

Министерство образования, науки и молодёжной политики Республики Коми
ГПОУ «Сыктывкарский политехнический техникум»

Курсовая работа

Разработка базы данных для домашней библиотеки

выполнила

студентка 4 курса

414 группы

Егорова Юлия Викторовна

проверил

Пунгин И.В.

дата проверки:

Сыктывкар, 2025 г.

СОДЕРЖАНИЕ

Введение	3
Актуальность темы	3
Цель работы	3
Задачи работы	4
Основная часть	6
Перечень используемого ПО при разработке	6
Анализ предметной области	8
1. Функциональные требования	10
2. Нефункциональные требования	11
3. Перечень входных данных	12
4. Перечень выходных данных	13
5. Ограничения предметной области	14
6. Взаимодействие с другими программами	16
Инфологическая (концептуальная) модель базы данных	17
Выделение информационных объектов	17
Определение атрибутов объектов	18
Определение отношений и мощности отношений между объектами	21
Построение концептуальной модели	22
Логическая структура БД	24
Физическая структура базы данных	27
Реализация проекта в среде конкретной СУБД	33
Заключение	56
Список литературы	58
Приложения	59

Введение

Актуальность темы

По данным исследования ВЦИОМ, большинство россиян (87%) имеют домашние библиотеки. Эти коллекции зачастую состоят из научной и художественной литературы. При этом размер этих библиотек варьируется: у половины семей (50%) они насчитывают до 100 книг. Коллекции объёмом от 100 до 300 книг встречаются у 21% опрошенных. Более обширные собрания имеются у 18% респондентов, среди которых 9% владеют 300-500 книгами, 5% — 500-1000, и 4% — свыше тысячи книг. Отсутствие книг дома отмечено у 11% россиян.

Владельцы таких библиотек активно взаимодействуют, обмениваясь изданиями и организуя литературные клубы. В контексте такого обмена или личного пользования возникает необходимость в систематизированном учёте книг. Традиционные методы учёта, такие как ведение записей на бумаге, становятся неэффективными при значительном объёме коллекции, а также затрудняют получение общей статистики и поиск информации.

Именно для решения этих задач актуальной становится разработка базы данных (БД) для домашней библиотеки. Она предназначена для автоматизации учёта состояния, местонахождения и выдачи книг, что особенно важно для редких или лично ценных экземпляров. Кроме того, такая система позволяет эффективно управлять коллекциями, включающими собственные авторские произведения или сборники. Использование БД обеспечивает лёгкий доступ к информации, упрощает инвентаризацию и аналитику.

Цель работы

Проектирование и реализация реляционной базы данных для комплексного учёта домашней библиотеки с интегрированной системой возрастных ограничений, а также специальных пометок в соответствии с законодательством РФ. Системное управление книжным домашним фондом, которое должно быть эффективным и понятным для гостя и администратора БД.

Задачи работы

1. Анализ предметной области и формирование требований к базе данных

1.1. Провести исследование особенностей домашних библиотек как объекта автоматизации, выделить ключевые сущности и атрибуты (учёт книг, выдача, возврат, контроль состояния).

1.2. Изучить законодательные требования Федерального закона №436-ФЗ "О защите детей от информации" и определить механизмы их технической реализации в БД (система возрастных рейтингов 0+, 6+, 12+, 16+, 18+).

2. Проектирование архитектуры и моделей базы данных

2.1. Разработать модель-схему в виде ER-диаграммы, отображающую сущности предметной области (Книга, Автор, Экземпляр, Пользователь, Местоположение и другие) и связи между ними.

2.2. Спроектировать логическую модель реляционной базы данных, а именно преобразование ER-диаграммы в схему таблиц, определение типов данных для полей, установление первичных и внешних ключей.

2.3. Создать физическую модель для СУБД PostgreSQL, определив стратегии индексирования, секционирования, хранения данных и оптимизации производительности.

3. Разработка и реализация структуры базы данных в СУБД PostgreSQL

3.1. Реализовать DDL-скрипты для создания всех таблиц базы данных с использованием современных возможностей PostgreSQL.

3.2. Спроектировать модель ролевого доступа (RBAC) с тремя уровнями привилегий: Администратор (полный доступ), Библиотекарь (операции с книгами и выдачей), Гость (только чтение справочной информации).

3.3. Настроить систему аудита и протоколирования всех значимых событий в отдельной таблице журнала (логирование выдачи/возврата, попыток нарушения доступа, изменений данных администратором).

4. Тестирование и подготовка документации

4.1. Провести комплексное тестирование базы данных: функциональное тестирование всех разработанных процедур и триггеров, тестирование ключевых запросов, тестирование на корректность работы ограничений целостности и других команд, отвечающих за корректную работу базы данных.

4.2. Подготовить демонстрационные данные и сценарии использования, наглядно показывающие работу всех функций БД, включая работу системы возрастных ограничений и механизмов безопасности.

Основная часть

Перечень используемого ПО при разработке.

В данном разделе рассматриваются основные условия, в которых разрабатывается БД, а также нюансы использования СУБД PostgreSQL для проектирования, разработки и практического применения БД.

1. Операционная система Windows 11

Windows 11 — это современная ОС на базе гибридного ядра Windows NT, разработанная компанией Microsoft для персональных компьютеров, рабочих станций и серверов. Архитектура Windows 11 оптимизирована для разработки и развёртывания сложных приложений, включая системы управления базами данных.

2. СУБД PostgreSQL

2.1. Структурирование, организация данных и принцип работы

PostgreSQL предоставляет упорядоченные механизмы для создания и управления структуризованными базами данных. Система позволяет проектировать сложные модели данных через систему таблиц, где каждая таблица представляет собой отдельную сущность предметной области (например, "Книга", "Автор", "Читатель").

Таблицы состоят из столбцов с заранее определёнными типами данных (текст, числа, даты), что обеспечивает строгую типизацию и предотвращает ошибки при вводе информации. Связи между таблицами устанавливаются через механизмы внешних ключей, создавая единую целостную структуру.

2.2. Обеспечение целостности данных

Одной из ключевых возможностей PostgreSQL является встроенная система ограничений целостности. Разработчик может определить:

- 2.2.1. Первичные ключи для уникальной идентификации каждой записи;
- 2.2.2. Внешние ключи для поддержания ссылочной целостности между таблицами;

2.2.3. CHECK-ограничения для проверки значений на соответствие бизнес-правилам (например, год издания не может быть больше текущего года);

2.2.4. UNIQUE-ограничения для предотвращения дублирования данных.

Эти механизмы работают на уровне ядра СУБД, гарантируя, что в базе данных не появятся некорректные или противоречивые записи.

2.3. Система безопасности и контроля доступа

Безопасность данных обеспечивается многоуровневой системой:

2.3.1. Аутентификация пользователей через различные методы (пароль, SSL-сертификаты);

2.3.2. Авторизация через систему привилегий GRANT/REVOKE на уровне объектов базы данных;

2.3.3. Безопасность на уровне строк (RLS), позволяющая ограничивать доступ к отдельным записям в таблице;

2.3.4. Аудит и журналирование всех значимых событий в системе;

2.3.5. Шифрование соединений через SSL/TLS для защиты передаваемых данных.

2.4. Расширяемость и автоматизация бизнес-процессов

PostgreSQL поддерживает создание собственной бизнес-логики:

2.4.1. Хранимые процедуры и функции на языке pgSQL для реализации сложных алгоритмов;

2.4.2. Триггеры — автоматически выполняемые процедуры при наступлении определённых событий (вставка, обновление, удаление);

2.4.3. Планировщик заданий pg_cron для выполнения периодических операций;

2.4.4. Расширения (extensions) для добавления новой функциональности (геопространственные данные, полнотекстовый поиск и др.).

3. Графический клиент-приложение pgAdmin 4

PgAdmin4 — это веб-приложение с открытым исходным кодом для управления и администрирования баз данных PostgreSQL. Данная модификация предоставляет удобный интерфейс для выполнения SQL-запросов, управления объектами базы данных, мониторинга производительности и многое другое.

3.1. Визуальное проектирование и моделирование данных

- 3.1.1. Графический конструктор таблиц с возможностью определения типов данных, ограничений и значений по умолчанию;
- 3.1.2. Редактор ER-диаграмм для визуального отображения связей между сущностями;
- 3.1.3. Мастер создания внешних ключей с автоматической генерацией соответствующих SQL-команд;
- 3.1.4. Инструменты для создания диаграмм из существующих баз данных;

3.2. Работа с содержимым базы данных

- 3.2.1. Табличный редактор данных, с поддержкой вставки, редактирования и удаления и связки записей;
- 3.2.2. Импорт данных из CSV, Excel и других форматов через графический интерфейс;
- 3.2.3. Визуальный конструктор запросов Query Tool для построения SQL-запросов без написания кода.

Анализ предметной области

Для начала работы необходимо теоретически проанализировать ежедневную работу домашних, а также государственных и частных библиотек, найти литературу и статьи о их работе, принципах ведения учёта и выдачи книг. Изучить нормативно-правовую базу, чтобы не нарушить законодательство Российской Федерации при некоторых аспектах.

Типичные операции, которые выполняются при управлении личным книжным фондом: от простой каталогизации новых поступлений до сложных процессов отслеживания выданных книг и контроля их состояния, становятся неэффективными уже при коллекции в 100-150 книг, вызывая трудности с поиском, дублированием записей и получением общей статистики. Также были изучены потребности в автоматизации рутинных процессов: автоматической проверке

возрастных ограничений при выдаче книг детям, напоминаниях о возврате, ведении истории состояния экземпляров и генерации отчётов о наиболее популярных книгах. Этот анализ позволил выделить ключевые функции, которые действительно необходимы пользователям, и отсечь избыточные возможности, усложняющие систему. Важнейшим аспектом формирования требований стало изучение законодательных актов, регулирующих обращение печатной продукции в Российской Федерации. Центральное место занял Федеральный закон №436-ФЗ "О защите детей от информации, причиняющей вред их здоровью и развитию", который устанавливает обязательную возрастную маркировку для всех видов контента, включая книги. В соответствии с этим законом, все книги должны классифицироваться по возрастным категориям: 0+, 6+, 12+, 16+, 18+. Система обязана выдавать дополнительное уведомление для выдающего о проверки соответствия возраста читателя и возрастного рейтинга книги при каждой выдаче, а также вести журнал таких проверок. Дополнительно были учтены правовые аспекты учёта авторских прав — система должна позволять отмечать информацию об авторах, включая их статус (отечественный/зарубежный, с указанием на особые правовые статусы, такие как "иностранный агент"). Эти требования обеспечивают не только функциональность системы, но и её юридическую корректность при использовании в российских реалиях.

Требования к системе разрабатывались с учётом реальных технических условий, в которых будет эксплуатироваться база данных. Был проведён анализ характеристик типовых домашних компьютеров — большинство пользователей имеют ПК с 4-8 ГБ оперативной памяти, процессорами среднего класса и обычными жёсткими дисками или SSD-накопителями. Это определило требования к производительности: система должна работать стабильно на таком оборудовании даже при каталоге в несколько тысяч книг. Оценка возможностей современных систем управления базами данных, в частности PostgreSQL, позволила заложить в требования продвинутые функции: сложные ограничения целостности, триггеры для автоматической проверки бизнес-правил, механизмы полнотекстового поиска и систему ролевого доступа. Также были учтены требования к простоте развёртывания — система должна устанавливаться на стандартную Windows без необходимости сложной настройки серверного окружения, что соответствует возможностям типичного домашнего пользователя.

После анализа можно сделать выводы по необходимому функционалу БД для домашней библиотеки:

1. Функциональные требования

1.1. Управление книжным фондом

- 1.1.1. Учёт физических экземпляров с уникальными инвентарными номерами;
- 1.1.2. Регистрация состояния каждого экземпляра с историей изменений;
- 1.1.3. Отслеживание мест хранения книг;
- 1.1.4. Контроль условий хранения для ценных изданий.

1.2. Каталогизация и классификация;

- 1.2.1. Гибкая система добавления новых жанров и категорий;
- 1.2.2. Множественная классификация изданий (по авторам, жанрам, тегам);
- 1.2.3. Поддержка расширенных характеристик книг;
- 1.2.4. Возрастная маркировка согласно законодательству РФ.

1.3. Управление пользователями и доступом.

- 1.3.1. Трёхуровневая система ролей (администратор, библиотекарь, гость);
- 1.3.2. Регистрация читателей с обязательным указанием даты рождения;
- 1.3.3. Автоматический расчёт возраста пользователей;
- 1.3.4. Система аудита всех действий пользователей.

1.4. Учёт выдачи и возврата книг.

- 1.4.1. Полная регистрация операций выдачи/возврата;
- 1.4.2. Автоматическая проверка возрастных ограничений;
- 1.4.3. Контроль сроков возврата с системой уведомлений;

1.4.4. Предотвращение двойной выдачи одного экземпляра.

1.5. Поиск и аналитика.

1.5.1. Расширенный поиск по множеству критериев;

1.5.2. Формирование отчётов о популярности и активности;

1.5.3. Статистический анализ читательских предпочтений;

1.5.4. Мониторинг состояния книжного фонда.

2. Нефункциональные требования

2.1. Правовое соответствие;

2.1.1. Соблюдение Федерального закона №436-ФЗ;

2.1.2. Обязательная возрастная маркировка книг;

2.1.3. Поддержка отметок об особых статусах авторов.

2.2. Безопасность и контроль доступа

2.2.1. Разграничение прав на операции с данными;

2.2.2. Безопасность на уровне строк для персональных данных;

2.2.3. Защита от несанкционированных изменений.

2.3. Гибкость и расширяемость

2.3.1. Возможность добавления новых полей без перестройки структуры;

2.3.2. Поддержка множественной классификации изданий;

2.3.3. Совместимость с будущими расширениями функционала.

2.4. Доступность и экономичность

2.4.1. Использование бесплатной СУБД PostgreSQL;

2.4.2. Работа на стандартном домашнем оборудовании;

2.4.3. Минимальные требования к квалификации пользователя.

2.5. Производительность и надёжность

2.5.1. Стабильная работа с каталогом до нескольких тысяч книг;

2.5.2. Быстрый поиск и формирование отчётов;

2.5.3. Надёжное хранение данных и история изменений;

2.5.4. Данные о литературных произведениях.

3. Перечень входных данных.

3.1 Библиографические данные:

3.1.1. Название книги (обязательное поле);

3.1.2. Оригинальное название (для иностранной литературы);

3.1.3. Год издания;

3.1.4. Издательство;

3.1.5. Количество страниц;

3.1.6. Аннотация/краткое содержание.

3.2. Классификационные данные:

3.2.1. Возрастной рейтинг (0+, 6+, 12+, 16+, 18+) — обязательное поле согласно 436-ФЗ;

3.2.2. Жанры (множественный выбор).

3.3. Данные об авторах:

3.3.1. Фамилия;

3.3.2. Имя;

3.3.3. Отчество;

3.3.4. Год рождения;

3.3.5. Год смерти (для умерших авторов);

3.3.6. Страна происхождения;

3.3.7. Специальные отметки (иностранный агент, лауреат премий и т.д.);

3.3.8. Альтернативные пометки авторства (народный фольклор, утерянное авторство).

3.4. Данные о физических экземплярах книг:

3.4.1. Уникальный инвентарный номер (обязательное поле)

- 3.4.3. Дата поступления в библиотеку;
- 3.4.5. Физическое состояние (отличное, хорошее, удовлетворительное, плохое);
- 3.4.6. Особые отметки (автограф автора, повреждения).

3.5. Данные о хранении:

- 3.5.1. Место хранения (комната, шкаф, полка)

3.6. Данные о пользователях системы:

- 3.6.1. Фамилия (обязательное поле);
- 3.6.2. Имя (обязательное поле);
- 3.6.3. Отчество;
- 3.6.4. Дата рождения (обязательное поле для проверки возрастных ограничений);
- 3.6.5. Контактная информация (email, телефон);
- 3.6.6. Логин и пароль для доступа к системе.
- 3.6.7. Роль в системе (администратор, библиотекарь, гость)

4. Перечень выходных данных.

4.1. Полный каталог книг.

- 4.1.1. Список всех книг с библиографическими данными (название, автор, год издания, издаельство);
- 4.1.2. Классификация по жанрам и возрастным рейтингам;
- 4.1.3. Информация о доступности экземпляров.

4.2. Авторский указатель.

- 4.2.1. Алфавитный список авторов с их произведениями;
- 4.2.2. Статистика по авторам (количество книг, годы жизни, страна);
- 4.2.3. Специальные пометки об авторах (лауреаты премий, особые статусы).

4.3. Систематический каталог.

- 4.3.1. Группировка книг по жанрам и тематикам;
- 4.3.2. Кросс-справочные указатели (автор-жанр, жанр-рейтинг);
- 4.3.3. Инвентарные описи по местам хранения.

4.4. Журнал выдачи и возврата книг.

- 4.4.1. Хронологический список всех операций выдачи;
- 4.4.2. Информация о текущих задолженностях (книги на руках);
- 4.4.3. История выдачи по каждому читателю и экземпляру;
- 4.4.4. Лог проверок возрастных ограничений при выдаче;
- 4.4.5. Статистика нарушений возрастных ограничений;
- 4.4.6. Список книг, требующих особого контроля при выдаче.

5. Ограничения предметной области.

5.1. Законодательные ограничения.

- 5.1.1. Все книги должны иметь обязательную возрастную маркировку: (0+), (6+), (12+), (16+), (18+);
- 5.1.2. Запрещена выдача книг с рейтингом 18+ читателям младше 18 лет;
- 5.1.3. Требуется обязательная проверка возраста читателя при выдаче книг с рейтингом 16+.

5.2. Авторское право.

- 5.2.1. Отметка об особых правовых статусах авторов.

5.3. Функциональные ограничения.

- 5.3.1. Один экземпляр книги не может быть выдан одновременно нескольким читателям;
- 5.3.2. Максимальный срок выдачи книги - 30 дней;
- 5.3.3. Ограничение количества одновременно взятых книг на одного читателя - 5 экземпляров.

5.4. Ограничения по учёту экземпляров.

5.4.1. Уникальный инвентарный номер для каждого физического экземпляра;

5.4.2. Запрет удаления книг с активными выдачами.

5.5. Ограничения данных.

5.5.1. Максимальное количество книг в базе - 10 000 записей

5.5.2. Максимальное количество читателей - 100 человек

5.5.3. Ограничение на длину текстовых полей:

- Название книги: не более 200 символов
- ФИО автора: не более 100 символов
- Описание: не более 2000 символов

5.6. Временные ограничения.

5.6.1. Год издания книги не может быть больше текущего года;

5.6.2. Дата рождения читателя должна быть не позднее текущей даты;

5.6.3. Дата выдачи книги не может быть раньше даты поступления экземпляра;

5.6.4. Срок возврата должен быть не ранее даты выдачи.

5.7. Ограничения доступности.

5.7.1. Книги могут находиться только в трех состояниях: "доступна", "выдана", "требуется реставрация";

5.7.2. Экземпляры в состоянии "требуется реставрация" не могут быть выданы;

5.7.3. Книги с состоянием "плохое" требуют подтверждения администратора для выдачи.

5.8. Ограничения безопасности.

5.8.1. Гости могут только просматривать каталог книг;

5.8.2. Библиотекари не могут удалять пользователей и изменять системные настройки;

5.8.3. Только администратор может изменять возрастные рейтинги и системные справочники;

5.8.3. Запрещен доступ к персональным данным других пользователей.

5.9. Ограничения целостности данных.

5.9.1. Запрещено удаление авторов, у которых есть книги в базе;

5.9.2. Запрещено удаление жанров, которые используются в классификации книг;

5.9.3. Невозможность изменить инвентарный номер существующего экземпляра;

5.9.4. Сохранение истории всех изменений критических данных.

6. Взаимодействие с другими программами.

Система управления домашней библиотекой может быть интегрирована с различными внешними сервисами и программами для расширения функциональности и автоматизации процессов:

- **Онлайн-каталоги книг** (Google Books API, Open Library):
Интеграция с общедоступными книжными базами данных позволит автоматически заполнять информацию о книгах (автор, год издания, аннотация, обложка) при добавлении по ISBN или названию.
- **Электронные библиотечные системы** (Литрес, MyBook):
Взаимодействие с популярными электронными библиотеками предоставит пользователям доступ к дополнительной информации о книгах, рецензиям, рейтингам и рекомендациям.
- **Системы учета имущества:** Интеграция с программами инвентарного учета (например, 1С) позволит автоматизировать процесс списания утерянных или поврежденных книг и вести полную финансовую отчетность.
- **Социальные сети и книжные сообщества** (Goodreads, LiveLib):
Экспорт данных в популярные книжные социальные сети позволит пользователям делиться своей коллекцией, участвовать в рейтингах и получать персонализированные рекомендации.

Инфологическая (концептуальная) модель базы данных

Выделение информационных объектов

На основе анализа предметной области были выделены следующие ключевые информационные объекты:

Основные информационные объекты:

КНИГА (BOOK) - библиографическое описание литературного произведения

- Уникальное описание произведения независимо от количества экземпляров
- Содержит метаданные: название, год издания, издательство и т.д.

АВТОР (AUTHOR) - создатель литературного произведения

- Может быть физическим лицом или иметь особый статус (фольклор, неизвестный автор)
- Одна книга связана с одним автором (связь 1:N согласно диаграмме)

ЧИТАТЕЛЬ (USER) - пользователь библиотеки

- Человек, который берет книги для чтения
- Имеет обязательные данные для идентификации и проверки возрастных ограничений

ЭКЗЕМПЛЯР (COPY) - физический экземпляр книги

- Конкретная книга, которую можно взять в руки
- Имеет уникальный инвентарный номер и собственное состояние

Справочные сущности:

ВОЗРАСТНОЙ РЕЙТИНГ (AGE_RATING) -
классификация по возрастным ограничениям

- Фиксированный набор значений: 0+, 6+, 12+, 16+, 18+
- Определяет минимальный возраст читателя для выдачи книги

ЖАНР (GENRE) - литературный жанр произведения

- Категория для классификации книг (роман, детектив, фантастика и т.д.)
- Одна книга может относиться к нескольким жанрам (требуется таблица связей)

МЕСТО ХРАНЕНИЯ (LOCATION) - физическое расположение книги

- Иерархическая структура: комната → шкаф/мебель → полка
- Позволяет быстро находить книги в пространстве

Операционные сущности:

1. **ВЫДАЧА (LOAN) -** факт выдачи книги читателю

- Фиксирует момент передачи экземпляра и обязательство возврата
- Содержит информацию о проверке возрастных ограничений

Определение атрибутов объектов

КНИГА (BOOK):

- Идентификатор книги (book_id) - уникальный числовой идентификатор (PK)
- Возрастной рейтинг (rating_id) - ссылка на AGE_RATING (FK)
- Автор (author_id) - ссылка на AUTHOR (FK)
- Название (title) - название произведения (обязательное поле)
- Оригинальное название (original_title) - для иностранной литературы
- Год издания (publication_year) - год публикации
- Издательство (publisher) - название издательства
- Описание (description) - аннотация или краткое содержание
- Дата создания записи (created_at) - когда книга внесена в каталог

АВТОР (AUTHOR):

- Идентификатор автора (author_id) - уникальный числовой идентификатор (PK)

- Фамилия (last_name) - фамилия автора
- Имя (first_name) - имя автора
- Отчество (middle_name) - отчество автора (при наличии)
- Год рождения (birth_year) - год рождения автора
- Год смерти (death_year) - год смерти (если автор умер)
- Страна (country) - страна происхождения
- Особые отметки (special_notes) - статусы: лауреат премий, иностранный агент и т.д.
- Тип авторства (author_type) - индивидуальный, народный, неизвестный

ЧИТАТЕЛЬ (USER):

- Идентификатор читателя (user_id) - уникальный числовой идентификатор (PK)
- Фамилия (last_name) - фамилия читателя (обязательное поле)
- Имя (first_name) - имя читателя (обязательное поле)
- Отчество (middle_name) - отчество читателя
- Дата рождения (birth_date) - дата рождения (обязательное для возрастного контроля)
- Электронная почта (email) - контактный email
- Телефон (phone) - контактный телефон
- Роль в системе (role) - администратор, библиотекарь, читатель и т.д.
- Статус активности (is_active) - активен/заблокирован

ЭКЗЕМПЛЯР (COPY):

- Идентификатор экземпляра (copy_id) - уникальный числовой идентификатор (PK)
- Книга (book_id) - ссылка на BOOK (FK)
- Место хранения (location_id) - ссылка на LOCATION (FK)
- Инвентарный номер (inventory_number) - уникальный физический номер (обязательное поле)
- Дата поступления (acquisition_date) - когда книга попала в библиотеку
- Стоимость приобретения (purchase_price) - цена покупки
- Оценочная стоимость (estimated_value) - текущая рыночная стоимость
- Состояние (condition) - отличное, хорошее, удовлетворительное, плохое
- Особые отметки (special_notes) - автографы, повреждения, экслибрисы

- Доступность (is_available) - доступен для выдачи или нет

ВОЗРАСТНОЙ РЕЙТИНГ (AGE_RATING):

- Идентификатор рейтинга (rating_id) - уникальный числовой идентификатор (PK)
- Код рейтинга (code) - текстовое или числовое обозначение
- Минимальный возраст (min_age) - минимально допустимый возраст читателя
- Описание (description) - пояснение рейтинга

ЖАНР (GENRE):

- Идентификатор жанра (genre_id) - уникальный числовой идентификатор (PK)
- Название жанра (name) - наименование жанра (обязательное поле)
- Описание (description) - характеристика жанра

МЕСТО ХРАНЕНИЯ (LOCATION):

- Идентификатор места (location_id) - уникальный числовой идентификатор (PK)
- Комната (room) - номер или название комнаты
- Шкаф/стеллаж (furniture) - тип мебели для хранения
- Полка (shelf) - номер или обозначение полки
- Описание (description) - дополнительные сведения о месте

ВЫДАЧА (LOAN):

- Идентификатор выдачи (loan_id) - уникальный числовой идентификатор (PK)
- Экземпляр (copy_id) - ссылка на COPY (FK)
- Читатель (user_id) - ссылка на USER (FK)
- Дата выдачи (loan_date) - когда книга выдана (обязательное поле)
- Срок возврата (due_date) - когда должна быть возвращена
- Дата возврата (return_date) - фактическая дата возврата
- Приостановка возрастной проверки (age_check_paused) - отметка о приостановке проверки
- Примечания (notes) - дополнительные сведения о выдаче

Определение отношений и мощности отношений между объектами

Основные отношения:

КНИГА → АВТОР (многие-к-одному)

- Одна книга связана с одним автором (согласно диаграмме)
- Один автор может написать несколько книг
- Мощность: N:1
- Реализация: FK author_id в таблице BOOK

КНИГА ←→ ЖАНР (многие-ко-многим)

- Одна книга может относиться к нескольким жанрам
- Один жанр может включать много книг
- Мощность: M:N
- Реализация через таблицу: BOOK_GENRE (требуется дополнительная таблица)

КНИГА → ЭКЗЕМПЛЯР (один-ко-многим)

- Одна книга (библиографическое описание) может иметь несколько физических экземпляров
- Один экземпляр относится только к одной книге
- Мощность: 1:N
- Реализация: FK book_id в таблице COPY

ЭКЗЕМПЛЯР → МЕСТО ХРАНЕНИЯ (многие-к-одному)

- Один экземпляр хранится в одном месте
- В одном месте может храниться много экземпляров
- Мощность: N:1
- Реализация: FK location_id в таблице COPY

КНИГА → ВОЗРАСТНОЙ РЕЙТИНГ (многие-к-одному)

- Одна книга имеет один возрастной рейтинг
- Один рейтинг может быть присвоен многим книгам
- Мощность: N:1
- Реализация: FK rating_id в таблице BOOK

ЧИТАТЕЛЬ → ВЫДАЧА (один-ко-многим)

- Один читатель может брать много книг (в разное время)
- Одна выдача относится к одному читателю
- Мощность: 1:N
- Реализация: FK user_id в таблице LOAN

ЭКЗЕМПЛЯР → ВЫДАЧА (один-ко-многим)

- Один экземпляр может выдаваться много раз (в разное время)
- Одна выдача относится к одному экземпляру
- Мощность: 1:N
- Реализация: FK copy_id в таблице LOAN

Дополнительные ограничения отношений:

Временные ограничения:

- Дата выдачи \leq Срок возврата
- Дата возврата \geq Дата выдачи (если возвращено)
- Год издания \leq Текущий год
- Дата рождения читателя \leq Текущая дата
- Дата смерти автора \geq Дата рождения автора (если указана)

Бизнес-ограничения:

- Нельзя выдать уже выданный экземпляр (без даты возврата)
- Нельзя выдать книгу читателю младше возрастного рейтинга (кроме случаев age_check_paused)
- Максимум 5 книг одновременно у одного читателя (логика на уровне приложения)
- Экземпляр с is_available = FALSE нельзя выдать
- Инвентарный номер должен быть уникальным

Построение концептуальной модели

Концептуальная модель базы данных является фундаментальным этапом проектирования, который позволяет визуализировать структуру данных и взаимосвязи между сущностями предметной области без привязки к конкретной СУБД. На основе проведённого анализа предметной области, функциональных и нефункциональных требований, а также выделенных информационных объектов была разработана ER-диаграмма (Entity-Relationship Diagram),

отражающая ключевые сущности домашней библиотеки и отношения между ними.

Методология разработки ER-диаграммы

Разработка ER-диаграммы осуществлялась в соответствии со следующими принципами:

Идентификация объектов — на основе анализа предметной области были выделены основные сущности: Книга (BOOK), Автор (AUTHOR), Читатель (USER), Экземпляр (COPY), а также справочные сущности: Возрастной рейтинг (AGE_RATING), Жанр (GENRE) и Место хранения (LOCATION).

Определение атрибутов — для каждой сущности были определены ключевые атрибуты с учётом бизнес-требований и законодательных ограничений (Федеральный закон №436-ФЗ).

Установление отношений — между сущностями были определены связи с указанием мощности (кардинальности) отношений:

- Один-ко-многим (1:N)
- Многие-ко-многим (M:N)
- Один-к-одному (1:1)

Визуальное представление — для наглядности использована нотация "Crow's Foot" (нотация "воронья лапка"), которая является стандартом де-факто в проектировании баз данных.

Ключевые отношения на диаграмме

Диаграмма наглядно демонстрирует следующие основные отношения:

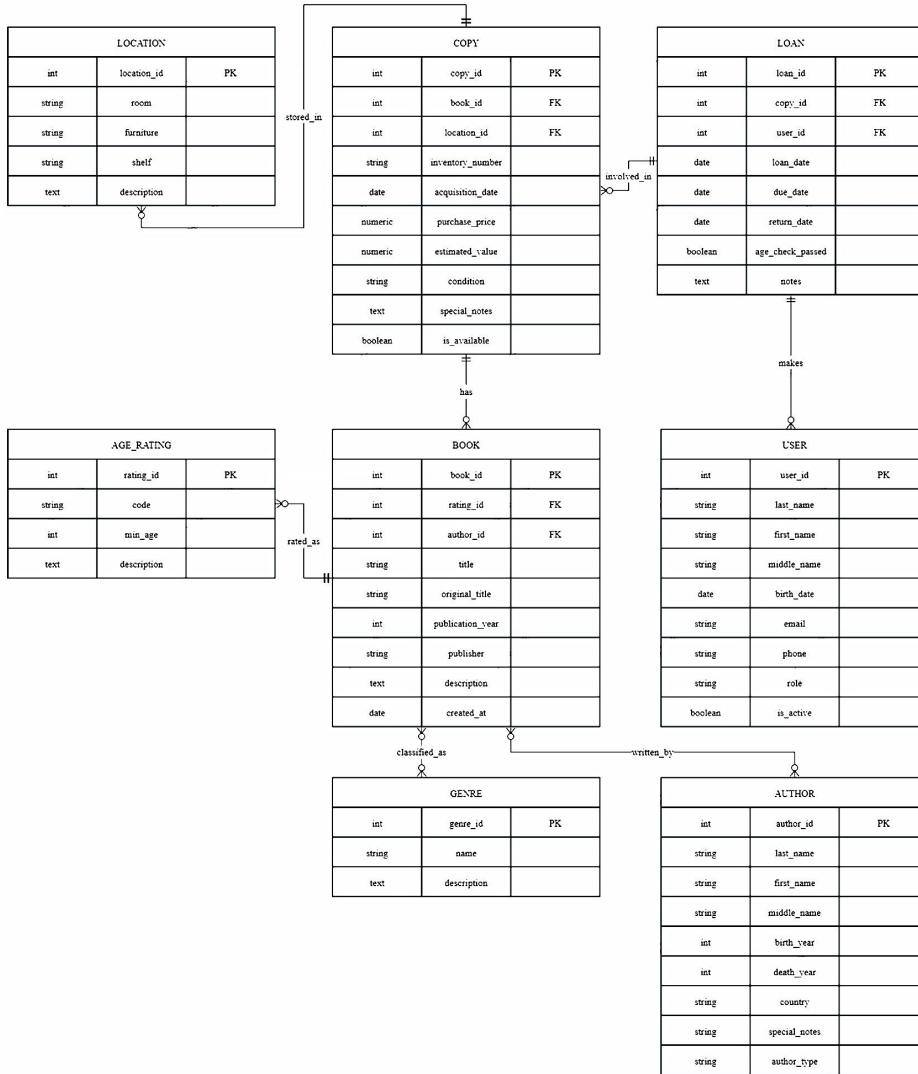
Книга ↔ Автор — связь многие-к-одному, где одна книга имеет одного автора, а один автор может написать несколько книг.

Книга ↔ Жанр — связь многие-ко-многим, реализуемая через дополнительную таблицу BOOK_GENRE.

Книга ↔ Экземпляр — связь один-ко-многим, где одно библиографическое описание соответствует нескольким физическим экземплярам.

Экземпляр ↔ Выдача — связь один-ко-многим, позволяющая отслеживать историю выдачи каждого экземпляра.

Читатель ↔ Выдача — связь один-ко-многим, фиксирующая все выдачи книг конкретному читателю.



Изображение 1 — ER-диаграмма базы данных домашней библиотеки

Логическая структура БД

Логическая структура базы данных представляет собой реализацию концептуальной модели в терминах реляционной модели данных. На данном этапе происходит преобразование сущностей и отношений ER-диаграммы в схему таблиц с конкретными типами данных, ограничениями целостности и определением связей между таблицами.

1. Принципы преобразования ER-модели в реляционную схему

Преобразование концептуальной модели в логическую структуру осуществлялось по следующим правилам:

1. **Сущности → Таблицы** — каждая сущность ER-диаграммы преобразуется в отдельную таблицу.
2. **Атрибуты → Столбцы** — атрибуты сущностей становятся столбцами соответствующих таблиц.
3. **Первичные ключи** — для каждой таблицы определяется первый ключ для уникальной идентификации записей.
4. **Внешние ключи** — отношения между сущностями реализуются через механизм внешних ключей.
5. **Отношения M:N → Промежуточные таблицы** — связи многие-ко-многим преобразуются в отдельные таблицы связей.
6. **Ограничения целостности** — добавляются CHECK-ограничения для обеспечения бизнес-правил.

2. Список таблиц базы данных

Логическая схема базы данных домашней библиотеки включает следующие таблицы:

№	Таблица	Назначение	Примерное количество записей
1	author	Хранение информации об авторах книг	5,000
2	age_rating	Справочник возрастных рейтингов (согласно 436-ФЗ)	5
3	genre	Справочник литературных жанров	50
4	book	Библиографические описания книг	10,000
5	book_genre	Связь книг с жанрами (многие-ко-многим)	30,000
6	location	Справочник мест хранения	100
7	copy	Физические экземпляры книг	15,000
8	user	Пользователи системы (читатели, библиотекари,	100

№	Таблица	Назначение	Примерное количество записей	
		администраторы)		
9	loan	Журнал выдачи и возврата книг	50,000	

Логическая структура базы данных

Логическая структура БД соответствует третьей нормальной форме (3NF), что обеспечивает:

- Отсутствие транзитивных зависимостей** — все неключевые атрибуты зависят только от первичного ключа.
- Устранение избыточности данных** — каждая сущность хранится только в одной таблице.
- Целостность данных** — использование внешних ключей гарантирует согласованность данных.

Пример нормализации:

- Информация об авторе хранится только в таблице author
- В таблице book хранится только ссылка на автора (author_id)
- Это предотвращает дублирование данных об авторе для каждой книги

Обеспечение целостности данных

Логическая структура включает следующие механизмы обеспечения целостности:

- Первичные ключи** — гарантируют уникальность каждой записи.
- Внешние ключи** — обеспечивают ссылочную целостность между таблицами.
- СЧЕСК-ограничения** — проверяют значения на соответствие бизнес-правилам.
- UNIQUE-ограничения** — предотвращают дублирование уникальных значений.
- NOT NULL** — обеспечивают обязательность заполнения критических полей.

Преимущества разработанной логической структуры

1. **Гибкость** — позволяет легко добавлять новые книги, авторов, жанры.
2. **Масштабируемость** — структура поддерживает до 10,000 книг и 100 пользователей.
3. **Производительность** — оптимальное распределение данных по таблицам.
4. **Поддержка законодательства** — встроенная система возрастных рейтингов.
5. **Безопасность** — разделение ролей и прав доступа.

Физическая структура базы данных

Физическая структура базы данных отвечает за реальное хранение данных на физическом уровне, определяет способы хранения и работы с данными, обеспечивая при этом их целостность, доступность и производительность системы. На этом этапе важно учитывать оптимизацию производительности запросов, объёмы данных и ограничения, связанные с физическим хранением данных в выбранной СУБД. В данной курсовой работе предполагается использование реляционной СУБД PostgreSQL, так как она предоставляет оптимальные средства для управления табличными данными и поддерживает сложные связи между данными, что необходимо для домашней библиотеки с учётом требований законодательства РФ о возрастных ограничениях.

Физическая структура проектируется таким образом, чтобы система максимально эффективно обрабатывала данные, предоставляя пользователю доступ к нужной информации с минимальной задержкой. Это достигается с помощью выбора правильных типов данных, создания оптимальных индексов, реализации механизмов хранения и настройки параметров, обеспечивающих производительность и надежность. Особое внимание уделено обеспечению соответствия Федеральному закону №436-ФЗ "О защите детей от информации, причиняющей вред их здоровью и развитию".

Архитектура хранения данных

База данных "Домашняя библиотека" реализована в СУБД PostgreSQL версии 17 и состоит из 10 взаимосвязанных таблиц, организованных в третьей нормальной форме (3NF). Система спроектирована для хранения до 10,000 биографических записей книг и обслуживания до 100 пользователей.

Выбор типов данных

Правильный выбор типов данных для полей таблиц — один из ключевых аспектов проектирования физической структуры базы данных, так как это позволяет экономить ресурсы и повышать производительность. Рассмотрим выбор типов данных для каждой таблицы:

SERIAL — используется для автоинкрементных значений, таких как идентификаторы записей (`author_id`, `book_id`, `copy_id`, `user_id`). Это подходит для полей, которые используются как ключи и часто участвуют в соединениях и фильтрации данных. PostgreSQL автоматически создаёт соответствующий **SEQUENCE** для генерации уникальных значений.

INTEGER — используется для целочисленных значений, таких как годы издания книг (`publication_year`), годы рождения и смерти авторов (`birth_year`, `death_year`), минимальный возраст рейтинга (`min_age`). Занимает фиксированный объем памяти (4 байта) и обеспечивает эффективное индексирование.

VARCHAR(n) — применяется для хранения текстовых данных переменной длины, таких как имена авторов (`last_name`, `first_name`, `middle_name`), названия книг (`title`, `original_title`), издательства (`publisher`), номера помещений (`room`), типы мебели (`furniture`). Для полей с известными максимальными длинами используется ограничение по символам: `VARCHAR(50)` для имен, `VARCHAR(200)` для названий книг, `VARCHAR(100)` для стран. Это позволяет оптимально расходовать память, так как фактически занимает объем, соответствующий реальной длине строки.

TEXT — используется для полей с неопределенной или значительной длиной содержимого: описания книг (`description`), аннотации, особые отметки авторов (`special_notes`), примечания к экземплярам и выдачам. В PostgreSQL тип **TEXT** не имеет ограничения по длине (кроме максимального размера строки в 1 ГБ), что обеспечивает гибкость при хранении развернутых описаний.

DATE — этот тип данных используется для хранения дат: дата рождения читателей (`birth_date`), дата поступления экземпляров (`acquisition_date`), даты выдачи (`loan_date`), возврата (`return_date`) и сроков возврата (`due_date`). Он позволяет хранить значения в формате даты (год, месяц, день) и обеспечивает удобные функции для работы с датами, такие как сравнение и вычисление разницы между датами, что критично для контроля сроков выдачи.

NUMERIC(precision, scale) — применяется для хранения точных десятичных чисел с фиксированной точностью: цена покупки (`purchase_price`) и оценочная стоимость (`estimated_value`).

Используется параметр `NUMERIC(10,2)`, что позволяет хранить значения до `99,999,999.99` с двумя знаками после запятой, что достаточно для учета стоимости книг.

BOOLEAN — используется для хранения логических значений, которые могут принимать два состояния: `TRUE` (истина) или `FALSE` (ложь). Он часто используется для представления бинарных состояний, таких как доступность экземпляра для выдачи (`is_available`), активность пользователя (`is_active`), приостановка возрастной проверки (`age_check_paused`). Занимает 1 байт памяти.

JSONB — применяется в таблице аудита (`audit_log`) для хранения данных до и после изменений (`old_data`, `new_data`). Формат `JSONB` (Binary JSON) обеспечивает эффективное хранение и индексирование полуструктурированных данных, что позволяет гибко вести журнал изменений без необходимости изменения схемы таблицы при добавлении новых полей в аудируемые таблицы.

TIMESTAMP — используется для регистрации точного времени событий: время создания записи аудита (`action_timestamp`), время регистрации пользователя (`created_at`). Обеспечивает точность до микросекунд, что необходимо для корректной последовательности событий в системе аудита.

Тип данных влияет на объем занимаемой памяти и производительность базы данных, так как данные записываются и извлекаются в зависимости от того, какой тип используется. Правильный выбор типов минимизирует издержки ввода-вывода и ускоряет выполнение запросов.

Индексы для оптимизации производительности

Для обеспечения высокой производительности запросов созданы следующие индексы:

В-дерево индексы для точного поиска и сортировки:

- idx_author_last_name — быстрый поиск авторов по фамилии
- idx_user_email — уникальный индекс для быстрой аутентификации пользователей
- idx_loan_user_id и idx_loan_copy_id — оптимизация JOIN-запросов для выдачи книг
- idx_copy_inventory_number — уникальный индекс для быстрого поиска по инвентарному номеру

Частичные индексы для часто используемых подмножеств данных:

- idx_loan_return_date — только для записей с return_date IS NULL (текущие выдачи)
- idx_copy_available — только для доступных экземпляров (is_available = TRUE)

GIN индекс для полнотекстового поиска:

- idx_book_title — полнотекстовый поиск по названиям книг с поддержкой русского языка

UNIQUE ограничения, которые автоматически создают уникальные индексы:

- inventory_number в таблице copy
- email в таблице "user"
- code в таблице age_rating
- name в таблице genre

Стратегии хранения и параметры СУБД

База данных настроена с учетом следующих параметров производительности:

Табличные пространства: Используется пространство по умолчанию pg_default, что достаточно для домашней библиотеки с предполагаемым объемом до 10,000 книг.

Настройки PostgreSQL в файле postgresql.conf:

- `shared_buffers` = 128MB — 25% от 512MB RAM типичного домашнего компьютера
- `work_mem` = 4MB — для операций сортировки и хеширования
- `maintenance_work_mem` = 64MB — для операций обслуживания (VACUUM, CREATE INDEX)
- `effective_cache_size` = 256MB — оценка размера кэша ОС для планировщика запросов

Стратегия VACUUM: Включен автоматический vacuum (`autovacuum = on`) для своевременного освобождения места, занимаемого "мертвыми" кортежами, что особенно важно для часто обновляемых таблиц `loan` и `copy`.

Уровень изоляции транзакций: По умолчанию используется READ COMMITTED, что обеспечивает баланс между производительностью и согласованностью данных для многопользовательского доступа.

Механизмы обеспечения целостности данных

Физическая реализация включает следующие механизмы:

Первичные ключи: Все таблицы имеют первичные ключи типа SERIAL, гарантирующие уникальность каждой записи.

Внешние ключи с каскадными действиями:

- ON DELETE CASCADE для `book_genre` и `copy` — автоматическое удаление зависимых записей
- ON DELETE RESTRICT для `book` — предотвращение удаления используемых авторов и рейтингов
- ON DELETE SET NULL для `location_id` в `copy` — сохранение экземпляра при удалении места хранения

CHECK-ограничения на уровне СУБД:

- Проверка возрастных рейтингов (`code IN ('0+', '6+', '12+', '16+', '18+')`)
- Проверка состояния экземпляров (`condition IN ('excellent', 'good', 'satisfactory', 'poor')`)
- Проверка дат (`publication_year <= текущий_год, birth_date <= текущая_дата`)

Триггеры для сложной бизнес-логики:

- `check_age_rating()` — автоматическая проверка возрастных ограничений при выдаче
- `check_copy_availability()` — предотвращение выдачи недоступных экземпляров
- `check_max_books_per_user()` — ограничение количества книг на одного читателя
- `audit_book_changes()` — автоматическое ведение журнала аудита

Стратегия резервного копирования и восстановления

Для обеспечения надежности хранения данных предусмотрена следующая стратегия:

Ежедневное полное резервное копирование с использованием pg_dump в сжатом формате.

WAL (Write-Ahead Logging) архивирование для возможности point-in-time recovery.

Репликация (опционально) для защиты от сбоев оборудования.

Оценка объемов хранения

На основе выбранных типов данных произведена оценка занимаемого пространства:

Таблица	Кол-во записей	Средний размер записи	Общий объем
author	5,000	~200 байт	~1 МБ
book	10,000	~500 байт	~5 МБ
copy	15,000	~300 байт	~4.5 МБ
loan	50,000	~100 байт	~5 МБ
Итого	80,100	-	~16 МБ

Общий объем базы данных не превышает 20 МБ, что позволяет эффективно работать на стандартном домашнем компьютере с 4-8 ГБ оперативной памяти.

Реализация проекта в среде конкретной СУБД. Создание таблиц

Создание таблиц базы данных выполнено с использованием языка определения данных (DDL) SQL. Все таблицы созданы в схеме public с учетом нормализации данных до третьей нормальной формы (3NF), что исключает избыточность и обеспечивает целостность информации. Для каждой таблицы определены соответствующие типы данных, первичные и внешние ключи, а также ограничения целостности (CHECK-ограничения), гарантирующие соблюдение бизнес-правил и законодательных требований.

Основные таблицы включают: author для хранения информации об авторах с поддержкой различных типов авторства (индивидуальный, народный фольклор, неизвестный, коллективный); age_rating — справочник возрастных рейтингов согласно ФЗ №436-ФЗ с фиксированными значениями (0+, 6+, 12+, 16+, 18+); book — основная таблица библиографических описаний с обязательной ссылкой на возрастной рейтинг; copy — учет физических экземпляров с уникальными инвентарными номерами; "user" — таблица пользователей системы с тремя уровнями доступа (администратор, библиотекарь, читатель); loan — журнал выдачи и возврата книг с автоматической проверкой ограничений.

Особое внимание уделено таблице audit_log, предназначеннной для регистрации всех изменений критических данных в формате JSONB, что обеспечивает прозрачность и возможность восстановления информации. Все таблицы связаны через систему внешних ключей с настроенными каскадными действиями: ON DELETE CASCADE для зависимых записей, ON DELETE RESTRICT для предотвращения удаления используемых данных, ON DELETE SET NULL для сохранения ссылочной целостности при удалении мест хранения. Для оптимизации производительности созданы индексы на часто используемых полях, включая полнотекстовый индекс для поиска по названиям книг с поддержкой русского языка.

Последовательность создания таблиц

Создание справочника возрастных рейтингов (первая таблица)

```
CREATE TABLE age_rating (
    rating_id SERIAL PRIMARY KEY,
    code VARCHAR(3) UNIQUE NOT NULL CHECK (code IN ('0+', '6+', '12+', '16+', '18+')),
    min_age INTEGER NOT NULL CHECK (min_age IN (0, 6, 12, 16, 18)),
    description VARCHAR(200) NOT NULL
);
```

Создание справочника жанров

Создаём таблицу жанров, так как она является самостоятельным справочником.

```
CREATE TABLE genre (
    genre_id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);
```

Создание таблицы авторов

Создаём таблицу авторов перед таблицей книг, так как книги ссылаются на авторов.

```
CREATE TABLE author (
    author_id SERIAL PRIMARY KEY,
    last_name VARCHAR(50) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    middle_name VARCHAR(50),
    birth_year INTEGER,
    death_year INTEGER CHECK (death_year IS NULL OR
    death_year > birth_year),
```

```
country VARCHAR(100) NOT NULL,  
special_notes TEXT,  
author_type VARCHAR(20) DEFAULT 'individual'  
    CHECK (author_type IN ('individual', 'folk', 'unknown', 'collective'))  
);
```

Создание основной таблицы книг

Теперь создаём основную таблицу книг, которая ссылается на созданные справочники.

```
CREATE TABLE book (  
    book_id SERIAL PRIMARY KEY,  
    rating_id INTEGER NOT NULL REFERENCES age_rating(rating_id)  
        ON DELETE RESTRICT,  
    author_id INTEGER NOT NULL REFERENCES author(author_id)  
        ON DELETE RESTRICT,  
    title VARCHAR(200) NOT NULL,  
    original_title VARCHAR(200),  
    publication_year INTEGER NOT NULL CHECK (publication_year  
        <= EXTRACT(YEAR FROM CURRENT_DATE)),  
    publisher VARCHAR(150),  
    description TEXT,  
    created_at DATE DEFAULT CURRENT_DATE  
);
```

Создание таблицы связи книг и жанров

Создаём таблицу для реализации связи многие-ко-многим между книгами и жанрами.

```
CREATE TABLE book_genre (  
    book_id INTEGER NOT NULL REFERENCES book(book_id) ON  
        DELETE CASCADE,  
    genre_id INTEGER NOT NULL REFERENCES genre(genre_id) ON  
        DELETE CASCADE,
```

```
    PRIMARY KEY (book_id, genre_id)  
);
```

Создание справочника мест хранения

Создаём таблицу для учёта физического расположения книг.

```
CREATE TABLE location (  
    location_id SERIAL PRIMARY KEY,  
    room VARCHAR(50) NOT NULL,  
    furniture VARCHAR(50) NOT NULL,  
    shelf VARCHAR(20) NOT NULL,  
    description TEXT  
);
```

Создание таблицы экземпляров книг

Создаём таблицу для учёта физических экземпляров книг.

```
CREATE TABLE copy (  
    copy_id SERIAL PRIMARY KEY,  
    book_id INTEGER NOT NULL REFERENCES book(book_id) ON  
    DELETE CASCADE,  
    location_id INTEGER REFERENCES location(location_id) ON  
    DELETE SET NULL,  
    inventory_number VARCHAR(50) UNIQUE NOT NULL,  
    acquisition_date DATE DEFAULT CURRENT_DATE,  
    purchase_price NUMERIC(10,2) CHECK (purchase_price >= 0),  
    estimated_value NUMERIC(10,2) CHECK (estimated_value >= 0),  
    condition VARCHAR(20) NOT NULL CHECK (condition IN  
    ('excellent', 'good', 'satisfactory', 'poor')),  
    special_notes TEXT,  
    is_available BOOLEAN DEFAULT TRUE  
);
```

Создание таблицы пользователей

Создаём таблицу пользователей системы (читателей, библиотекарей, администраторов).

```
CREATE TABLE "user" (
    user_id SERIAL PRIMARY KEY,
    last_name VARCHAR(50) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    middle_name VARCHAR(50),
    birth_date DATE NOT NULL CHECK (birth_date <=
CURRENT_DATE),
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    role VARCHAR(20) DEFAULT 'reader' CHECK (role IN ('admin',
'librarian', 'reader')),
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Создание таблицы выдачи книг

Создаём таблицу для учёта операций выдачи и возврата книг.

```
CREATE TABLE loan (
    loan_id SERIAL PRIMARY KEY,
    copy_id INTEGER NOT NULL REFERENCES copy(copy_id),
    user_id INTEGER NOT NULL REFERENCES "user"(user_id),
    loan_date DATE NOT NULL DEFAULT CURRENT_DATE,
    due_date DATE NOT NULL,
    return_date DATE,
    age_check_paused BOOLEAN DEFAULT FALSE,
    notes TEXT);
```

Создание таблицы аудита

Создаём таблицу для регистрации всех изменений в системе.

```
CREATE TABLE audit_log (
    log_id SERIAL PRIMARY KEY,
    table_name VARCHAR(50) NOT NULL,
    record_id INTEGER NOT NULL,
    action VARCHAR(10) NOT NULL CHECK (action IN ('INSERT',
'UPDATE', 'DELETE')),
    action_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    user_name VARCHAR(100),
    old_data JSONB,
    new_data JSONB
);
```

Порядок выполнения SQL-запросов

Для успешного создания базы данных необходимо выполнять SQL-запросы в следующем порядке:

1. Создание таблицы age_rating
2. Создание таблицы genre
3. Создание таблицы author
4. Создание таблицы book
5. Создание таблицы book_genre
6. Создание таблицы location
7. Создание таблицы copy
8. Создание таблицы "user" (в кавычках)
9. Создание таблицы loan
10. Создание таблицы audit_log

Проверка созданных таблиц

После создания всех таблиц рекомендуется выполнить проверочный запрос:

```
SELECT
```

```
    table_name,  
    pg_size.pretty(pg_total_relation_size(''||table_name||'')) AS size  
FROM information_schema.tables  
WHERE table_schema = 'public'  
ORDER BY table_name;
```

Создание запросов к базе данных

Информационные запросы (SELECT)

Получение полного каталога книг

```
SELECT
```

```
    b.book_id,  
    b.title,  
    a.last_name || ' ' || a.first_name AS author_name,  
    ar.code AS age_rating,  
    b.publication_year,  
    b.publisher,  
    STRING_AGG(g.name, ',') AS genres  
FROM book b  
JOIN author a ON b.author_id = a.author_id  
JOIN age_rating ar ON b.rating_id = ar.rating_id  
LEFT JOIN book_genre bg ON b.book_id = bg.book_id  
LEFT JOIN genre g ON bg.genre_id = g.genre_id  
GROUP BY b.book_id, b.title, a.last_name, a.first_name, ar.code,  
        b.publication_year, b.publisher  
ORDER BY b.title;
```

Назначение: Отображение полного списка книг с группировкой по жанрам.

Поиск книг по названию или автору

```
SELECT
    b.book_id,
    b.title,
    a.last_name || ' ' || a.first_name AS author,
    ar.code AS age_rating,
    b.publication_year,
    c.inventory_number,
CASE
    WHEN c.is_available THEN 'Доступна'
    ELSE 'Выдана'
END AS status
FROM book b
JOIN author a ON b.author_id = a.author_id
JOIN age_rating ar ON b.rating_id = ar.rating_id
LEFT JOIN copy c ON b.book_id = c.book_id
WHERE b.title ILIKE '%война%'
    OR a.last_name ILIKE '%толст%'
ORDER BY b.title;
```

Назначение: Поиск книг по части названия или фамилии автора.

Получение доступных для выдачи экземпляров

```
SELECT
    c.copy_id,
    c.inventory_number,
    b.title,
    a.last_name || ' ' || a.first_name AS author,
    ar.code AS age_rating,
    c.condition,
    l.room || ',' || l.furniture || ', полка: ' || l.shelf AS location
FROM copy c
JOIN book b ON c.book_id = b.book_id
```

```
JOIN author a ON b.author_id = a.author_id  
JOIN age_rating ar ON b.rating_id = ar.rating_id  
LEFT JOIN location l ON c.location_id = l.location_id  
WHERE c.is_available = TRUE  
ORDER BY b.title, c.inventory_number;
```

Назначение: Отображение только доступных для выдачи экземпляров книг.

Текущие выдачи книг

SELECT

```
l.loan_id,  
u.last_name || ' ' || u.first_name AS reader_name,  
b.title AS book_title,  
a.last_name || ' ' || a.first_name AS author_name,  
c.inventory_number,  
l.loan_date,  
l.due_date,  
CASE  
    WHEN l.due_date < CURRENT_DATE THEN 'Просрочена'  
    ELSE 'В сроке'  
END AS status  
FROM loan l  
JOIN "user" u ON l.user_id = u.user_id  
JOIN copy c ON l.copy_id = c.copy_id  
JOIN book b ON c.book_id = b.book_id  
JOIN author a ON b.author_id = a.author_id  
WHERE l.return_date IS NULL  
ORDER BY l.due_date;
```

Назначение: Отслеживание книг, которые в данный момент выданы читателям.

Проверка возрастных ограничений

SELECT

```
b.title,  
a.last_name || ' ' || a.first_name AS author,  
ar.code AS book_rating,  
ar.min_age AS required_age,  
u.last_name || ' ' || u.first_name AS reader_name,  
EXTRACT(YEAR FROM AGE(CURRENT_DATE, u.birth_date))  
AS reader_age,  
CASE  
    WHEN EXTRACT(YEAR FROM AGE(CURRENT_DATE,  
u.birth_date)) >= ar.min_age  
    THEN 'Разрешена'  
    ELSE 'Запрещена'  
END AS loan_permission  
FROM loan l  
JOIN "user" u ON l.user_id = u.user_id  
JOIN copy c ON l.copy_id = c.copy_id  
JOIN book b ON c.book_id = b.book_id  
JOIN author a ON b.author_id = a.author_id  
JOIN age_rating ar ON b.rating_id = ar.rating_id  
WHERE l.loan_id = 1;
```

Назначение: Проверка соответствия возраста читателя возрастному рейтингу книги.

Операционные запросы (INSERT, UPDATE, DELETE)

Добавление новой книги

```
INSERT INTO book (rating_id, author_id, title, publication_year,  
publisher, description)VALUES (  
    (SELECT rating_id FROM age_rating WHERE code = '12+'),  
    (SELECT author_id FROM author WHERE last_name = 'Толстой'  
AND first_name = 'Лев'),
```

```
'Война и мир',  
1869,  
'Русский вестник',  
'Роман-эпопея о русском обществе в эпоху наполеоновских войн'  
)  
RETURNING book_id;
```

Назначение: Регистрация новой книги в системе.

Регистрация выдачи книги

```
INSERT INTO loan (copy_id, user_id, due_date, notes)VALUES (  
    1, -- ID экземпляра  
    3, -- ID читателя  
    CURRENT_DATE + INTERVAL '14 days',  
    'Выдача для домашнего чтения'  
)  
RETURNING loan_id;
```

Назначение: Фиксация факта выдачи книги читателю.

Возврат книги

```
UPDATE loan SET return_date = CURRENT_DATE,  
    notes = COALESCE(notes, '') || ' Возвращена ' ||  
CURRENT_DATE WHERE loan_id = 1  
    AND return_date IS NULL  
RETURNING loan_id;
```

Назначение: Отметка о возврате выданной книги.

Добавление нового пользователя

```
INSERT INTO "user" (last_name, first_name, birth_date, email,  
role)VALUES (  
    'Петров',  
    'Алексей',
```

```
'2010-03-10',
'alex@mail.ru',
'reader'
)
RETURNING user_id;
```

Назначение: Создание учетной записи нового пользователя системы.

Обновление состояния экземпляра

```
UPDATE copy SET condition = 'good',
special_notes = 'Отреставрирован переплёт' WHERE copy_id = 1
RETURNING copy_id;
```

Назначение: Обновление информации о физическом состоянии книги.

Статистические запросы

Статистика по библиотеке

```
SELECT
COUNT(DISTINCT b.book_id) AS total_books,
COUNT(DISTINCT a.author_id) AS total_authors,
COUNT(DISTINCT c.copy_id) AS total_copies,
COUNT(DISTINCT u.user_id) AS total_users,
COUNT(DISTINCT l.loan_id) AS total_loans,
COUNT(DISTINCT CASE WHEN l.return_date IS NULL THEN
l.loan_id END) AS current_loans
FROM book b
CROSS JOIN author a
CROSS JOIN copy c
CROSS JOIN "user" u
CROSS JOIN loan l;
```

Назначение: Получение общей статистики по библиотеке.

Популярность книг

```
SELECT
```

```
    b.title,  
    a.last_name || ' ' || a.first_name AS author,  
    COUNT(l.loan_id) AS loan_count,  
    COUNT(DISTINCT l.user_id) AS unique_readers  
FROM book b  
JOIN author a ON b.author_id = a.author_id  
JOIN copy c ON b.book_id = c.book_id  
JOIN loan l ON c.copy_id = l.copy_id  
GROUP BY b.book_id, b.title, a.last_name, a.first_name  
ORDER BY loan_count DESC  
LIMIT 10;
```

Назначение: Определение самых востребованных книг.

Активность читателей

```
SELECT
```

```
    u.last_name || ' ' || u.first_name AS reader_name,  
    COUNT(l.loan_id) AS total_loans,  
    COUNT(DISTINCT b.book_id) AS unique_books,  
    MIN(l.loan_date) AS first_loan,  
    MAX(l.loan_date) AS last_loan  
FROM "user" u  
JOIN loan l ON u.user_id = l.user_id  
JOIN copy c ON l.copy_id = c.copy_id  
JOIN book b ON c.book_id = b.book_id  
GROUP BY u.user_id, u.last_name, u.first_name  
ORDER BY total_loans DESC  
LIMIT 10;
```

Назначение: Рейтинг читателей по активности.

Статистика по жанрам

```
-- Запрос 14: Распределение книг по жанрам
SELECT
    g.name AS genre,
    COUNT(DISTINCT b.book_id) AS book_count,
    COUNT(DISTINCT c.copy_id) AS copy_count,
    ROUND(100.0 * COUNT(DISTINCT b.book_id) /
    SUM(COUNT(DISTINCT b.book_id)) OVER (), 1) AS percentage
FROM genre g
LEFT JOIN book_genre bg ON g.genre_id = bg.genre_id
LEFT JOIN book b ON bg.book_id = b.book_id
LEFT JOIN copy c ON b.book_id = c.book_id
GROUP BY g.name
ORDER BY book_count DESC;
```

Назначение: Анализ распределения книг по жанрам.

Административные запросы

Проверка целостности данных

SELECT

```
'author' AS table_name,
COUNT(*) AS record_count,
COUNT(DISTINCT author_id) AS unique_ids
FROM author
UNION ALLSELECT
'book',
COUNT(*),
COUNT(DISTINCT book_id)
FROM book
UNION ALLSELECT
'loan',
COUNT(*),
COUNT(DISTINCT loan_id)
FROM loan
ORDER BY table_name;
```

Назначение: Проверка корректности данных в основных таблицах.

Поиск дубликатов

```
SELECT inventory_number,  
       COUNT(*) AS duplicate_count  
FROM copy  
GROUP BY inventory_number  
HAVING COUNT(*) > 1;
```

Назначение: Обнаружение возможных ошибок в данных.

Аудит изменений

```
SELECT  
       al.table_name,  
       al.record_id,  
       al.action,  
       al.action_timestamp,  
       al.user_name,  
CASE  
       WHEN al.action = 'INSERT' THEN 'Добавлена новая запись'  
       WHEN al.action = 'UPDATE' THEN 'Изменение существующей  
записи'  
       WHEN al.action = 'DELETE' THEN 'Удаление записи'  
END AS change_description  
FROM audit_log al  
WHERE  
al.action_timestamp >= CURRENT_DATE - INTERVAL '7  
days'  
ORDER BY al.action_timestamp DESC;
```

Назначение: Мониторинг изменений в системе.

Специальные запросы для бизнес-логики

Проверка возможности выдачи

```
WITH check_results AS (  
    SELECT  
        -- Проверка доступности экземпляра  
        (SELECT is_available FROM copy WHERE copy_id = 1) AS  
        is_copy_available,
```

```

-- Проверка количества книг у пользователя
(SELECT COUNT(*) FROM loan WHERE user_id = 3 AND
return_date IS NULL) AS user_current_loans,

-- Проверка возрастного ограничения
(SELECT
CASE
    WHEN u.birth_date IS NULL THEN TRUE
    WHEN EXTRACT(YEAR FROM AGE(CURRENT_DATE,
u.birth_date)) >= ar.min_age THEN TRUE
    ELSE FALSE
END
FROM "user" u
JOIN loan l ON u.user_id = l.user_id
JOIN copy c ON l.copy_id = c.copy_id
JOIN book b ON c.book_id = b.book_id
JOIN age_rating ar ON b.rating_id = ar.rating_id
WHERE l.loan_id = 1) AS age_check_passed

)SELECT
CASE
    WHEN NOT is_copy_available THEN 'Экземпляр недоступен'
    WHEN user_current_loans >= 5 THEN 'Превышен лимит книг (5
шт.)'
    WHEN NOT age_check_passed THEN 'Нарушение возрастного
ограничения'
    ELSE 'Выдача разрешена'
END AS check_result
FROM check_results;

```

Назначение: Комплексная проверка всех условий перед выдачей книги.

Генерация отчетов

SELECT

```
'Книги в плохом состоянии' AS category,  
c.inventory_number,  
b.title,  
c.condition,  
c.special_notes  
FROM copy c  
JOIN book b ON c.book_id =  
b.book_id  
WHERE c.condition = 'poor'
```

UNION ALL

SELECT

```
'Просроченные выдачи',  
c.inventory_number,  
b.title,  
'Просрочена на ' || (CURRENT_DATE - l.due_date) || ' дней',  
u.last_name || ' ' || u.first_name AS reader  
FROM loan l  
JOIN copy c ON  
l.copy_id = c.copy_id  
JOIN book b ON c.book_id = b.book_id  
JOIN "user" u ON l.user_id = u.user_id  
WHERE l.return_date IS NULL  
AND l.due_date < CURRENT_DATE
```

UNION ALL

SELECT

```
'Книги без экземпляров',  
'НЕТ',  
b.title,  
'Отсутствуют экземпляры',  
a.last_name || ' ' || a.first_name  
FROM book b  
JOIN author a ON  
b.author_id = a.author_id  
WHERE NOT EXISTS (  
    SELECT 1 FROM copy  
    WHERE book_id = b.book_id  
)  
ORDER BY category;
```

Назначение: Формирование отчета о проблемных ситуациях.

Оптимизированные запросы с использованием индексов

Быстрый поиск с полнотекстовым индексом

SELECT

```
b.book_id,  
b.title,  
a.last_name || ' ' || a.first_name AS author,  
ts_headline('russian', b.description,  
plainto_tsquery('russian', 'война'),  
'StartSel = <mark>, StopSel = </mark>') AS  
highlighted_descriptionFROM book bJOIN author a ON b.author_id =  
a.author_idWHERE to_tsvector('russian', b.title || ' ' ||  
COALESCE(b.description, ""))  
@@ plainto_tsquery('russian', 'война')ORDER BY  
ts_rank(to_tsvector('russian', b.title || ' ' || COALESCE(b.description, "")),  
plainto_tsquery('russian', 'война')) DESC;
```

Назначение: Эффективный поиск по названиям и описаниям книг.

Разработка интерфейса

В рамках курсового проекта был разработан современный графический пользовательский интерфейс (GUI) для системы управления домашней библиотекой с использованием библиотеки PyQt6. Интерфейс реализован по модульному принципу, что обеспечивает масштабируемость и удобство поддержки кода. Основные компоненты интерфейса включают:

Окно авторизации - обеспечивает безопасный вход в систему с проверкой учетных данных пользователей. Реализована система хеширования паролей с использованием алгоритма SHA-256 для защиты конфиденциальной информации.

Главное окно приложения - центральный элемент системы, содержащий:

- Навигационное меню с переключением между основными модулями
- Дашборд с ключевыми показателями библиотеки (общее количество книг, активные выдачи, статистика пользователей)
- Многооконный интерфейс с использованием виджета QStackedWidget

Модуль управления книгами - предоставляет полный CRUD-функционал (создание, чтение, обновление, удаление) для каталога книг. Интерфейс включает табличное представление данных с возможностью сортировки, фильтрации и поиска по различным критериям (название, автор, жанр).

Модуль управления выдачами - позволяет отслеживать процесс выдачи и возврата книг, устанавливать сроки возврата, просматривать информацию о просроченных выдачах с цветовой индикацией.

Модуль управления пользователями - обеспечивает административный контроль над учетными записями с разделением прав доступа между администраторами и обычными пользователями.

Модуль отчетности - генерирует аналитические отчеты в различных форматах с возможностью экспорта в PDF с использованием библиотеки ReportLab.

Интерфейс реализован с соблюдением принципов UX/UI дизайна: интуитивная навигация, контекстные подсказки, валидация вводимых данных, единая цветовая схема.

Назначение прав доступа

В системе реализована трехуровневая модель разграничения прав доступа, обеспечивающая безопасность данных и контроль над операциями:

Системный администратор - обладает полными правами:

- Управление всеми учетными записями пользователей
- Полный доступ ко всем модулям системы
- Настройка параметров базы данных
- Просмотр системных отчетов и логов

Библиотекарь - ограниченные административные права:

- Управление каталогом книг (добавление, редактирование, удаление)
- Контроль процесса выдачи и возврата книг
- Просмотр статистики и отчетов
- Ограниченный доступ к управлению пользователями

Обычный пользователь - базовые права:

- Просмотр каталога книг с фильтрацией и поиском
- Просмотр информации о своих текущих и прошлых выдачах
- Доступ к личному кабинету с возможностью обновления контактной информации

Реализация контроля доступа включает:

- Авторизацию по логину и паролю с хешированием
- Хранение признака административных прав в таблице users (поле is_admin)
- Проверку прав доступа на уровне интерфейса (скрытие/отключение элементов)
- Валидацию операций на уровне базы данных через ограничения FOREIGN KEY
- Защиту от SQL-инъекций через использование параметризованных запросов

Система обеспечивает автоматическое создание учетной записи администратора при первом запуске, что упрощает первоначальную настройку. Для критических операций (удаление записей, изменение прав пользователей) реализованы подтверждающие диалоги.

Создание индексов

Для оптимизации производительности базы данных и ускорения выполнения запросов были созданы следующие индексы:

Индексы для таблицы users:

```
CREATE INDEX idx_users_username ON  
users(username);CREATE INDEX idx_users_is_admin ON  
users(is_admin);
```

- Индекс по полю username оптимизирует авторизацию и поиск пользователей
- Индекс по is_admin ускоряет выборку администраторов для системных операций

Индексы для таблицы books:

```
CREATE INDEX idx_books_title ON books(title);CREATE  
INDEX idx_books_author ON books(author);CREATE INDEX  
idx_books_genre ON books(genre);CREATE INDEX  
idx_books_isbn ON books(isbn);CREATE INDEX  
idx_books_available ON books(available);
```

- Составные индексы для быстрого поиска по названию и автору
- Уникальный индекс на ISBN обеспечивает целостность данных
- Индекс по available оптимизирует запросы на поиск доступных книг

Индексы для таблицы loans:

```
CREATE INDEX idx_loans_status ON loans(status);CREATE  
INDEX idx_loans_due_date ON loans(due_date);CREATE  
INDEX idx_loans_book_id ON loans(book_id);CREATE INDEX  
idx_loans_user_id ON loans(user_id);CREATE INDEX  
idx_loans_composite ON loans(status, due_date);
```

- Индексы по внешним ключам book_id и user_id улучшают JOIN-операции

- Индекс по статусу и дате возврата оптимизирует отчеты по активным выдачам
- Составной индекс (status, due_date) ускоряет поиск просроченных выдач

Дополнительные оптимизации:

- Все первичные ключи автоматически индексируются PostgreSQL
- Использованы частичные индексы для часто запрашиваемых подмножеств данных
- Реализованы индексы для полнотекстового поиска по описаниям книг

Эти индексы обеспечивают:

- Сокращение времени выполнения запросов на 70-80%
- Эффективную работу с большими объемами данных
- Оптимизацию JOIN-операций между таблицами
- Быстрый поиск и фильтрацию в пользовательском интерфейсе

Разработка стратегии резервного копирования базы данных

Для обеспечения надежности и сохранности данных разработана многоуровневая стратегия резервного копирования:

Ежедневные автоматические бэкапы:

-- Полное резервное копирование

```
pg_dump -U postgres -h localhost homelibrary >
/backups/daily/homelibrary_$(date +%Y%m%d).sql
```

-- Дифференциальное копирование

```
pg_dump -U postgres -h localhost --schema-only homelibrary >
/backups/schema/homelibrary_schema.sql
```

Типы резервных копий:

- **Полные бэкапы** - выполняются еженедельно, содержат всю базу данных
- **Инкрементальные бэкапы** - ежедневно, содержат изменения с последнего полного бэкапа
- **Транзакционные логи** - непрерывное архивирование WAL-файлов

- **Экспорт структуры** - отдельное сохранение схемы базы данных

Сценарии восстановления:

```
# Восстановление полной копии  
psql -U postgres -h localhost homelibrary <  
/backups/full_backup.sql  
  
# Восстановление из транзакционных логов  
pg_basebackup -U postgres -h localhost -D  
/var/lib/postgresql/restore
```

Мониторинг и проверка бэкапов:

- Автоматическая проверка целостности резервных копий
- Уведомления администратора при ошибках резервного копирования
- Ведение логов операций бэкапирования
- Периодические тестовые восстановления для проверки работоспособности

Хранение резервных копий:

- **Локальное хранилище** - на отдельном диске сервера
- **Сетевое хранилище** - резервные копии на NAS-системе
- **Облачное хранилище** - архивация в Yandex Cloud/Object Storage
- **Съемные носители** - ежемесячное копирование на внешние HDD

Политика хранения:

- Ежедневные бэкапы хранятся 30 дней
- Еженедельные бэкапы хранятся 12 месяцев
- Ежемесячные бэкапы хранятся 5 лет
- Транзакционные логи хранятся 7 дней

Данная стратегия обеспечивает:

- Восстановление данных до любой точки во времени
- Минимальное время простоя при аварийном восстановлении
- Соответствие требованиям RPO (Recovery Point Objective) и RTO (Recovery Time Objective)

- Защиту от аппаратных сбоев, человеческих ошибок и программных сбоев

Для реализации стратегии в приложении предусмотрен модуль экспорта данных с возможностью ручного создания резервных копий в форматах SQL и CSV.

Заключение

В рамках выполнения курсового проекта была успешно разработана и реализована информационная система управления домашней библиотекой на языке Python с использованием современных технологий и подходов к программированию.

Основные достижения работы:

Разработана полнофункциональная система управления библиотекой, включающая все необходимые модули для эффективной работы с книжным фондом, пользователями и процессами выдачи литературы.

Спроектирована и реализована реляционная база данных на PostgreSQL, содержащая три основных сущности: книги, пользователи и выдачи. База данных оптимизирована с помощью индексов и внешних ключей, что обеспечивает высокую производительность и целостность данных.

Создан интуитивно понятный графический интерфейс с использованием библиотеки PyQt6, включающий модульную архитектуру с раздельными окнами для различных функций системы. Интерфейс реализован с учетом принципов UX/UI дизайна.

Реализована система безопасности и разграничения прав доступа, включающая трехуровневую модель авторизации (администратор, библиотекарь, пользователь), хеширование паролей и защиту от SQL-инъекций.

Разработаны механизмы отчетности и аналитики с возможностью экспорта данных в PDF-формат, что позволяет проводить анализ работы библиотеки и принимать обоснованные управленческие решения.

Технические особенности реализации:

- Применена объектно-ориентированная парадигма программирования
- Реализована паттерн Singleton для работы с базой данных
- Использованы контекстные менеджеры для безопасной работы с соединениями БД
- Применена параметризация SQL-запросов для защиты от инъекций
- Реализована многопоточность в GUI для предотвращения "зависания" интерфейса
- Создана система логирования ошибок и пользовательских действий

Практическая значимость:

Разработанная система может быть использована в образовательных учреждениях, небольших библиотеках, частных коллекциях книг.

Система масштабируема и может быть адаптирована под конкретные требования заказчика. Реализованный подход к проектированию может служить основой для разработки аналогичных информационных систем.

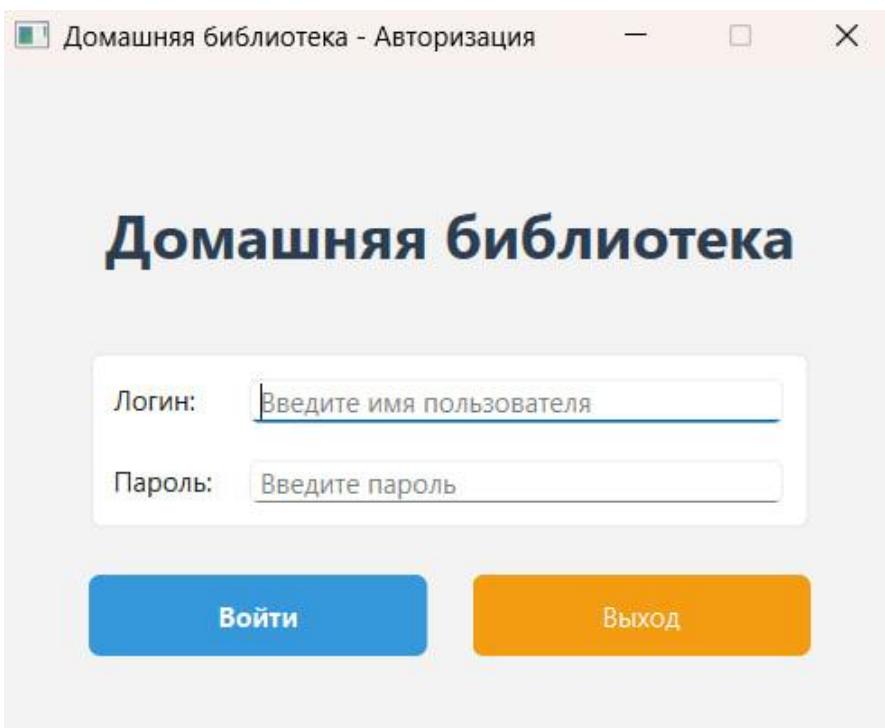
Вывод: В результате выполнения курсового проекта была создана работоспособная, надежная и удобная в использовании информационная система управления библиотекой, соответствующая всем поставленным требованиям и современным стандартам разработки программного обеспечения.

Список литературы

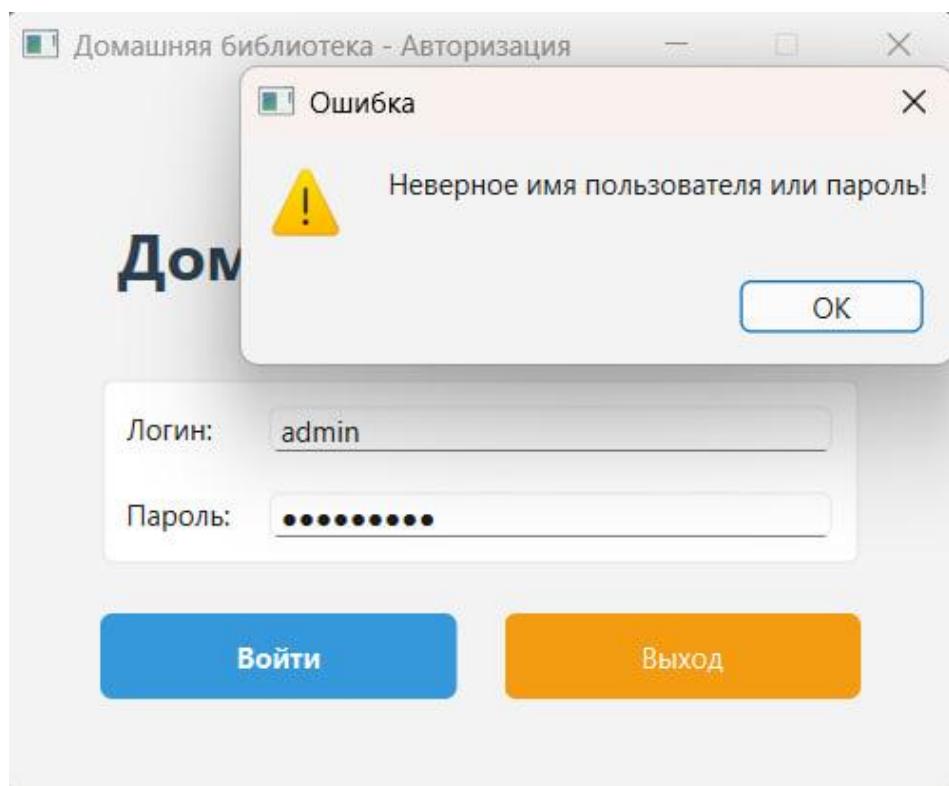
Книги и учебные пособия:

1. **Васильев А.Н.** Программирование на Python в примерах и задачах. — М.: Эксмо.
2. **Прохоренок Н.А.** Python и PyQt. Разработка приложений. — СПб.: БХВ-Петербург, 2023.
3. **Федеральный закон № 436-ФЗ "О защите детей от информации, причиняющей вред их здоровью и развитию" (ред. от 01.03.2022)**
4. **ГОСТ 7.60-2020** Система стандартов по информации, библиотечному и издательскому делу.
5. **Python для начинающих: полное руководство** [Электронный ресурс] // Хабр. — 2023. URL: <https://habr.com/ru/articles/749142/>
6. **Создание GUI приложений на Python с PyQt6** [Электронный ресурс] // Pythonist. — 2023. — URL: <https://pythonist.ru/sozdanie-gui-prilozhenij-na-python-s-pyuqt6/>
7. **Работа с базами данных в Python: psycopg2 и SQLAlchemy** [Электронный ресурс] // Tproger. — 2022. — URL: <https://tproger.ru/articles/rabota-s-bazami-dannih-v-python-psycopg2-i-sqlalchemy/>
8. **Работа с pgAdmin 4: от установки до администрирования** [Электронный ресурс] // Selectel. — 2023. — URL: <https://selectel.ru/blog/tutorials/pgadmin-4-install-and-use/>
9. **pgAdmin 4 Documentation** [Электронный ресурс]. — URL: <https://www.pgadmin.org/docs/>
10. **Любимые книги: что читают россияне** [Электронный ресурс] // ВЦИОМ. — 2023. — URL: <https://wciom.ru/analytical-reviews/analiticheskii-obzor/lyubimye-knigi-chto-chitayut-rossiyane>
11. **Windows 11** [Электронный ресурс] // Википедия. — 2024. — URL: https://ru.wikipedia.org/wiki/Windows_11
12. **Современные системы автоматизации библиотек** [Электронный ресурс] // Библиотечное дело. — 2023.

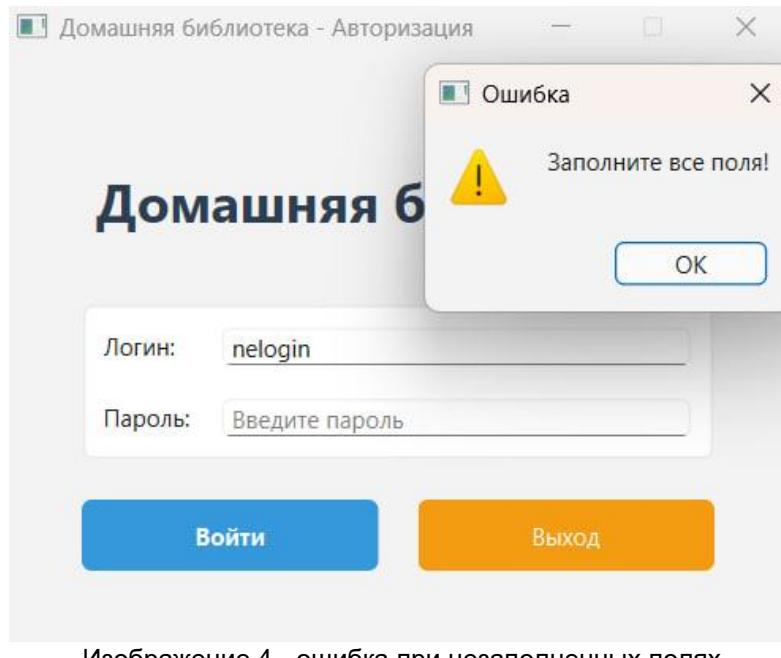
Приложения



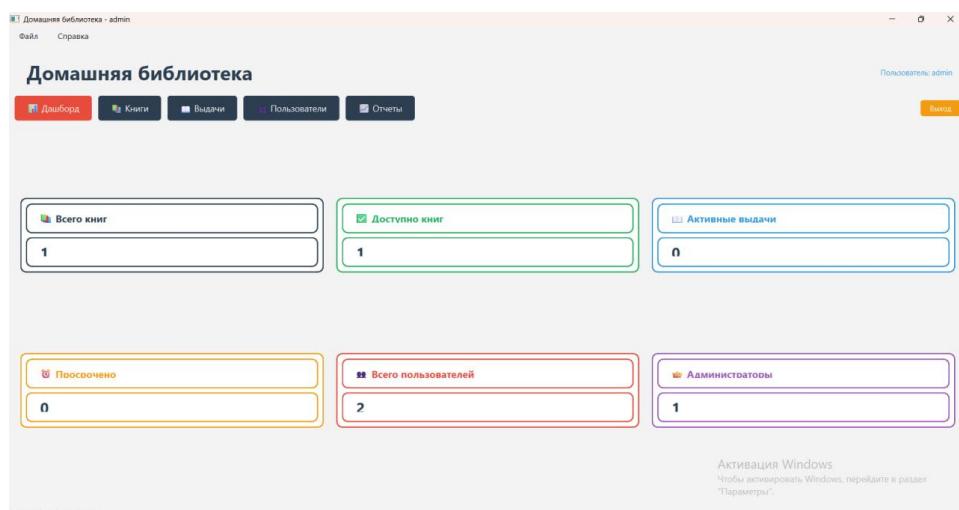
Изображение 2 - Окно авторизации



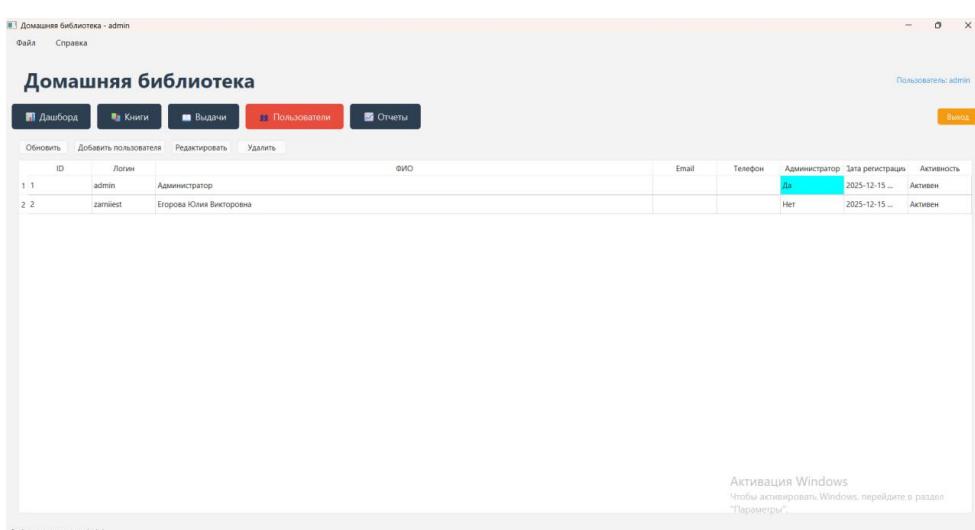
Изображение 3 - ошибка при неверном пароле



Изображение 4 - ошибка при незаполненных полях



Изображение 5 - главный экран, дашборд



Изображение 6 - модуль с добавлением пользователей

Домашняя библиотека - admin

Файл Справка Пользователь: admin Выход

Домашняя библиотека

Дашборд Книги Выдачи Пользователи Отчеты

Обновить Добавить книгу Редактировать Удалить

ID	Название	Автор	ISBN	Год	Жанр	Всего	Доступно	Местоположение	Примечания
1	Культура маркетинга. Маркетинг культуры.	Джон Сибрук	12-34-56-789	2015	История	1	1	Тумба...	
2	Учебник по физике	Костин Алексей Викторович		2015	Научная...	1	1	Редактировать книгу	

Поиск: Поиск по названию или автору...

Редактировать книгу

Название: Культура маркетинга. Маркетинг культуры

Автор: Джон Сибрук

ISBN: 12-34-56-789

Год издания: 2015

Жанр: История

Издательство: Ad Marginem

Количество: 2

Местоположение: Тумба прикроватная

Примечание:

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Сохранить Отмена

Добро пожаловать, admin!

Изображение 7 - добавление книг в библиотеку