL update\_book\_rating(NEW.book\_id);

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`%`\*/ /\*!50003 TRIGGER `trg\_after\_rating\_delete` AFTER DELETE ON `ratings` FOR EACH ROW BEGIN

CALL update\_book\_rating(OLD.book\_id);

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `restock\_requests`

--

DROP TABLE IF EXISTS `restock\_requests`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `restock\_requests` (

`id` int NOT NULL AUTO\_INCREMENT,

`book\_id` int DEFAULT NULL,

`quantity` int DEFAULT NULL,

`request\_date` date DEFAULT NULL,

`status` varchar(50) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `book\_id` (`book\_id`),

CONSTRAINT `restock\_requests\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `restock\_requests`

--

LOCK TABLES `restock\_requests` WRITE;

/\*!40000 ALTER TABLE `restock\_requests` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `restock\_requests` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `review\_comments`

--

DROP TABLE IF EXISTS `review\_comments`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `review\_comments` (

`id` int NOT NULL AUTO\_INCREMENT,

`review\_id` int DEFAULT NULL,

`user\_id` int DEFAULT NULL,

`comment` text,

`commented\_at` datetime DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

KEY `review\_id` (`review\_id`),

KEY `user\_id` (`user\_id`),

CONSTRAINT `review\_comments\_ibfk\_1` FOREIGN KEY (`review\_id`) REFERENCES `reviews` (`id`) ON DELETE CASCADE,

CONSTRAINT `review\_comments\_ibfk\_2` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `review\_comments`

--

LOCK TABLES `review\_comments` WRITE;

/\*!40000 ALTER TABLE `review\_comments` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `review\_comments` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `reviews`

--

DROP TABLE IF EXISTS `reviews`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `reviews` (

`id` int NOT NULL AUTO\_INCREMENT,

`user\_id` int DEFAULT NULL,

`book\_id` int DEFAULT NULL,

`review\_text` text,

`review\_date` datetime DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

KEY `user\_id` (`user\_id`),

KEY `book\_id` (`book\_id`),

CONSTRAINT `reviews\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE,

CONSTRAINT `reviews\_ibfk\_2` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `reviews`

--

LOCK TABLES `reviews` WRITE;

/\*!40000 ALTER TABLE `reviews` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `reviews` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `series`

--

DROP TABLE IF EXISTS `series`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `series` (

`id` int NOT NULL AUTO\_INCREMENT,

`title` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `series`

--

LOCK TABLES `series` WRITE;

/\*!40000 ALTER TABLE `series` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `series` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `shopping\_carts`

--

DROP TABLE IF EXISTS `shopping\_carts`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `shopping\_carts` (

`id` int NOT NULL AUTO\_INCREMENT,

`user\_id` int DEFAULT NULL,

`updated\_at` datetime DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

KEY `user\_id` (`user\_id`),

CONSTRAINT `shopping\_carts\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `shopping\_carts`

--

LOCK TABLES `shopping\_carts` WRITE;

/\*!40000 ALTER TABLE `shopping\_carts` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `shopping\_carts` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `suppliers`

--

DROP TABLE IF EXISTS `suppliers`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `suppliers` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

`contact\_email` varchar(255) DEFAULT NULL,

`contact\_phone` varchar(20) DEFAULT NULL,

`address\_id` int NOT NULL,

PRIMARY KEY (`id`),

KEY `address\_id` (`address\_id`),

CONSTRAINT `suppliers\_ibfk\_1` FOREIGN KEY (`address\_id`) REFERENCES `addresses` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `suppliers`

--

LOCK TABLES `suppliers` WRITE;

/\*!40000 ALTER TABLE `suppliers` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `suppliers` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `user\_activity`

--

DROP TABLE IF EXISTS `user\_activity`;

/\*!50001 DROP VIEW IF EXISTS `user\_activity`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `user\_activity` AS SELECT

1 AS `user\_id`,

1 AS `user\_email`,

1 AS `orders\_count`,

1 AS `total\_spent`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `user\_info`

--

DROP TABLE IF EXISTS `user\_info`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user\_info` (

`user\_id` int NOT NULL,

`first\_name` varchar(100) NOT NULL,

`last\_name` varchar(100) NOT NULL,

`middle\_name` varchar(100) DEFAULT NULL,

`phone` varchar(20) DEFAULT NULL,

`date\_of\_birth` date DEFAULT NULL,

`gender\_id` int DEFAULT NULL,

PRIMARY KEY (`user\_id`),

KEY `idx\_user\_info\_gender\_id` (`gender\_id`),

CONSTRAINT `user\_info\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE,

CONSTRAINT `user\_info\_ibfk\_2` FOREIGN KEY (`gender\_id`) REFERENCES `genders` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `user\_info`

--

LOCK TABLES `user\_info` WRITE;

/\*!40000 ALTER TABLE `user\_info` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `user\_info` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `user\_order\_details`

--

DROP TABLE IF EXISTS `user\_order\_details`;

/\*!50001 DROP VIEW IF EXISTS `user\_order\_details`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `user\_order\_details` AS SELECT

1 AS `order\_id`,

1 AS `email`,

1 AS `title`,

1 AS `quantity`,

1 AS `order\_date`,

1 AS `status\_id`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `user\_preferences`

--

DROP TABLE IF EXISTS `user\_preferences`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user\_preferences` (

`user\_id` int NOT NULL,

`favorite\_genre\_id` int NOT NULL,

PRIMARY KEY (`user\_id`,`favorite\_genre\_id`),

KEY `idx\_user\_preferences\_genre\_id` (`favorite\_genre\_id`),

CONSTRAINT `user\_preferences\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE,

CONSTRAINT `user\_preferences\_ibfk\_2` FOREIGN KEY (`favorite\_genre\_id`) REFERENCES `genres` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `user\_preferences`

--

LOCK TABLES `user\_preferences` WRITE;

/\*!40000 ALTER TABLE `user\_preferences` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `user\_preferences` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `user\_role\_assignments`

--

DROP TABLE IF EXISTS `user\_role\_assignments`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user\_role\_assignments` (

`user\_id` int NOT NULL,

`role\_id` int NOT NULL,

PRIMARY KEY (`user\_id`,`role\_id`),

KEY `idx\_user\_role\_assignments\_role\_id` (`role\_id`),

CONSTRAINT `user\_role\_assignments\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE,

CONSTRAINT `user\_role\_assignments\_ibfk\_2` FOREIGN KEY (`role\_id`) REFERENCES `user\_roles` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `user\_role\_assignments`

--

LOCK TABLES `user\_role\_assignments` WRITE;

/\*!40000 ALTER TABLE `user\_role\_assignments` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `user\_role\_assignments` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `user\_roles`

--

DROP TABLE IF EXISTS `user\_roles`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user\_roles` (

`id` int NOT NULL AUTO\_INCREMENT,

`name` varchar(100) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `user\_roles`

--

LOCK TABLES `user\_roles` WRITE;

/\*!40000 ALTER TABLE `user\_roles` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `user\_roles` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `users`

--

DROP TABLE IF EXISTS `users`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `users` (

`id` int NOT NULL AUTO\_INCREMENT,

`email` varchar(255) NOT NULL,

`password\_hash` varchar(255) NOT NULL,

`created\_at` datetime DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

UNIQUE KEY `email` (`email`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `users`

--

LOCK TABLES `users` WRITE;

/\*!40000 ALTER TABLE `users` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `users` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Temporary view structure for view `warehouse\_inventory`

--

DROP TABLE IF EXISTS `warehouse\_inventory`;

/\*!50001 DROP VIEW IF EXISTS `warehouse\_inventory`\*/;

SET @saved\_cs\_client = @@character\_set\_client;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 CREATE VIEW `warehouse\_inventory` AS SELECT

1 AS `warehouse\_id`,

1 AS `address\_id`,

1 AS `book\_id`,

1 AS `book\_title`,

1 AS `quantity`\*/;

SET character\_set\_client = @saved\_cs\_client;

--

-- Table structure for table `warehouses`

--

DROP TABLE IF EXISTS `warehouses`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `warehouses` (

`id` int NOT NULL AUTO\_INCREMENT,

`address\_id` int NOT NULL,

PRIMARY KEY (`id`),

KEY `address\_id` (`address\_id`),

CONSTRAINT `warehouses\_ibfk\_1` FOREIGN KEY (`address\_id`) REFERENCES `addresses` (`id`) ON DELETE RESTRICT

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `warehouses`

--

LOCK TABLES `warehouses` WRITE;

/\*!40000 ALTER TABLE `warehouses` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `warehouses` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `wishlist\_items`

--

DROP TABLE IF EXISTS `wishlist\_items`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `wishlist\_items` (

`id` int NOT NULL AUTO\_INCREMENT,

`wishlist\_id` int DEFAULT NULL,

`book\_id` int DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `wishlist\_id` (`wishlist\_id`),

KEY `book\_id` (`book\_id`),

CONSTRAINT `wishlist\_items\_ibfk\_1` FOREIGN KEY (`wishlist\_id`) REFERENCES `wishlists` (`id`) ON DELETE CASCADE,

CONSTRAINT `wishlist\_items\_ibfk\_2` FOREIGN KEY (`book\_id`) REFERENCES `books` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `wishlist\_items`

--

LOCK TABLES `wishlist\_items` WRITE;

/\*!40000 ALTER TABLE `wishlist\_items` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `wishlist\_items` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `wishlists`

--

DROP TABLE IF EXISTS `wishlists`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `wishlists` (

`id` int NOT NULL AUTO\_INCREMENT,

`user\_id` int DEFAULT NULL,

`updated\_at` datetime DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

KEY `user\_id` (`user\_id`),

CONSTRAINT `wishlists\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `wishlists`

--

LOCK TABLES `wishlists` WRITE;

/\*!40000 ALTER TABLE `wishlists` DISABLE KEYS \*/;

/\*!40000 ALTER TABLE `wishlists` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Current Database: `bookstore`

--

USE `bookstore`;

--

-- Final view structure for view `author\_sales`

--

/\*!50001 DROP VIEW IF EXISTS `author\_sales`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`root`@`%` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `author\_sales` AS select `a`.`id` AS `author\_id`,`a`.`name` AS `author\_name`,count(`oi`.`book\_id`) AS `total\_sold` from ((`authors` `a` join `book\_authors` `ba` on((`a`.`id` = `ba`.`author\_id`))) join `order\_items` `oi` on((`ba`.`book\_id` = `oi`.`book\_id`))) group by `a`.`id`,`a`.`name` order by `total\_sold` desc \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `book\_full\_info`

--

/\*!50001 DROP VIEW IF EXISTS `book\_full\_info`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`root`@`%` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `book\_full\_info` AS select `b`.`id` AS `id`,`b`.`title` AS `title`,`b`.`description` AS `description`,`b`.`price` AS `price`,`b`.`rating` AS `rating`,`p`.`name` AS `publisher`,group\_concat(distinct `a`.`name` separator ', ') AS `authors`,group\_concat(distinct `g`.`name` separator ', ') AS `genres` from (((((`books` `b` left join `publishers` `p` on((`b`.`publisher\_id` = `p`.`id`))) left join `book\_authors` `ba` on((`b`.`id` = `ba`.`book\_id`))) left join `authors` `a` on((`ba`.`author\_id` = `a`.`id`))) left join `book\_genres` `bg` on((`b`.`id` = `bg`.`book\_id`))) left join `genres` `g` on((`bg`.`genre\_id` = `g`.`id`))) group by `b`.`id` \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `book\_reviews\_summary`

--

/\*!50001 DROP VIEW IF EXISTS `book\_reviews\_summary`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`root`@`%` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `book\_reviews\_summary` AS select `b`.`id` AS `book\_id`,`b`.`title` AS `book\_title`,round(avg(`r`.`rating\_value`),1) AS `avg\_rating`,count(`r`.`id`) AS `reviews\_count` from (`books` `b` left join `ratings` `r` on((`b`.`id` = `r`.`book\_id`))) group by `b`.`id`,`b`.`title` order by `avg\_rating` desc,`reviews\_count` desc \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `monthly\_sales`

--

/\*!50001 DROP VIEW IF EXISTS `monthly\_sales`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`root`@`%` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `monthly\_sales` AS select date\_format(`o`.`order\_date`,'%Y-%m') AS `month`,count(distinct `o`.`id`) AS `orders\_count`,round(sum((`oi`.`quantity` \* `b`.`price`)),2) AS `revenue` from ((`orders` `o` join `order\_items` `oi` on((`o`.`id` = `oi`.`order\_id`))) join `books` `b` on((`oi`.`book\_id` = `b`.`id`))) group by `month` order by `month` desc \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `popular\_books`

--

/\*!50001 DROP VIEW IF EXISTS `popular\_books`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`root`@`%` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `popular\_books` AS select `b`.`id` AS `id`,`b`.`title` AS `title`,count(`oi`.`book\_id`) AS `total\_orders` from (`books` `b` join `order\_items` `oi` on((`b`.`id` = `oi`.`book\_id`))) group by `b`.`id`,`b`.`title` order by `total\_orders` desc \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `user\_activity`

--

/\*!50001 DROP VIEW IF EXISTS `user\_activity`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`root`@`%` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `user\_activity` AS select `u`.`id` AS `user\_id`,`u`.`email` AS `user\_email`,count(`o`.`id`) AS `orders\_count`,round(ifnull(sum((`oi`.`quantity` \* `b`.`price`)),0),2) AS `total\_spent` from (((`users` `u` left join `orders` `o` on((`u`.`id` = `o`.`user\_id`))) left join `order\_items` `oi` on((`o`.`id` = `oi`.`order\_id`))) left join `books` `b` on((`oi`.`book\_id` = `b`.`id`))) group by `u`.`id`,`u`.`email` order by `total\_spent` desc \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `user\_order\_details`

--

/\*!50001 DROP VIEW IF EXISTS `user\_order\_details`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`root`@`%` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `user\_order\_details` AS select `o`.`id` AS `order\_id`,`u`.`email` AS `email`,`b`.`title` AS `title`,`oi`.`quantity` AS `quantity`,`o`.`order\_date` AS `order\_date`,`osh`.`status\_id` AS `status\_id` from ((((`orders` `o` join `users` `u` on((`o`.`user\_id` = `u`.`id`))) join `order\_items` `oi` on((`o`.`id` = `oi`.`order\_id`))) join `books` `b` on((`oi`.`book\_id` = `b`.`id`))) join `order\_status\_history` `osh` on((`o`.`id` = `osh`.`order\_id`))) \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

--

-- Final view structure for view `warehouse\_inventory`

--

/\*!50001 DROP VIEW IF EXISTS `warehouse\_inventory`\*/;

/\*!50001 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50001 SET @saved\_cs\_results = @@character\_set\_results \*/;

/\*!50001 SET @saved\_col\_connection = @@collation\_connection \*/;

/\*!50001 SET character\_set\_client = utf8mb4 \*/;

/\*!50001 SET character\_set\_results = utf8mb4 \*/;

/\*!50001 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/;

/\*!50001 CREATE ALGORITHM=UNDEFINED \*/

/\*!50013 DEFINER=`root`@`%` SQL SECURITY DEFINER \*/

/\*!50001 VIEW `warehouse\_inventory` AS select `w`.`id` AS `warehouse\_id`,`w`.`address\_id` AS `address\_id`,`b`.`id` AS `book\_id`,`b`.`title` AS `book\_title`,ifnull(`i`.`quantity`,0) AS `quantity` from ((`warehouses` `w` join `books` `b`) left join `inventory` `i` on(((`i`.`warehouse\_id` = `w`.`id`) and (`i`.`book\_id` = `b`.`id`)))) order by `w`.`id`,`b`.`title` \*/;

/\*!50001 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50001 SET character\_set\_results = @saved\_cs\_results \*/;

/\*!50001 SET collation\_connection = @saved\_col\_connection \*/;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2025-05-09 17:33:51

ВЕДОМОСТЬ ДОКУМЕНТОВ

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | | Наименование | | | | Дополнительные сведения | |
|  | | | | | Текстовые документы | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР КП 1–40 01 01 020 ПЗ | | | | | Пояснительная записка | | | | 70 с. | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | Графические документы | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.251003-020 СА | | | | | Физическая схема. База данных книжного магазина | | | | Формат А1 | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  | БГУИР КП I- 40 01 01 020 ПЗ | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм | Лист | № докум. | Подп. | Дата | «База данных книжного магазина»  Ведомость курсового  Проекта | Литера | | | Лист | Листов |
| Разраб. | | Панкратьев Е.С. |  |  | Т |  |  | 71 | 71 |
| Провер. | | Марина И.М. |  |  | Кафедра ПОИТ  гр. 251003 | | | | |