Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Модели данных и системы управления базами данных»

|  |  |
| --- | --- |
|  | *К защите допустить:* |
|  | И. о. заведующего кафедрой информатики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. И. Сиротко |
|  |  |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**БИБЛИОТЕКА**

БГУИР КП 1-40 04 01 011 ПЗ

Студент             Е. А. Киселева

Руководитель             В. С. Плиска

Минск 2024

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc186164979)

[1 Анализ литературных источников и обзор существующих аналогов 6](#_Toc186164980)

[1.1 Анализ литературных источников 6](#_Toc186164981)

[1.2 Существующие аналоги приложения 10](#_Toc186164982)

[1.3 Требования к приложению 17](#_Toc186164983)

[2 Формирование функциональных требований и выбор       
    инструментов   разработки 20](#_Toc186083930)

[2.1 Функциональные требования к приложению 20](#_Toc186164985)

[2.2 Выбор инструментов разработки 22](#_Toc186164986)

[3 Проектирование базы данных 31](#_Toc186164987)

[3.1 Инфологическая модель 31](#_Toc186164988)

[3.2 Даталогической модель 36](#_Toc186164989)

[3.3 Создание модели миграции данных 37](#_Toc186164990)

[4 Разработка базы данных 38](#_Toc186164991)

[4.1 Создание сущностей базы данных 38](#_Toc186164992)

[4.2 Создание хранимых процедур 38](#_Toc186164993)

[4.3 Создание триггеров 42](#_Toc186164994)

[4.4 Создание функций 44](#_Toc186164995)

[4.5 Создание индексов 45](#_Toc186164996)

[5 Тестирование работоспособности базы данных 46](#_Toc186164997)

[Заключение 48](#_Toc186164998)

[Список литературных источников 49](#_Toc186164999)

[Приложение А (обязательное) Листинг кода 50](#_Toc186165000)

[Приложение Б (обязательное) Схема базы данных 56](#_Toc186165003)

[Приложение В (обязательное) Ведомость документов 57](#_Toc186165006)

****Введение****

В условиях цифровой трансформации библиотеки сталкиваются с необходимостью модернизации процессов хранения, учета и предоставления информации. Это обуславливает актуальность разработки современных информационных систем, которые обеспечивают централизованное управление данными и удобный доступ пользователей к библиотечным ресурсам.

Для эффективного функционирования сети библиотек требуется современная информационная система, которая позволяет управлять данными о книгах, читателях и их взаимодействии с библиотекой.

Целью курсового проекта является разработка базы данных (БД) согласно выбранной теме. К задачам курсового проекта относятся:

– определить сущности проектируемой БД и их связи;

– нормализовать БД до третьей нормальной формы;

– создать пул запросов;

– создать индексы для наиболее часто используемых сущностей;

– создать пул триггеров и хранимых процедур для работы с БД.

С учетом выбранной темы необходимо разработать базу данных для веб-приложения сети библиотек, которая обеспечит централизованное управление данными и автоматизацию ключевых процессов.

Для достижения поставленной цели в работе решаются следующие задачи:

1 Проведение анализа литературных источников и обзор существующих аналогов информационных систем для библиотек, выявление их преимуществ и недостатков, формулировка требований к разрабатываемой системе.

2 Определение функциональных требований к базе данных и выбор оптимальной системы управления базами данных на основе анализа доступных инструментов и их характеристик.

3 Разработка проектной документации, включая инфологическую и даталогическую модели, модель миграции данных, IDEF-диаграммы и описание форматов хранения данных.

4 Создание физической базы данных, включая реализацию скриптов для её построения, настройку индексов, триггеров, процедур, функций и других элементов, а также, при необходимости, разработку ORM.

5 Проведение тестирования базы данных, включая проверку на корректность ограничений, тесты на аномалии, проверку работы процедур, триггеров и других компонентов системы.

В результате выполнения поставленных задач будет создана база данных, которая обеспечит надежное хранение данных, эффективный поиск и управление библиотечными ресурсами, а также удовлетворит потребности пользователей сети библиотек.

Курсовая записка составлена в соответствии со стандартом предприятия СТП 01-2024. [1]

# Анализ литературных источников и обзор             существующих аналогов

Для разработки базы данных, которая станет основой веб-приложения сети библиотек, важно проанализировать существующие решения, обеспечивающие управление библиотечными ресурсами через веб-приложения. На этом этапе рассматриваются современные подходы к созданию сайтов для библиотек, анализируются их функциональные возможности, архитектура и используемые технологии.

Целью данного анализа является выявление сильных и слабых сторон существующих аналогов, что позволит сформулировать обоснованные требования к создаваемому сайту. На основе этих требований впоследствии будут определены задачи базы данных, включая структуру хранения данных, функционал обработки запросов и поддержку пользовательских операций.

Результатом данного этапа станет перечень функциональных требований к веб-приложению сети библиотек, который станет отправной точкой для проектирования базы данных и дальнейшей реализации системы.

## Анализ литературных источников

Системы управления данными в библиотеках предоставляют возможность автоматизировать ключевые процессы, повышая доступность ресурсов и удобство их использования для читателей. Они также играют важную роль в обеспечении эффективного управления фондами и взаимодействия с пользователями через цифровые платформы.

Можно выделить следующие преимущества использования систем управления базами данных в библиотеках:

1 Централизованное хранение данных: предоставление централизованного хранилища информации о библиотечных фондах, читателях и услугах, что позволяет избежать дублирования данных, обеспечивает их согласованность и упрощает доступ к информации.

2 Автоматизация процессов: системы управления данными помогают автоматизировать рутинные операции, такие как учет книг, регистрация читателей, обработка запросов на бронирование и продление сроков сдачи. Это сокращает время выполнения задач и снижает нагрузку на персонал.

3 Гибкость и масштабируемость: адаптация к изменениям в потребностях библиотеки, включая увеличение объемов данных, добавление новых функций и интеграцию с внешними сервисами.

4 Улучшение пользовательского опыта: обеспечение доступа к информации о книгах, услугах и мероприятиях в режиме онлайн, что делает работу с библиотекой более удобной для пользователей. Например, они могут бронировать книги, просматривать свои читательские записи и получать уведомления о сроках возврата.

5 Круглосуточный доступ: цифровые платформы позволяют пользователям получать доступ к библиотечным ресурсам в любое время и из любого места, что значительно повышает удобство использования.

Системы управления данными в библиотеках необходимы для веб-приложений библиотек, а тем более для библиотек, объединенных в целую сеть, так как помогают автоматизировать процессы, лучше структурировать информацию и предоставить пользователям больше возможностей.

База данных является основой таких систем, обеспечивая хранение и обработку больших объемов информации. Это систематизированное хранилище данных, структурированное таким образом, чтобы они могли быть легко найдены и обработаны. База данных является неотъемлемой частью такой системы, так как с помощью нее можно автоматизировать множество процессов.

База данных – это представленные в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью ЭВМ.

В современных приложениях работа с базой данных ведется посредством готового программного обеспечения, которое содержит в себе все необходимые инструменты и компоненты для успешного взаимодействия с хранилищем данных. Такое программное обеспечение называется системой управления базой данных (СУБД). Основными функциями СУБД являются:

– создание баз данных, изменение, удаление и объединение их по определенным признакам;

– доступ к данным и хранение данных, в том числе больших массивов, в структурированном виде и нужном формате;

– защита данных от взлома и нежелательных изменений при помощи распределенного доступа;

– выгрузка и сортировка данных по заданным фильтрам при помощи SQL-запросов;

– поддержка целостности баз данных, резервное копирование и восстановление после сбоев. [2]

СУБД классифицируются на разные типы в зависимости от моделей используемых данных, по степени распределенности и по способу доступа к базе данных. Классифицируются СУБД по следующим признакам:

– модель данных (иерархические, сетевые, реляционные, объектно-ориентированные, объектно-реляционные);

– степени распределенности (локальные, распределенные);

– способ доступа к БД (файл-серверные, клиент-серверные, встраиваемые). [3]

Базы данных имеют два уровня представления: логическое и физическое.

Физическое представление отражает способ хранения данных на носителях, таких как жесткие диски, и включает такие аспекты, как размещение файлов, индексация и кэширование. Физическое представление базы данных состоит из физических объектов, таких как файлы, блоки, страницы и сегменты. Эти объекты определяют способ хранения, доступа и обработки данных на физическом носителе, таком как диск, память или сеть.

Физическое представление базы данных может быть описано с помощью физической схемы, которая показывает расположение и размер физических объектов, параметры и настройки, которые влияют на производительность и эффективность базы данных, также оно может быть оптимизировано с помощью различных методов и техник, таких как сжатие, шардирование, кэширование, индексирование и т.д.

Логическое представление базы данных – это абстрактный уровень, на котором данные описываются с точки зрения их структуры и связей, без учета деталей их физического хранения. Оно определяет, как данные организованы и связаны между собой, используя понятные пользователям концепции, такие как таблицы, атрибуты и связи.

Основу логического представления составляют таблицы, каждая из которых отражает конкретный объект или сущность. Строки таблицы представляют записи, а столбцы – их характеристики, называемые атрибутами. Важным элементом является первичный ключ, который обеспечивает уникальную идентификацию записей, а также внешние ключи, которые устанавливают связи между таблицами. Такие связи позволяют моделировать отношения между объектами.

Для обеспечения корректности данных логическое представление включает ограничения, такие как уникальность значений, обязательность заполнения полей и поддержание ссылочной целостности. Дополнительно используются индексы для ускорения поиска и представления (виртуальные таблицы), которые упрощают доступ к сложным данным.

Логическое представление базы данных может быть описано с помощью концептуальной схемы, которая показывает все элементы данных и их связи в виде графической диаграммы. Логическое представление также разделяется на несколько уровней абстракции: внешний, логический и внутренний.

На внешнем уровне данные предоставляются в удобной форме для конкретных пользователей или приложения. Логический уровень представляет данные в формате, понятном всем пользователям и приложениям. Внутренний уровень представляет данные в формате, понятном системе управления базой данных.

Главное преимущество логического представления заключается в том, что оно делает данные понятными для пользователей и приложений, обеспечивая при этом гибкость изменений. Логическая структура данных может быть изменена без влияния на физическое хранение, что упрощает управление базой данных и адаптацию к новым требованиям.

Разделение на логическое и физическое представления необходимо для обеспечения гибкости, удобства работы с данными и независимости уровней проектирования. Логическое представление позволяет сосредоточиться на структуре и связях данных, понятных пользователям и приложениям, в то время как физическое – оптимизирует хранение и обработку данных на уровне оборудования. Такое разделение упрощает управление базой данных, позволяет изменять физическую организацию без влияния на бизнес-логику и облегчает адаптацию системы к новым требованиям. [4]

В процессе работы с базой данных могут возникать аномалии, представляющие собой ошибки, связанные с несоответствием структуры данных требованиям модели предметной области. Аномалии затрудняют управление данными и могут приводить к потере целостности базы. Основные виды аномалий включают:

1 Аномалии вставки. Такие аномалии возникают, когда добавление новой записи в таблицу становится невозможным из-за отсутствия части данных. Это приводит к необходимости либо оставлять поля пустыми, либо заполнять их фиктивными значениями, либо вовсе отказываться от добавления записи.

2 Аномалии удаления. Эти аномалии проявляются, когда удаление ненужной записи приводит к потере важных данных, все еще нужных приложению.

3 Аномалии обновления. Такие ошибки возникают, если изменение данных в одном месте не обновляет связанные данные в других местах таблицы, что приводит к несоответствию информации.

Для устранения подобных аномалий используется процесс нормализации базы данных. Нормализация включает группировку и(или) распределение атрибутов по разным таблицам и создание четких связей между ними. Этот подход минимизирует избыточность, предотвращает ошибки и обеспечивает эффективное хранение и обработку данных.

Нормализация отношений осуществляется путем последовательного прохождения нормальных форм, каждая из которых накладывает свои требования на структуру таблиц.

Для соответствия первой нормальной форме таблица должна содержать только атомарные (неделимые) значения. Все столбцы должны содержать однотипные данные, а строки быть уникальными. Например, если в библиотечной базе данных в одном поле хранятся несколько авторов книги, это нарушает первую нормальную форму. В таком случае необходимо создать отдельную таблицу для авторов и связать её с книгами.

Таблица находится во второй нормальной форме, если она уже приведена к первой нормальной форме и все неключевые атрибуты зависят от всего первичного ключа, а не только от его части. Это требование особенно важно для таблиц с составным ключом. Например, если информация об издателе книги зависит только от её идентификатора, а не от экземпляра, необходимо вынести данные об издателе в отдельную таблицу.

Таблица соответствует третьей нормальной форме, если она находится во второй нормальной форме и все её неключевые атрибуты зависят исключительно от первичного ключа. Это исключает транзитивные зависимости. Например, если в таблице книг хранится адрес издателя, его нужно вынести в отдельную таблицу, связав через идентификатор издателя.

Помимо третьей нормальной формы специалисты выделяют усиленную разновидность третьей нормальной формы – нормальную форму Бойса-Кодда. Эта форма усиливает требования третьей нормальной формы, устраняя ситуации, когда в таблице могут существовать атрибуты, не зависящие от потенциальных ключей. В библиотечной базе данных это может проявляться в случае, если информация о жанре книги хранится как отдельное поле, но не имеет явной связи с идентификатором книги.

Таблица находится в четвертой нормальной форме, если она соответствует нормальной форме Бойса-Кода и не содержит многозначных зависимостей. Например, если один автор пишет книги разных жанров, а таблица объединяет их в одном поле, это нарушает четвертую нормальную форму. Данные должны быть разнесены в отдельные таблицы.

Таблица соответствует пятой нормальной форме, если её данные не могут быть разделены на подмножества без утраты информации. Это требование редко применяется на практике, но полезно для сложных моделей с многомерными зависимостями. Для обычного же разработчика БД пятая нормальная форма представляет скорее теоретический, нежели практический интерес.

Таким образом, нормализация данных, осуществляющаяся на этапе логического проектирования БД, позволяет структурировать базу данных таким образом, чтобы избежать избыточности, аномалий и ошибок. Каждая нормальная форма помогает улучшить качество структуры, делая базу данных более удобной и эффективной для использования и поддержки. Однако на практике для построения приемлемой логической модели БД необходимо пройти только три или четыре ступени нормальных форм:

- все поля в таблицах неделимы и не содержат повторяющихся групп

- все неключевые поля в таблицах зависят от первичного ключа

- в таблицах отсутствуют избыточные не ключевые поля

- в таблицах устранены многозначные зависимости

Это позволяет устранить основные проблемы структуры данных, такие как избыточность, аномалии вставки, удаления и обновления. Эти этапы обеспечивают целостность и оптимизацию данных без излишнего усложнения модели. Более высокие нормальные формы, как правило, применяются реже, так как их требования становятся избыточными для большинства практических задач и могут приводить к чрезмерному усложнению структуры базы данных. [5]

Таким образом, проектирование базы данных выступает одним из ключевых этапов в планировании и разработке приложения, наряду с выбором платформы и архитектуры. От грамотно спроектированной структуры базы данных зависит не только эффективность хранения и обработки данных, но и стабильность работы всей системы. Продуманная база данных позволяет минимизировать избыточность, предотвратить возникновение аномалий, обеспечить целостность данных и оптимизировать взаимодействие с ними. Все это делает проектирование базы данных основополагающим шагом для создания надежного и масштабируемого приложения.

## Существующие аналоги приложения

Для анализа существующих аналогов веб-приложений библиотек были рассмотрены несколько популярных решений, используемых в различных библиотеках. Эти сайты предоставляют функционал для управления ресурсами, взаимодействия с пользователями и автоматизации основных библиотечных процессов.

**1.2.1** Веб-приложение национальной библиотеки Республики Беларусь

Веб-приложение Национальной библиотеки Беларуси является крупнейшим информационным ресурсом страны, предоставляющим доступ к разнообразным физическим и электронным библиотечным материалам. Он выполняет роль центрального узла для всех категорий пользователей, включая студентов, исследователей, преподавателей и широкую общественность. Графический интерфейс главной страницы сайта национальной библиотеки Республики Беларусь представлен на рисунке 1.1.

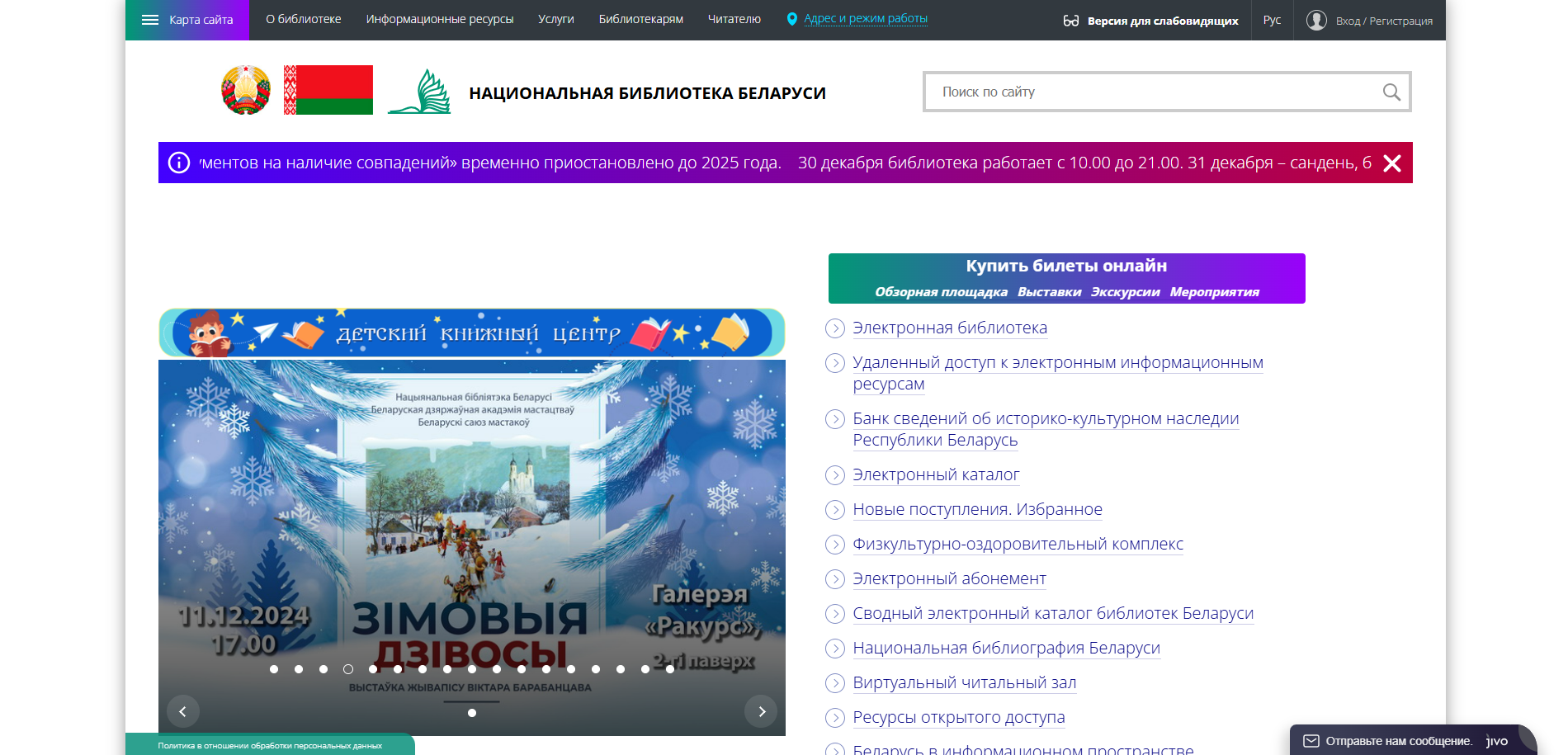


Рисунок 1.1 – Графический интерфейс сайта национальной библиотеки Республики Беларусь

Основной функционал сайта включает поиск по электронному каталогу, который охватывает обширный фонд библиотеки. Каталог поддерживает расширенные фильтры, такие как поиск по автору, названию, ключевым словам, месту издания, языку и другим параметрам. Это позволяет пользователям быстро находить необходимые книги, статьи и другие материалы. Особое внимание уделяется электронным ресурсам – на сайте можно получить доступ к сканированным версиям редких книг, научным журналам, краеведческим материалам и архивным документам. Электронная библиотека сайта национальной библиотеки Республики Беларусь представлена на рисунке 1.2.

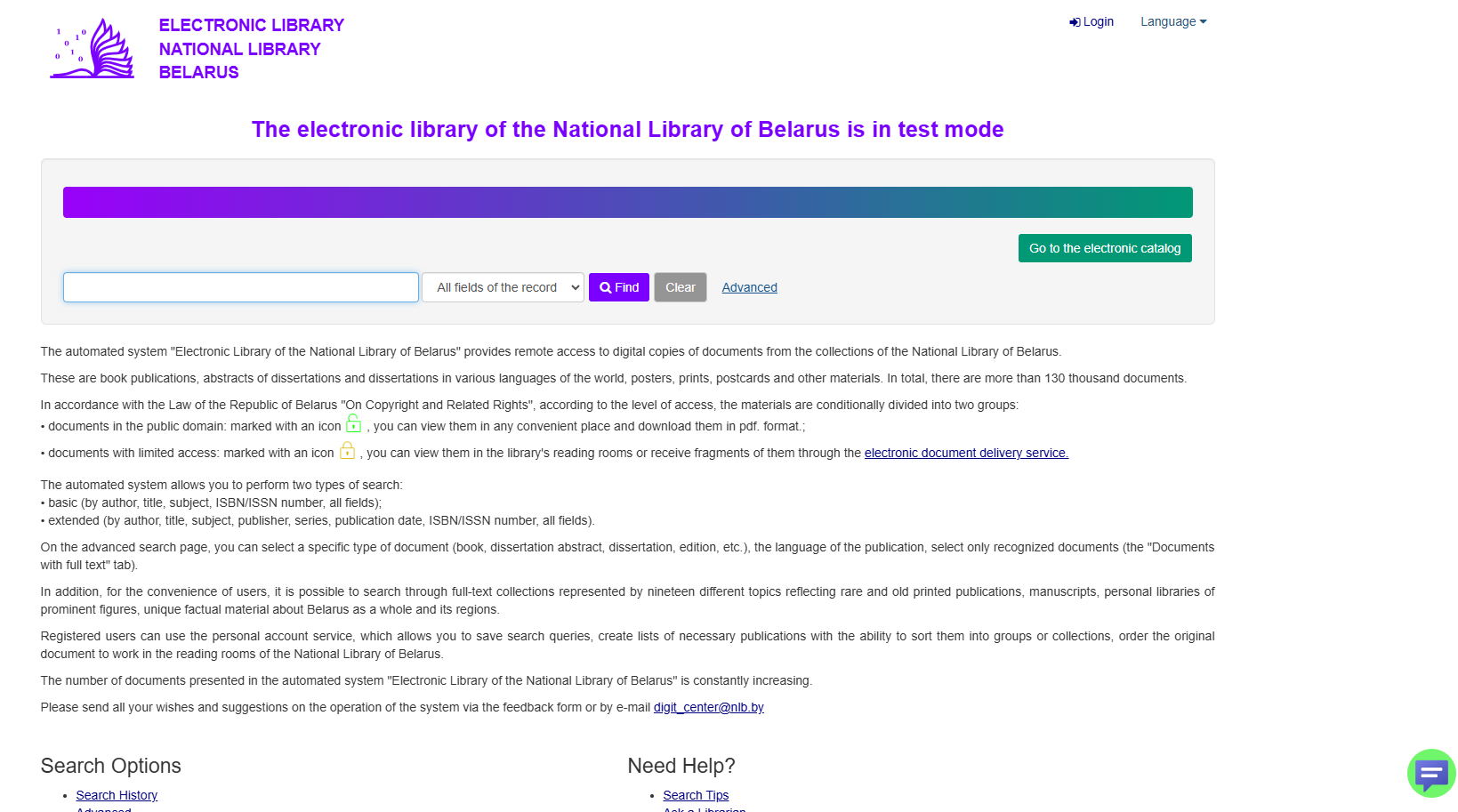


Рисунок 1.2 – Электронная библиотека сайта национальной библиотеки Республики Беларусь

Для удобства пользователей предусмотрен личный кабинет, который позволяет сохранять историю поиска, добавлять материалы в избранное, а также отслеживать состояние выданных книг. Однако функция бронирования книг и управления читательским билетом через сайт реализована ограниченно, что требует посещения библиотеки для выполнения данных операций. Личный кабинет сайта национальной библиотеки Республики Беларусь представлен на рисунке 1.3.

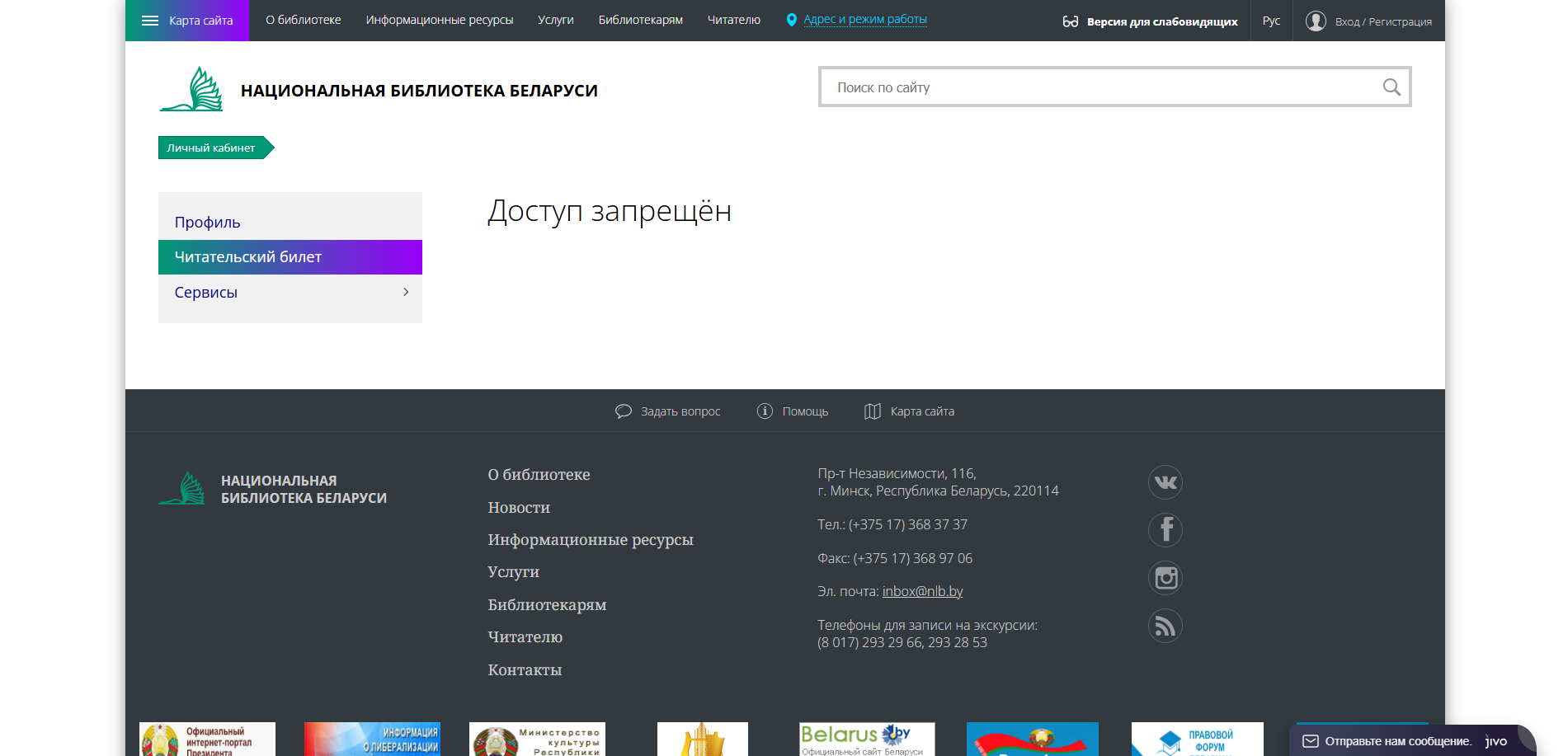


Рисунок 1.3 – Личный кабинет сайта национальной библиотеки Республики Беларусь

Веб-приложение также предлагает множество дополнительных сервисов, таких как расписание мероприятий, возможность заказа экскурсий и получение справочной информации о библиотеке. Особенностью Национальной библиотеки является акцент на научные исследования и культурное развитие, что отражено в разделе научных публикаций и мультимедийных материалов.

Несмотря на широкий функционал, у веб-приложения есть определенные ограничения. Во-первых, онлайн-сервисы для работы с физическими книгами, такие как бронирование или продление срока сдачи, недостаточно развиты. Во-вторых, интерфейс сайта может быть сложным для некоторых категорий пользователей из-за большого количества разделов и функций.

На основании анализа сайта Национальной библиотеки Беларуси можно сделать вывод, что он выполняет роль важного информационного ресурса, обеспечивая доступ к обширным коллекциям данных. Однако для увеличения удобства и интерактивности системы стоит развивать функции управления физическими ресурсами и упрощать взаимодействие пользователей с библиотечными сервисами через интернет. [6]

**1.2.2** Веб-приложение библиотеки Library Genesis

«Library Genesis» – это онлайн-хранилище и поисковая система, которая предоставляет бесплатный доступ к миллионам книг, научных статей и других публикаций. Это ресурс, который широко используется студентами, исследователями и любителями литературы для поиска и скачивания материалов. На платформе можно найти не только научные работы и учебники, но и художественную литературу. Графический интерфейс главной страницы сайта «Library Genesis» представлен на рисунке 1.4.



Рисунок 1.4 – Графический интерфейс сайта «Library Genesis»

Один из основных плюсов «Library Genesis» – это широкий ассортимент материалов, доступных для скачивания без необходимости платить за них, что позволяет пользователям без ограничений получать необходимые книги и статьи, что способствует распространению знаний и поддерживает концепцию открытой науки.

Другим важным преимуществом является удобный механизм поиска. Платформа предоставляет возможность искать материалы по различным критериям, таким как название, автор, ISBN и другие параметры. Это делает поиск нужных публикаций быстрым и эффективным. В дополнение, доступность материалов без платы привлекает пользователей, не имеющих финансовых возможностей для покупки дорогих учебников или научных журналов.

Однако, несмотря на все эти преимущества, у «Library Genesis» есть и значительные минусы. Основным из них является нарушение авторских прав. Платформа предоставляет доступ к материалам, зачастую без разрешения правообладателей, что может привести к юридическим последствиям для пользователей в странах с жестким законодательством в области авторских прав. Кроме того, основной сайт «Library Genesis» периодически блокируется, и пользователи вынуждены искать зеркала или прокси-сайты для получения доступа к ресурсам, что делает доступ к платформе нестабильным.

Кроме того, существует риск безопасности. Некоторые зеркала LibGen могут быть ненадежными, содержать вредоносное ПО или фишинговые элементы, что требует от пользователей особой осторожности при использовании сайта. Несмотря на это, для многих людей LibGen остается незаменимым инструментом для доступа к научной и образовательной информации.

В целом, LibGen является ценным ресурсом для тех, кто ищет бесплатный доступ к книгам и статьям, однако следует учитывать правовые и безопасностные риски, связанные с его использованием. [7]

**1.2.3** Веб-приложение библиотеки БГУИР

Библиотека Белорусского государственного университета информатики и радиоэлектроники предоставляет студентам, преподавателям и исследователям доступ к обширным информационным ресурсам. Её официальный сайт предлагает разнообразные услуги и инструменты для эффективного поиска и использования материалов. Графический интерфейс главной страницы сайта библиотеки БГУИР представлен на рисунке 1.5.

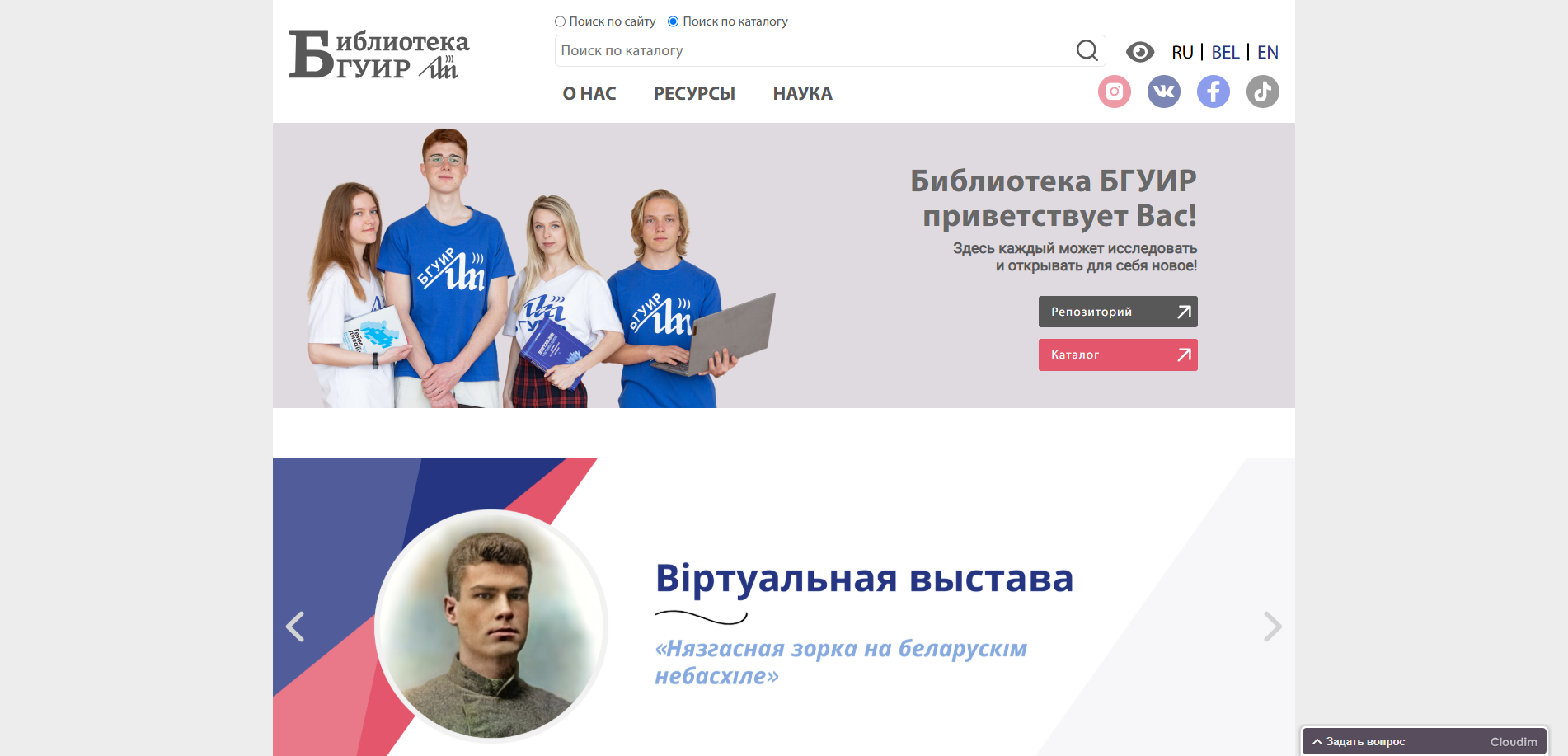


Рисунок 1.5 – Графический интерфейс сайта библиотеки БГУИР

Центральным элементом онлайн-сервиса является электронный каталог. Он включает информацию о книгах, брошюрах, диссертациях, авторефератах, отчетах о научно-исследовательских работах и других изданиях, находящихся в фонде библиотеки. Пользователи могут осуществлять поиск по различным критериям, таким как автор, название, издательство, а также использовать расширенные и сквозные поисковые функции. Для доступа к полным текстам и дополнительным функциям требуется вход в личный кабинет с использованием фамилии и номера читательского билета.

Библиотека БГУИР расположена в четырёх учебных корпусах университета в Минске и включает семь читальных залов, три абонемента и 121 компьютерное рабочее место. Предоставляется доступ к беспроводной сети Wi-Fi, что обеспечивает удобство работы с электронными ресурсами.

Сайт библиотеки предоставляет доступ к электронным книгам, статьям, стандартам и другим цифровым материалам. С сентября 2023 года электронный каталог интегрирован с Электронной библиотечной системой «Айбукс», что расширяет доступ к дополнительным электронным ресурсам.

Пользователи могут заказать книги, получить информацию о выданных и заказанных экземплярах, а также ознакомиться с новыми поступлениями в библиотеку. Для получения помощи и обратной связи предусмотрены соответствующие разделы на сайте.

Таким образом, сайт библиотеки БГУИР предоставляет комплексный доступ к информационным ресурсам университета, сочетая физические и электронные материалы для удовлетворения образовательных и исследовательских потребностей пользователей. [8]

**1.2.3** Веб-приложение «WorldCat»

«WorldCat» – это крупнейшая международная база данных библиотечных записей, охватывающая более 2 миллиардов записей из более чем 10 000 библиотек по всему миру. Платформа предоставляет пользователям возможность искать книги, журналы, статьи, аудиоматериалы и другие ресурсы, доступные в библиотеках. Графический интерфейс главной страницы сайта «WorldCat» представлен на рисунке 1.6.

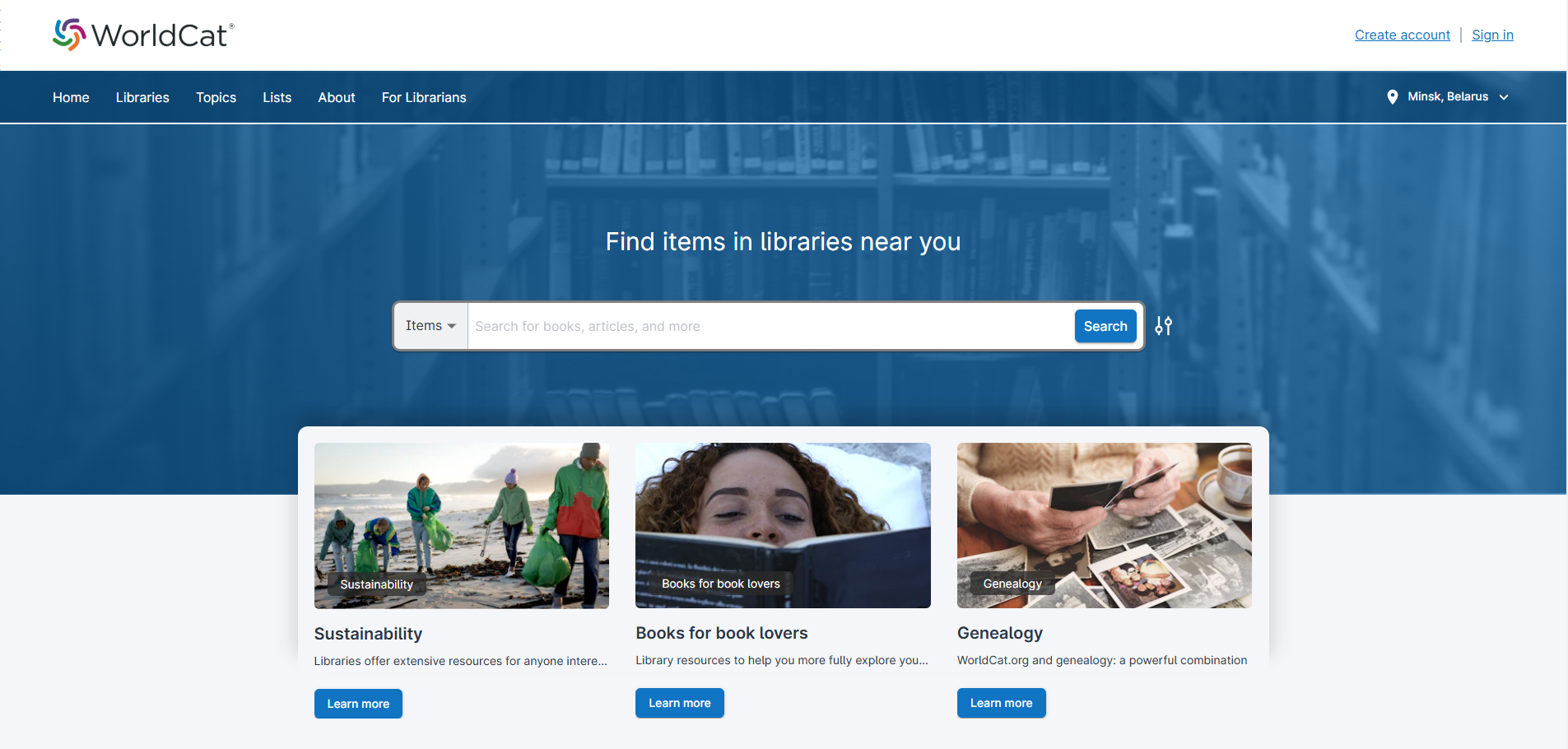


Рисунок 1.6 – Графический интерфейс сайта «WorldCat»

Одним из главных преимуществ «WorldCat» является его обширная база данных, которая делает его одним из самых полных источников информации о библиотечном контенте. Она охватывает не только общедоступные книги, но и редкие и специализированные материалы, что позволяет найти ресурсы, которые могут быть недоступны в других библиотеках.

«WorldCat» предлагает удобный поиск, позволяет искать материалы по ключевым словам, автору, названию и другим критериям. Этот гибкий подход позволяет быстро и точно находить необходимые ресурсы. Платформа также интегрирована с многими библиотечными сервисами, такими как межбиблиотечный абонемент, что дает возможность пользователям заказывать книги и другие материалы из различных библиотек. Кроме того, «WorldCat» поддерживает несколько языков, что делает его доступным для пользователей по всему миру, и предоставляет доступ к электронным книгам и статьям, что удобно для студентов и исследователей. Поиск на сайте «WorldCat» представлен на рисунке 1.7.

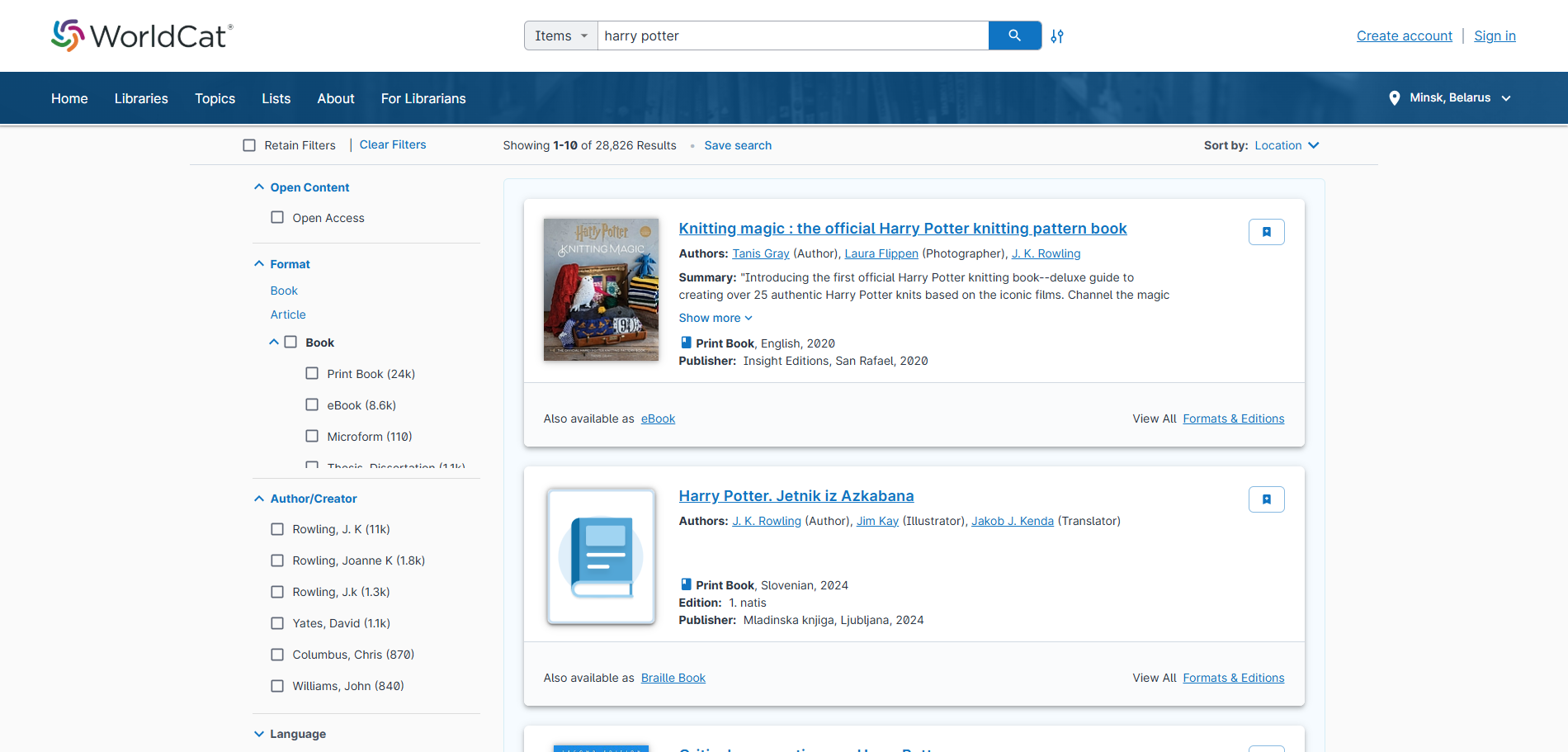


Рисунок 1.7 – Поиск на сайте «WorldCat»

Однако, несмотря на все преимущества, у «WorldCat» есть и недостатки. Один из них заключается в том, что платформа не всегда предоставляет доступ к полным текстам материалов, особенно к статьям и книгам, которые находятся в платных коллекциях. Хотя «WorldCat» предоставляет ссылки на эти ресурсы, доступ к полным текстам часто ограничен. Еще одним минусом является сложность интерфейса для новых пользователей. Несмотря на свою функциональность, интерфейс «WorldCat» может быть трудным для освоения, особенно при поиске конкретных материалов или заказе через межбиблиотечный абонемент.

Кроме того, доступность материалов зависит от библиотек в определенных географических регионах, что может быть неудобно для пользователей, находящихся далеко от крупных библиотечных центров. «WorldCat» также не предоставляет прямого доступа к материалам, как некоторые онлайн-магазины или электронные библиотеки, а полагается на локальные библиотеки для выдачи ресурсов. Еще одним минусом является ограниченная мобильная версия платформы, которая не всегда так удобна для пользователей, как версия для ПК.

Тем не менее, «WorldCat» остается мощным инструментом для поиска библиотечных ресурсов по всему миру. Он является незаменимым для исследователей, студентов и библиографов, нуждающихся в обширном поиске информации, несмотря на некоторые ограничения по доступу и интерфейсу.

При разработке собственного веб-приложения для сети библиотек необходимо учесть сильные стороны существующих аналогов, такие как удобство поиска, возможность заказов через межбиблиотечный абонемент и поддержку различных форматов контента. В то же время, стоит обратить внимание на недостатки – ограничения доступа к полным текстам, сложности с интерфейсом и ограниченную мобильную доступность. Учитывая эти аспекты, необходимо будет разработать систему, которая будет максимально удобной для пользователей, обеспечивать доступ к библиотечным ресурсам в онлайн-формате, а также реализовать механизмы для эффективного поиска, заказа и доступа к материалам. [9]

## Требования к приложению

Эффективная работа сети библиотек невозможна без современного веб-приложения, которое автоматизирует ключевые процессы и обеспечивает удобный доступ к ресурсам для читателей и сотрудников. Для создания такой системы необходимо четко определить требования к её функциональности, интерфейсу и безопасности. Далее будут подробно рассмотрены основные требования, охватывающие все аспекты управления сетью библиотек.

Любое современное веб-приложение должно соответствовать стандартам безопасности, доступности и удобства использования. Общие требования определяют ключевые технические и пользовательские характеристики системы, обеспечивая ее надежность и совместимость с различными устройствами. Общие требования включают:

1 Целевая аудитория: библиотекари, читатели, администраторы.

2 Поддержка платформ: доступ с персональных компьютеров, планшетов и мобильных устройств.

3 Интерфейс: интуитивно понятный пользовательский интерфейс, поддержка нескольких языков.

4 Аутентификация и авторизация: система ролей (читатель, библиотекарь).

5 Безопасность данных: шифрование данных, регулярное резервное копирование.

Книги являются центральным элементом любой библиотеки. Эффективное управление книгами и связанными с ними данными (авторы, жанры, категории, теги) обеспечивает высокую доступность и удобство поиска для пользователей. Управление книгами и авторами включает в себя:

1 Добавление, редактирование и удаление книг для библиотекарей.

2 Привязка книг к авторам, жанрам, категориям, тегам и языкам.

3 Учет форматов книг (например, печатные, электронные).

4 Возможность добавления экземпляров книг с уникальными идентификаторами, просмотра в каком количестве и где находится экземпляр, а также информации об экземпляре (издатель, год издания).

5 Просмотр информации об авторах (даты рождения, страна).

Эффективная система управления читателями и их библиотечными картами упрощает обслуживание и контроль доступа к ресурсам библиотеки. Эта система включает в себя:

1 Регистрацию и учет читателей.

2 Выпуск, продление и блокировку библиотечных карт.

3 Привязку библиотечной карты к читателю.

4 Управление привилегиями пользователей (например, премиум доступ к книгам для читателей).

Система заимствований обеспечивает контроль над движением книг и соблюдением сроков возврата. Автоматизация этого процесса позволяет снизить ошибки и улучшить качество обслуживания читателей. В систему заимствований входят:

1 Возможность бронирования и выдачи книг.

2 Отслеживание сроков возврата книг.

3 Учет просроченных возвратов и начисление штрафов.

4 Отображение истории заимствований для читателей и библиотекарей.

Сеть библиотек часто включает несколько филиалов и читальных залов. Эффективное управление пространством и ресурсами помогает оптимизировать доступ к книгам и рабочим зонам. Для этого необходимы:

1 Учет филиалов библиотек.

2 Управление читальными залами: название, время работы, вместимость.

3 Система позиций книг: этаж, стеллаж, полка.

4 Отображение доступности книг в разных филиалах.

Отзывы и рейтинги помогают библиотекам лучше понимать предпочтения читателей и улучшать качество своих услуг. Возможность оставлять комментарии и оценки способствует повышению вовлеченности пользователей. Для этого должны быть:

1 Возможность читателям оставлять отзывы о книгах.

2 Система рейтингов (оценка книг по шкале).

3 Модерация отзывов библиотекарем.

Система подписок позволяет читателям получать дополнительные услуги и доступ к эксклюзивным материалам. Управление подписками помогает контролировать статусы и уведомления и включает в себя:

1 Возможность оформления подписок на дополнительные услуги.

2 Управление сроками действия подписок.

3 Система уведомлений о завершении подписки.

Поиск является одной из важнейших функций любой библиотеки, так как библиотека – это источник информации, как развлекательной, так и образовательной или и вовсе необходимой для работы. Эффективные инструменты поиска и фильтрации позволяют читателям и библиотекарям быстро находить нужную информацию.

1 Удобные поиск и сортировка книг по названию, автору, жанру, категории, тегам, тематике, языку.

2 Фильтрация доступных книг по филиалам сети библиотек.

3 Для библиотекарей – поиск читателей по номеру библиотечной карты и личным данным.

Идентификационные карты обеспечивают быстрый и безопасный доступ к услугам библиотеки. Система идентификации повышает уровень защиты персональных данных и включает в себя:

1 Возможность использования идентификационных карт для аутентификации.

2 Хранение персональных номеров и дат выдачи карт.

3 Идентификация пользователей и всей информации о них с помощью читательских билетов.

Веб-приложение для сети библиотек должно быть гибким, надежным и масштабируемым. Реализация описанных требований к веб-приложению позволит создать современную систему, которая не только улучшит качество обслуживания читателей и оптимизирует рабочие процессы библиотекарей, но и станет мощным инструментом для аналитики и управления библиотечным фондом. Система сможет предоставлять подробные отчеты об использовании ресурсов, популярности различных жанров и авторов, а также о востребованности отдельных книг, что позволит библиотекам адаптировать свои фонды под текущие запросы читателей. Кроме того, веб-приложение должно обеспечивать возможность интеграции с внешними ресурсами, такими как электронные библиотеки и книжные магазины, расширяя тем самым доступный читателям контент. Развитие системы должно быть непрерывным, с учетом обратной связи от пользователей и изменений в библиотечном деле, что обеспечит ее актуальность и эффективность в долгосрочной перспективе. В конечном итоге, веб-приложение должно стать не просто инструментом управления, а полноценной платформой для взаимодействия между библиотеками, читателями и миром знаний.

# Формирование функциональных требований              и выбор инструментов разработки

Для создания эффективного веб-приложения сети библиотек необходимо четко определить его функциональные возможности и выбрать подходящие инструменты разработки. Этот этап играет ключевую роль в обеспечении соответствия конечного продукта потребностям пользователей, а также в достижении высокой производительности, надежности и масштабируемости системы.

Формирование функциональных требований позволяет структурировать задачи, которые будет решать веб-приложение, обозначить ключевые процессы, подлежащие автоматизации, и определить, какие возможности будут доступны для разных категорий пользователей – читателей и библиотекарей.

Выбор инструментов разработки, в свою очередь, направлен на обеспечение оптимального баланса между производительностью, гибкостью и удобством сопровождения системы. На этом этапе рассматриваются современные фреймворки, языки программирования, библиотеки и платформы, которые наиболее подходят для реализации поставленных задач.

Результатом данного этапа станет четкий перечень функциональных требований к веб-приложению и обоснованный выбор инструментов для его разработки, что позволит перейти к этапу проектирования системы и ее практической реализации.

## Функциональные требования к приложению

Для веб-приложения по управлению сетью библиотек можно выделить следующие функциональные требования:

1 Управление книгами: возможность добавлять, редактировать и удалять книги. При добавлении указывается название книги, год выпуска, количество страниц, жанр, категория, предмет, а также авторы. Для каждой книги можно назначить теги, форматы и языки, чтобы обеспечить гибкость поиска и классификации.

2 Управление экземплярами книг: создание записей о конкретных экземплярах книг с указанием привязки к определенной библиотеке, секции, стеллажу и полке. Указывается ISBN, статус доступности (доступен, выдан и т. д.), а также количество доступных копий для каждого экземпляра.

3 Работа с читателями: регистрация новых читателей с вводом их данных, включая имя, фамилию, электронную почту, телефон и никнейм. Возможность редактировать и просматривать информацию о читателях, а также удалять их при необходимости.

4 Выдача читательских билетов: генерация уникального номера читательского билета для каждого читателя. Возможность привязки читательских билетов к привилегиям (например, стандартный доступ, премиум-доступ, а также привилегиям в зависимости от социальных групп, например, студент и пенсионер). Хранение даты выдачи билета.

5 Управление подписками: создание подписок с указанием их типа (например, ежемесячная, годовая), стоимости и срока действия. Просмотр активных подписок у читателей и возможность их продления или удаления.

6 Управление персоналом: ведение учета библиотекарей с хранением их данных, включая имя, фамилию и номер удостоверения личности. Привязка библиотекарей к действиям, таким как выдача или прием книг.

7 Выдача и возврат книг: регистрация операций по выдаче книг читателям с указанием экземпляра книги, читателя, библиотекаря и даты выдачи. Возможность фиксировать дату возврата и рассчитывать штрафы за просрочку возврата.

8 Управление библиотеками и залами: добавление новых библиотек с указанием их названия, адреса, телефона и часов работы. Создание записей о читальных залах с указанием их названия, количества мест и привязки к определенной библиотеке.

9 Штрафы: автоматическое начисление штрафов за просрочку возврата книг. Возможность фиксировать сумму штрафа, причину его начисления и статус оплаты. Просмотр истории штрафов по каждому читателю.

10 Рейтинги и отзывы: возможность оставлять оценки книгам (рейтинги) в диапазоне от 1 до 5. Читатели могут оставлять текстовые отзывы на книги. Просмотр рейтингов и отзывов, чтобы отслеживать популярность книг.

11 Поиск и фильтрация: реализация функции поиска книг по различным параметрам, включая название, автора, жанр, язык и теги. Фильтрация экземпляров книг по их доступности, а также поиск читателей и их активности.

12 Управление издателями: хранение данных об издателях, включая название, город, страну и электронную почту. Привязка издателей к экземплярам книг для ведения учета публикаций и отчетности.

13 Интерфейс администратора: полноценный интерфейс для администраторов, предоставляющий возможность управлять всеми сущностями базы данных, включая книги, экземпляры, читателей, библиотекарей, подписки, штрафы, привилегии и издателей.

Особое внимание в приложении будет уделено автоматизации процессов, таких как выдача и возврат книг, начисление штрафов, управление рейтингами и отзывами. Это позволит повысить эффективность работы библиотек и улучшить пользовательский опыт читателей.

Приложение будет универсальным инструментом для автоматизации библиотечной системы, что сделает её более прозрачной и удобной как для сотрудников, так и для клиентов.

## Выбор инструментов разработки

Всего в современном проектировании и разработке баз данных выделяют два основных вида: SQL базы данных и NoSQL базы данных. Их отличие в том, что SQL базы данных используют реляционную теорию, а NoSQL – нет. Для более явного визуального представления рассмотрим рисунок 2.1, на котором изображены основные подвиды баз данных.



Рисунок 2.1 – Основные виды SQL и NoSQL баз данных

SQL и NoSQL базы данных представляют два подхода к хранению и управлению данными, которые различаются структурой, принципами работы и сферами применения. SQL базы данных, также известные как реляционные, организуют данные в таблицы с фиксированными схемами и используют язык SQL для выполнения запросов. Они идеально подходят для приложений, где важна строгая структура данных и гарантированная целостность, например, в банковских системах или учете заказов.

NoSQL базы данных, напротив, предлагают гибкую модель хранения, поддерживая различные форматы данных, такие как документы, графы или ключ-значение. Они лучше справляются с обработкой больших объемов информации, не требуя жесткой структуры, и позволяют легко масштабироваться горизонтально. Это делает их популярным выбором для приложений с быстро меняющимися данными, таких как социальные сети или системы аналитики.

Таким образом, выбор между SQL и NoSQL базами данных зависит от особенностей конкретного проекта и требований к его функциональности.

**2.2.1** Нереляционные базы данных

Нереляционные базы данных, или NoSQL, представляют собой современный подход к хранению и управлению данными, который отличается от традиционной табличной структуры реляционных баз данных. Вместо таблиц данные хранятся в различных форматах, таких как документы, пары ключ-значение, графы или колонки. Такая гибкость позволяет нереляционным базам адаптироваться под множество различных задач, особенно в тех случаях, когда структура данных постоянно меняется или сложно заранее определить строгую схему.

Одной из ключевых особенностей NoSQL баз является возможность хранения неструктурированных или полуструктурированных данных. Это делает их удобным выбором для приложений, которые работают с большими объемами информации, например, с данными из социальных сетей, логами или интернет вещей. Благодаря способности обрабатывать данные без сложных преобразований структуры, такие базы обеспечивают высокую производительность и сокращают время на разработку.

Еще одним важным преимуществом является масштабируемость. Нереляционные базы данных легко масштабируются горизонтально, то есть за счет добавления новых серверов. Это особенно важно для распределенных систем и облачных приложений, где нагрузка может расти непредсказуемо. Также они оптимизированы для работы в условиях высокой скорости записи и чтения, что делает их идеальным выбором для аналитических платформ, кэширования и систем с высокой интенсивностью операций.

Однако такой подход имеет свои ограничения. Например, многие нереляционные базы данных не обеспечивают полного соблюдения ACID-гарантий, как реляционные системы, что может повлиять на надежность данных в критически важных приложениях. Вместо этого они часто следуют принципу BASE, где согласованность данных достигается со временем. Также отсутствие универсального языка запросов, подобного SQL, может усложнить освоение и работу с ними, поскольку каждая система использует свои собственные методы взаимодействия.

Нереляционные базы данных чаще всего выбирают для проектов, где важно быстрое масштабирование, гибкость в структуре данных и высокая скорость обработки. Они стали неотъемлемой частью современной разработки, особенно в таких областях, как Big Data, социальные сети и аналитические системы, и прекрасно дополняют реляционные базы данных, предлагая новые возможности для работы с динамичными и разнообразными данными.

**2.2.2** Хранилища данных документов

Хранилища данных документов, или документно-ориентированные базы данных, являются одним из типов нереляционных баз данных, которые специально разработаны для хранения и управления структурированными или полуструктурированными данными в виде документов. Каждый документ представляет собой независимую запись, которая обычно хранится в формате JSON, BSON, XML или аналогичных структурах, что делает данные легко читаемыми и гибкими в использовании. Структура хранения данных в хранилище документов представлена на рисунке 2.2.

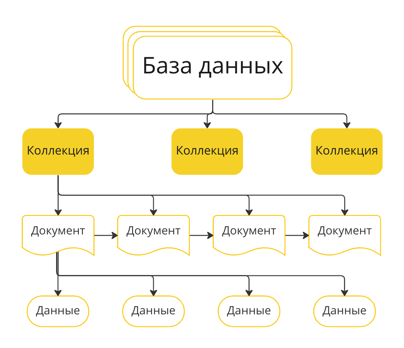


Рисунок 2.2 – Структура хранения данных в хранилище документов

Основное преимущество хранилищ документов заключается в их способности хранить сложные и разнообразные данные без необходимости создания строгой схемы. В отличие от реляционных баз данных, где изменения структуры требуют модификации таблиц и связей, в документных базах данные можно добавлять или изменять на лету. Это делает их особенно полезными для приложений с быстро меняющимися требованиями или с данными, которые имеют сложные вложенные структуры, например, пользовательские профили, заказы с множеством атрибутов или логи событий.

Документные базы данных оптимизированы для работы с данными, которые часто читаются и модифицируются. Они предоставляют высокую производительность за счет использования ключей для поиска и простого доступа к документам. Это делает их подходящими для приложений, где требуется высокая скорость отклика, таких как системы управления контентом, интернет-магазины или мобильные приложения. Гибкость модели данных также позволяет разработчикам сохранять данные в естественном для приложения формате, сокращая необходимость в преобразовании или нормализации данных.

Однако использование хранилищ данных документов имеет и свои ограничения. Несмотря на высокую производительность при работе с отдельными документами, выполнение сложных аналитических запросов, требующих объединения данных, может быть менее эффективным по сравнению с реляционными базами. Также отсутствие строгой схемы может усложнить управление данными в долгосрочной перспективе, особенно если структура документов становится слишком разрозненной или неоднородной.

Примеры документно-ориентированных баз данных включают MongoDB, CouchDB и Firebase Firestore. Эти системы широко используются в веб-разработке, электронных коммерциях и аналитических платформах благодаря своей гибкости, производительности и способности работать с разнообразными данными. Хранилища данных документов хорошо подходят для приложений, где важна простота работы с данными и быстрая адаптация к изменениям. [10]

**2.2.3** Хранилище пар «ключ-значение»

Хранилища пар «ключ-значение» представляют собой один из типов нереляционных баз данных, предназначенный для эффективного хранения и управления данными в формате простых ассоциативных пар. Каждая запись в таком хранилище состоит из уникального ключа и соответствующего ему значения, которое может быть как простым, например, строкой или числом, так и более сложным, например, массивом или сериализованным объектом. Эта структура позволяет быстро и легко получать доступ к данным, просто обращаясь к ним по ключу, без необходимости выполнения сложных запросов или фильтрации.

Основное преимущество хранилищ пар «ключ-значение» заключается в их высокой производительности и простоте. Доступ к данным осуществляется напрямую по ключу, что значительно снижает время обработки запроса. Такой подход особенно полезен для кэширования данных, управления пользовательскими сеансами или хранения конфигурационных параметров приложения. Гибкость в структуре значений позволяет разработчикам хранить данные в наиболее подходящем для их приложения виде, избегая необходимости жестко следовать заранее определённой схеме. Пример формата хранения данных в хранилище пар «ключ-значение» представлен на рисунке 2.3.

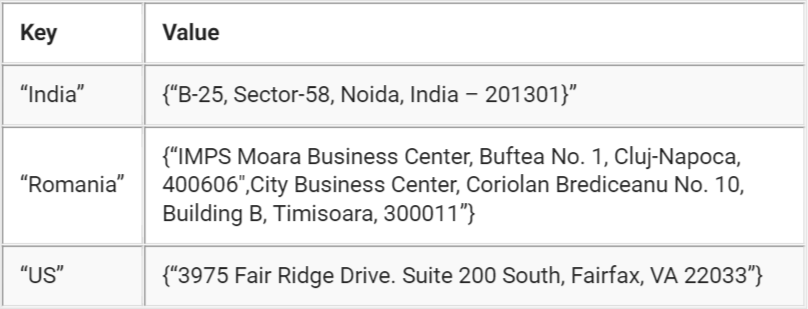


Рисунок 2.3 – Пример формата хранения данных в хранилище пар «ключ-значение»

Одним из ключевых достоинств хранилищ пар «ключ-значение» является их способность к горизонтальному масштабированию. Данные могут быть распределены между несколькими узлами или серверами, что обеспечивает высокую доступность и отказоустойчивость системы. Это делает такие хранилища отличным выбором для распределённых систем и облачных приложений, где важны надёжность и масштабируемость.

Однако хранилища пар «ключ-значение» имеют и свои ограничения. Они плохо подходят для выполнения сложных аналитических запросов или операций, требующих объединения данных. Также отсутствие строгой схемы может привести к неупорядоченному хранению данных и усложнить их долгосрочное управление. В случаях, когда данные требуют структурированных связей и регулярного выполнения сложных операций, реляционные базы данных могут оказаться более подходящим решением.

Примеры популярных хранилищ пар «ключ-значение» включают Redis, Memcached и Amazon DynamoDB. Эти системы широко используются в веб-разработке, облачных вычислениях и системах реального времени благодаря своей простоте, скорости и способности справляться с большими объёмами данных. Хранилища пар «ключ-значение» идеально подходят для приложений, где важны высокая производительность, гибкость в хранении данных и возможность масштабирования, что делает их незаменимым инструментом в современном программировании. [11]

**2.2.4** Хранилища данных графов

Хранилища данных графов управляют сведениями двух типов: узлами и ребрами. Узлы в этом случае представляют сущности, а ребра определяют связи между ними. Узлы и грани имеют свойства, которые предоставляют сведения о конкретном узле или грани, примерно, как столбцы в реляционной таблице. Грани могут иметь направление, указывающее на характер связи.

Хранилища данных графов позволяют приложениям эффективно выполнять запросы, которые проходят через сеть узлов и ребер, а также анализировать связи между сущностями. Пример данных, структурированных в виде графа, представлен на рисунке 2.4.

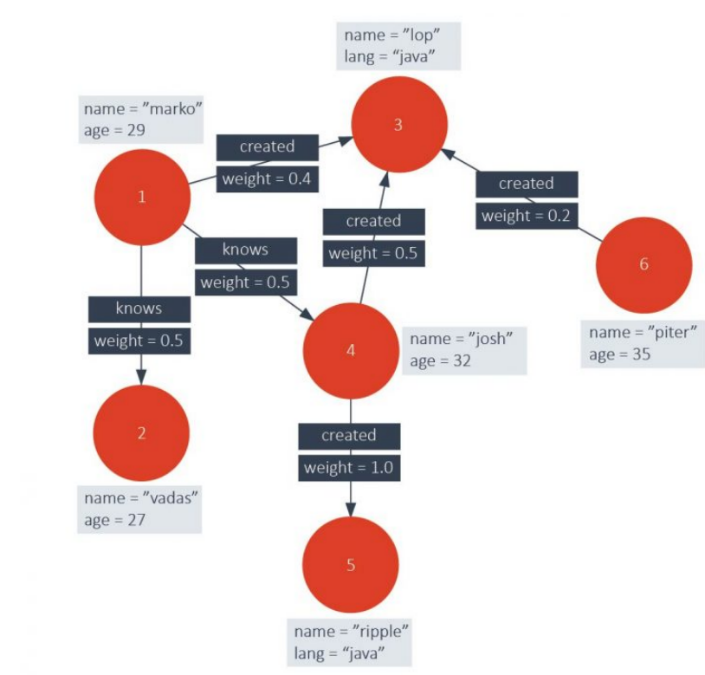


Рисунок 2.4 – Пример данных, структурированных в виде графа

Сущностями здесь являются сотрудники и отделы, а грани определяют отношения подчинения и отдел, в котором работает каждый сотрудник. Стрелки на ребрах этого графа показывают направление связей.

Такая структура позволяет легко выполнять такие запросы, как «найти всех сотрудников, которые прямо или косвенно подчиняются Ольге» или «найти всех, кто работает в одном отделе с Глебом». Процессы сложного анализа выполняются быстро даже на больших графах с большим количеством сущностей и связей. Многие БД графов предоставляют язык запросов, который можно использовать для эффективного обхода сети связей. [12]

**2.2.5** Реляционные базы данных

Реляционные базы данных – это тип базы данных, в которой хранятся и систематизируются точки данных с определенными связями для быстрого доступа. В реляционной базе данные организованы в таблицы, содержащие информацию о каждом объекте и представляющие заранее определенные категории через строки и столбцы. Такое структурирование данных делает доступ к ним эффективным и гибким, поэтому реляционные базы данных наиболее распространены. Реляционные базы данных также созданы для понимания языка структурированных запросов (SQL), стандартизированного языка программирования, который используется для хранения, управления и извлечения данных. В SQL есть встроенный язык для создания таблиц, называемый языком определения данных (DDL), и язык для манипулирования данными, называемый языком манипулирования данными (DML).

Реляционный означает указание или конституирование отношения. В контексте баз данных то, как мы определяем реляционность, применяется в первую очередь к самим данным. Реляционные наборы данных имеют заранее определенные отношения между собой. Например, база данных, содержащая информацию о клиентах компании, может также включать данные об отдельных транзакциях, прикрепленных к каждой учетной записи. Реляционные базы данных фокусируют внимание на отношениях между хранимыми элементами данных. Пример структуры реляционной базы данных на рисунке 2.5.

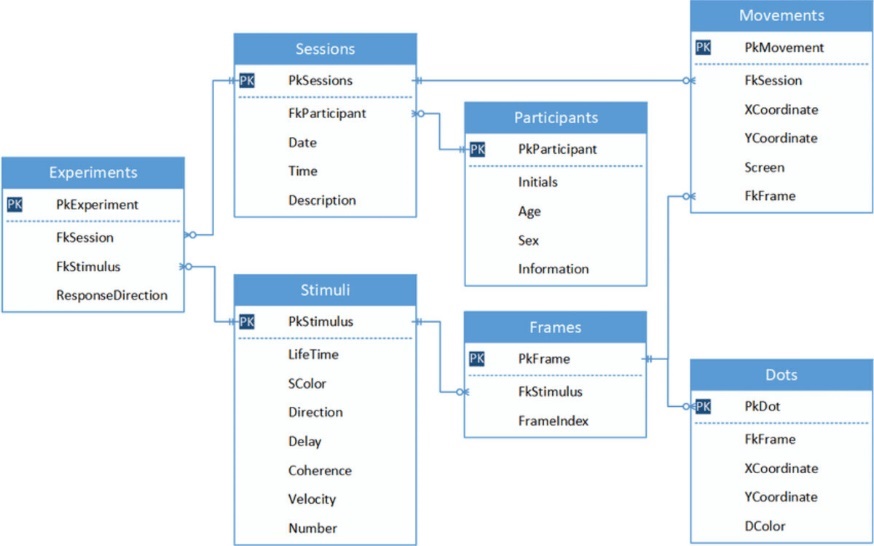


Рисунок 2.5 – Пример структуры реляционной базы данных

Реляционные базы данных обычно используют таблицы, данные в которых организованы в строки, содержащие сущности, и столбцы, содержащие атрибуты сущностей. Этот процесс известен как нормализация. Каждая строка содержит уникальный идентификатор или ключ, который связывает таблицы вместе для установления связи. При запросе к реляционной базе данных ключ используется для поиска связанных данных в наборах данных. Например, служба технической поддержки может захотеть отслеживать взаимодействие с клиентами по типу проблемы, времени на ее решение и степени удовлетворенности клиентов. В этой базе данных единый идентификатор клиента создает связь и обеспечивает хорошее функционирование структуры таблицы.

Реляционные базы данных полезны для любых информационных потребностей, когда точки данных связаны друг с другом, а также должны управляться согласованным, безопасным и основанным на правилах способом. Именно это делает реляционные базы данных наиболее популярными для бизнеса и предприятий. Когда компании хотят извлечь ценную информацию из своих собственных данных, они полагаются на реляционные базы данных для получения полезной аналитики. Многие отчеты, которые компании создают для отслеживания запасов, финансов, продаж или прогнозирования будущего, создаются с использованием реляционных баз данных.

Данные в реляционных базах данных хранятся, просматриваются и извлекаются из таблиц со связями. В реляционной базе данных схема базы данных определяет, как данные организованы как логически, так и физически.

Реляционные базы данных имеют так называемый режим согласованности или целостности, основанный на четырех критериях: атомарность, согласованность, изоляция и долговечность (ACID). Вот значение каждого свойства базы данных ACID:

1 Атомарность: определяет элементы, составляющие полную транзакцию.

2 Согласованность: определяет правила поддержания целостности данных после транзакции.

3 Изоляция: делает последствия транзакций невидимыми для других, поэтому они не конфликтуют друг с другом.

4 Долговечность: гарантирует, что изменения данных станут постоянными после каждой зафиксированной транзакции.

Эти критерии делают реляционные базы данных полезными в приложениях, требующих высокой точности.

Примеры реляционных баз данных включают SQL Server, управляемый экземпляр Azure SQL, базу данных MySQL, PostgreSQL и Oracle. [13]

**2.2.6** PostgreSQL

PostgreSQL – это многофункциональная, объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом, которая широко используется в самых разных сферах – от небольших веб-приложений до масштабных корпоративных систем. Она славится своей надёжностью, безопасностью, гибкостью и высокой производительностью, что делает её одним из ведущих решений на рынке баз данных. PostgreSQL сочетает строгие стандарты SQL с расширенными возможностями работы с полуструктурированными данными, такими как JSON и XML, что позволяет эффективно использовать её как для традиционных реляционных задач, так и для современных сценариев, требующих более гибкой модели хранения данных.

Одним из ключевых преимуществ PostgreSQL является её способность обрабатывать сложные запросы и поддержка транзакций с гарантией ACID, что обеспечивает целостность и согласованность данных даже в условиях высокой нагрузки. Система предлагает многоуровневое управление параллельным доступом (MVCC), которое предотвращает блокировки при одновременных операциях чтения и записи. Это позволяет эффективно обслуживать множество пользователей и выполнять ресурсоёмкие аналитические запросы без потери производительности. Индексация данных представлена широким набором методов, включая B-Tree, Hash, GiST, GIN и BRIN индексы, что позволяет ускорять поиск и оптимизировать выполнение запросов на больших объёмах данных.

Расширяемость PostgreSQL позволяет разработчикам адаптировать систему под свои нужды. Пользовательские функции, операторы, индексы и типы данных могут быть добавлены для решения уникальных задач. Встроенная поддержка языков программирования, таких как PL/pgSQL, Python и JavaScript, позволяет создавать хранимые процедуры и сложные бизнес-правила прямо на уровне базы данных. Это делает PostgreSQL особенно удобной для сложных и многозадачных систем, где требуется обработка данных с минимальными задержками.

Отдельное внимание стоит уделить безопасности PostgreSQL. Система поддерживает контроль доступа на уровне строк (Row-Level Security), шифрование данных и широкие возможности аутентификации через LDAP, PAM и Kerberos. Гибкая система прав доступа позволяет детально контролировать, кто и к каким данным имеет доступ, вплоть до уровня отдельных таблиц и столбцов. Это делает PostgreSQL популярным выбором для финансовых систем, медицинских платформ и других областей, где безопасность данных критически важна.

Высокая доступность и отказоустойчивость обеспечиваются за счёт физической и логической репликации данных. Физическая репликация позволяет создавать точные копии базы данных для обеспечения отказоустойчивости, в то время как логическая репликация позволяет синхронизировать определённые таблицы или наборы данных между серверами. Это позволяет строить распределённые системы с минимальными рисками потери данных и длительными простоями.

PostgreSQL успешно применяется для аналитических задач благодаря поддержке сложных оконных функций, CTE (Common Table Expressions), агрегатных функций и возможности выполнения глубоких многозвеньевых запросов. Это делает её незаменимой в системах бизнес-анализа, предиктивной аналитики и платформ для принятия управленческих решений. Гибкость системы позволяет легко интегрировать PostgreSQL с другими технологиями, такими как Python, Java, Node.js и PHP, что облегчает разработку и развертывание приложений.

Модели реляционных баз данных могут варьироваться от небольших настольных систем до крупных облачных систем. Они используют базу данных SQL или могут обрабатывать операторы SQL для запросов и обновлений. Реляционные модели определяются логическими структурами данных (таблицами, индексами и представлениями) и хранятся отдельно от физических структур хранения (физических файлов). Согласованность данных является отличительной чертой моделей реляционных баз данных, поскольку они поддерживают целостность данных в приложениях и копиях баз данных, также называемых экземплярами. При использовании базы данных реляционной модели несколько экземпляров базы данных всегда содержат одни и те же данные.

Реляционные базы данных, разработанные в облаке, автоматически настраиваются для обеспечения высокой доступности. Это означает, что данные реплицируются или копируются на нескольких участниках, причем каждый участник находится в отдельных зонах доступности. Таким образом, данные по-прежнему доступны, даже если отдельный центр обработки данных не работает.

Традиционные реляционные базы данных созданы для обработки больших объемов структурированных данных. Это делает реляционные базы данных особенно хорошо подходящими для больших структурированных данных, поскольку они полагаются на SQL и могут использовать системы управления базами данных для управления данными. Однако более крупные и сложные наборы данных больших данных содержат все больше разнообразия, а это означает, что данные становятся все менее и менее структурированными и поступают из новых источников. [14]

**2.2.7** Выбор базы данных для разрабатываемой системы

Исходя из анализа подходов в предыдущем пункте, следует сделать вывод, что предметная область библиотеки будет реализована с помощью реляционной базы данных. Потому что выбранная предметная область имеет четкие атрибуты, сущности и связи, которые достаточно просто можно представить в виде таблиц и ключей в реляционной теории разработки баз данных. Из реляционных баз данных для реализации предметной области будет выбрана PostgreSQL, так как она является современной СУБД с поддержкой всех необходимых для предметной области функций.

# Проектирование базы данных

Проектирование базы данных — это процесс, включающий в себя последовательные этапы разработки, внедрения и сопровождения систем управления данными для бизнеса. Его главная задача заключается в создании логической и физической моделей будущей базы данных, обеспечивающих эффективное хранение и обработку информации.

## Инфологическая модель

Стартовой точкой в проектировании БД является создание инфологической модели. Инфологическая модель представляет собой описание сущностей и связанных вместе с ними атрибутов.

Предметная область включает в себя следующие сущности и атрибуты:

1 Сущность «Автор»: данная сущность имеет следующие атрибуты:

– фамилия;

– имя;

– дата рождения;

– страна.

Данная сущность связана с сущностью «Книга» через таблицу «Автор\_Книга» (один автор может написать несколько книг, несколько авторов могут написать одну книгу).

2 Сущность «Жанр»: данная сущность имеет следующие атрибуты:

– название.

Связана с сущностью «Книга» (у нескольких книг может быть один жанр).

3 Сущность «Категория»: данная сущность имеет следующие атрибуты:

– название.

Связана с сущностью «Книга» (у нескольких книг может быть одна категория).

4 Сущность «Предмет»: данная сущность имеет следующие атрибуты:

– название.

Связана с сущностью «Книга» (несколько книг могут относиться к одному предмету).

5 Сущность «Книга»: данная сущность имеет следующие атрибуты:

– название;

– количество страниц;

– год издания.

Связана с сущностью «Автор» через сущность «Автор\_Книга» (одна книга может иметь нескольких авторов, у нескольких авторов может быть одна книга).

Связана с сущностью «Жанр» (может иметь один жанр).

Связана с сущностью «Категория» (может иметь одну категорию).

Связана с сущностью «Предмет» (может относиться к одному предмету).

Связана с сущностью «Формат» (может иметь несколько форматов).

Связана с сущностью «Язык» через таблицу «Книга\_Язык» (может быть на разных языках).

Связана с сущностью «Тег» через таблицу «Книга\_Тег» (может иметь несколько тегов).

Связана с сущностью «Экземпляр» (одна книга может иметь несколько экземпляров).

Связана с сущностью «Оценка» (книга может иметь несколько оценок)

6 Сущность «Формат»: данная сущность имеет следующие атрибуты:

– название.

Связана с сущностью «Книга» (одна книга может иметь несколько форматов).

Связана с сущностью «Форма» (один формат может быть привязан к нескольким формам).

7 Сущность «Форма»: данная сущность имеет следующие атрибуты:

– название.

Связана с сущностью «Формат» (может быть связано с несколькими форматами).

8 Сущность «Тег»: данная сущность имеет следующие атрибуты:

– название.

Связана с сущностью «Книга» через сущность «Книга\_Тег» (одна книга может быть отмечена несколькими тегами).

9 Сущность «Язык»: данная сущность имеет следующие атрибуты:

– название;

– код.

Связана с сущностью «Книга» через сущность «Книга\_Язык» (книга может быть на разных языках).

10 Сущность «Издательство»: данная сущность имеет следующие атрибуты:

– название;

– город;

– страна;

– электронная почта.

Связана с сущностью «Экземпляр» (одно издательство может издавать несколько книг).

11 Сущность «Экземпляр»: данная сущность имеет следующие атрибуты:

– ISBN;

– год выпуска.

Связана с сущностью «Книга» (одна книга может иметь несколько экземпляров).

Связана с сущностью «Издательство» (один экземпляр имеет одно издательство).

Связана с сущностью «Доступность» (один экземпляр имеет одну доступность).

Связана с сущностью «Взятие» (экземпляр может быть взят несколько раз).

Связана с сущностью «Позиция» (экземпляр может иметь место в библиотеке)

12 Сущность «Доступность»: данная сущность имеет следующие атрибуты:

– количество.

Связана с сущностью «Экземпляр» (одна доступность имеет один экземпляр).

Связана с сущностью «Позиция» (позиция определяет доступность экземпляра в библиотеке).

13 Сущность «Позиция»: данная сущность имеет следующие атрибуты:

– секция;

– стеллаж;

– полка.

Связана с сущностью «Библиотека» (позиция относится к одной библиотеке).

Связана с сущностью «Доступность» (одна позиция имеет одну доступность).

14 Сущность «Библиотека»: данная сущность имеет следующие атрибуты:

– название;

– часы работы;

– адрес;

– номер телефона.

Связана с сущностью «Позиция» (библиотека может иметь несколько позиций для хранения книг).

Связана с сущностью «Читальный зал» (библиотека может иметь несколько читальных залов).

15 Сущность «Читальный зал»: данная сущность имеет следующие атрибуты:

– название;

– часы работы;

– количество мест;

– номер телефона.

Связана с сущностью «Библиотека» (один читальный зал может относиться к одной библиотеке).

16 Сущность «Библиотечная карта»: данная сущность имеет следующие атрибуты:

– номер карты;

– дата выдачи.

Связана с сущностью «Читатель» (один читатель имеет одну библиотечную карту).

Связана с сущностью «Привилегия» (одна карта может иметь одну привилегию).

Связана с сущностью «Подписка» (библиотечная карта может иметь несколько подписок)

17 Сущность «Привилегия»: данная сущность имеет следующие атрибуты:

– название;

– описание.

Связана с сущностью «Библиотечная карта» (одна привилегия может быть связана с несколькими библиотечными картами).

18 Сущность «Тип»: данная сущность имеет следующие атрибуты:

– название;

– описание.

Связана с сущностью «Подписка» (один тип подписки может быть связан с несколькими подписками).

19 Сущность «Подписка»: данная сущность имеет следующие атрибуты:

– название;

– цена;

– срок действия.

Связана с сущностью "Библиотечная карта" (одна библиотека может иметь несколько подписок).

Связана с сущностью "Тип" (одна подписка имеет один тип).

20 Сущность «Удостоверение личности»: данная сущность имеет следующие атрибуты:

– персональный номер;

– должность;

– срок действия.

Связана с сущностью «Библиотекарь» (одно удостоверение личности имеет одного библиотекаря).

21 Сущность «Библиотекарь»: данная сущность имеет следующие атрибуты:

– имя;

– фамилия.

Связана с сущностью «Удостоверение личности» (один библиотекарь имеет одно удостоверение личности).

Связана с сущностью «Взятие» (библиотекарь может оформить несколько взятий).

22 Сущность «Читатель»: данная сущность имеет следующие атрибуты:

– никнейм;

– имя;

– фамилия;

– электронная почта;

– номер телефона.

Связана с сущностью «Библиотечная карта» (один читатель может иметь одну библиотечную карту).

Связана с сущностью «Взятие» (читатель может взять несколько взятий).

Связана с сущностью "Обзор" (читатель может написать несколько обзоров)

23 Сущность «Взятие»: данная сущность имеет следующие атрибуты:

– дата выдачи;

– дата возврата.

Связана с сущностью «Экземпляр» (один взятие связан с одним экземпляром книги).

Связана с сущностью «Читатель» (один взятие связан с одним читателем).

Связана с сущностью «Библиотекарь» (один взятие оформляется одним библиотекарем).

Связана с сущностью «Штраф» (один взятие может иметь один штраф).

24 Сущность «Штраф»: данная сущность имеет следующие атрибуты:

– сумма;

– причина.

Связана с сущностью «Взятие» (один штраф может быть связан с одним взятием).

25 Сущность «Оценка»: данная сущность имеет следующие атрибуты:

– оценка.

Связана с сущностью «Книга» (одна книга может иметь несколько оценок).

Связана с сущностью «Обзор» (одна оценка может быть в одном обзоре)

26 Сущность «Обзор»: данная сущность имеет следующие атрибуты:

– дата;

– текст.

Связана с сущностью «Читатель» (один обзор написан одним читателем)

Связана с сущностью «Оценка» (один обзор связан с одной оценкой).

Схема инфологической модели базы данных представлена на рисунке 3.1.

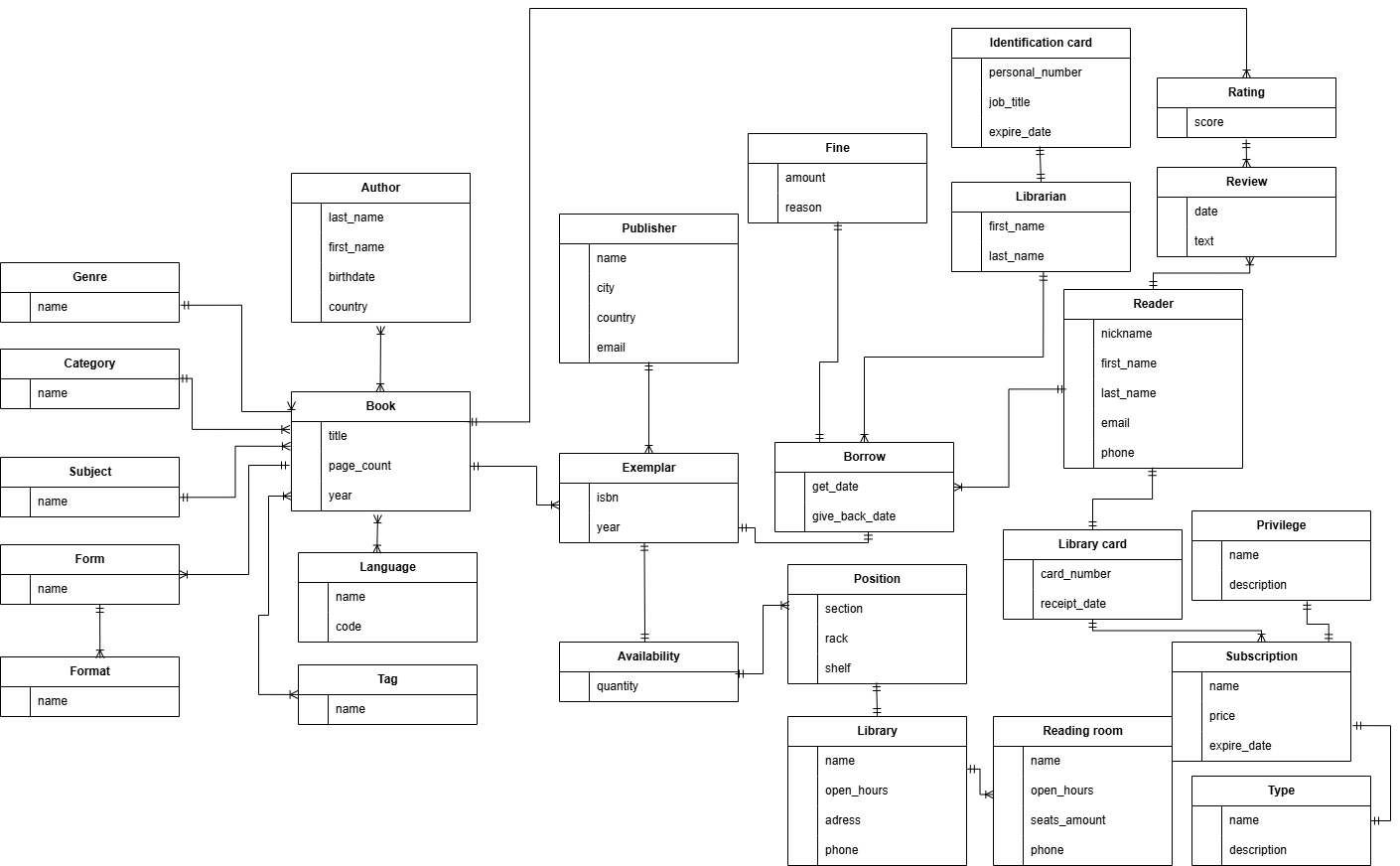


Рисунок 3.1 – Инфологическая модель

Следовательно, инфологическая модель, будучи абстрактным представлением, обеспечивает простое и понятное видение структуры данных и взаимодействий между элементами системы, что является необходимым условием для успешного проектирования базы данных.

## Даталогической модель

Даталогическая модель данных представлена в приложении Б.

Даталогическая модель – это конкретное представление о данных, которые будут храниться в базе данных.

Этапы составления даталогической модели:

1 Создание таблиц: это первый шаг при составлении даталогической модели. Необходимо создать таблицы для каждой сущности, которая была определена в инфологической модели.

2 Создание полей: после того как созданы таблицы, необходимо создать поля (столбцы) для каждого атрибута, который был определен в инфологической модели.

3 Определение типов данных: для каждого поля необходимо определить тип данных, который будет храниться в этом поле (например, число, строка, дата и т.д.).

4 Создание связей: после того как созданы таблицы и поля, необходимо создать связи между таблицами. Связи могут быть однородными или разнородными, как и в инфологической модели.

5 Создание ключей для каждой таблицы необходимо создать ключ (Primary Key), который будет уникальным идентификатором записи в этой таблице. Также можно создать внешние ключи (Foreign Key), которые связывают записи в разных таблицах.

Даталогическая модель описывает способ хранения данных и их структуру. Эта модель используется для создания таблиц и связей между ними.

## Создание модели миграции данных

В процессе реализации системы возможны ситуации, когда потребуется перенос данных из существующих источников (например, старой базы данных) в новую систему. Это может происходить по причине перехода на новую версию системы, интеграции с другими системами или улучшения структуры данных. Модель миграции данных для данного проекта будет включать в себя подготовку данных, трансформацию их в нужный формат и перенос в целевую систему, что гарантирует соответствие всех данных новым стандартам и структурам.

Миграция данных в данном контексте будет включать несколько ключевых этапов:

1 Анализ текущих данных: на первом этапе необходимо понять структуру текущих данных и определить, какие данные будут мигрированы. Для этого нужно провести анализ всех таблиц, которые будут переноситься, и их связи.

2 Извлечение данных: на этом этапе данные извлекаются из старой базы данных. Для каждой таблицы будут созданы отдельные запросы на извлечение данных. Извлечение может происходить с учетом определенных фильтров или периодичности. Также могут быть использованы инструменты для миграции данных, такие как Talend или Apache Nifi, для автоматизации процесса.

3 Трансформация данных: после извлечения данных необходимо привести их к требуемому формату. Например, трансформация может включать: преобразование типов данных, преобразование значений, маппинг сущностей между системами.

4 Загрузка данных в целевую систему: после того как данные будут извлечены и преобразованы, их необходимо загрузить в новую систему. Это будет включать выполнение SQL-запросов на вставку данных в целевые таблицы. Если данные поступают из разных источников, может быть использована автоматизация для загрузки их в соответствующие таблицы.

5 Пост-миграционный анализ и валидация: после загрузки данных важно провести валидацию на целостность и корректность: проверка наличия всех записей, сравнение количества записей в исходной и целевой таблице, проверка связей между таблицами.

Миграция данных – это неотъемлемая часть любого проекта, связанного с изменением структуры или обновлением базы данных. Для успешной миграции данных необходимо внимательно продумать все этапы – от анализа текущих данных до загрузки и проверки целостности в новой системе.

# Разработка базы данных

После того, как были разработаны инфологическая и даталогическая модели базы данных, осуществлён выбор СУБД и языка программирования, стало возможным приступить к непосредственной реализации сущностей базы данных.

## 4.1 Создание сущностей базы данных

Программный SQL-код, реализующий сущность проектной области представлен в приложении А. Данный код создает сущности согласно инфологической и даталогической моделям данных.

## 4.2 Создание хранимых процедур

Нередко операция с данными представляет набор инструкций, которые необходимо выполнить в определенной последовательности. Например, при добавлении данных покупки товара необходимо внести данные в таблицу заказов. Однако перед этим надо проверить, а есть ли покупаемый товар в наличии. Возможно, при этом понадобится проверить еще ряд дополнительных условий. То есть фактически процесс покупки товара охватывает несколько действий, которые должны выполняться в определенной последовательности. И в этом случае более оптимально будет инкапсулировать все эти действия в один объект – хранимую процедуру.

**4.2.1** Хранимая процедура get\_book\_by\_id

Хранимая процедура get\_book\_by\_id реализована для поиска книг по переданному в процедуру идентификатору книги. Реализация данной хранимой процедуры представлена на рисунке 4.1.

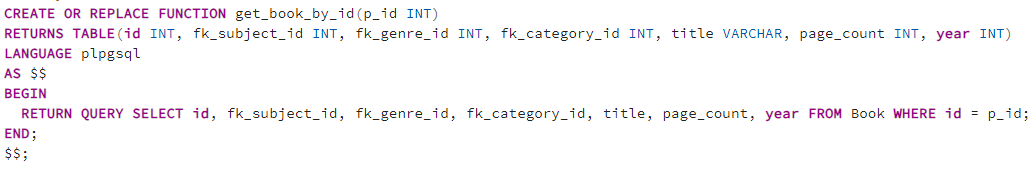


Рисунок 4.1 – Реализация хранимой процедуры get\_book\_by\_id

Процедура get\_book\_by\_id позволяет находить в таблице простым вызовом процедуры, передавая идентификатор книги. Так же процедура гарантирует, что выдача данных после нахождения книги будет в одном и том же формате.

**4.2.2** Хранимая процедура add\_new\_reader

Хранимая процедура add\_new\_reader реализована для создания новых читателей, которые регистрируются на сайте бибилиотеки. Реализация данной хранимой процедуры представлена на рисунке 4.2.

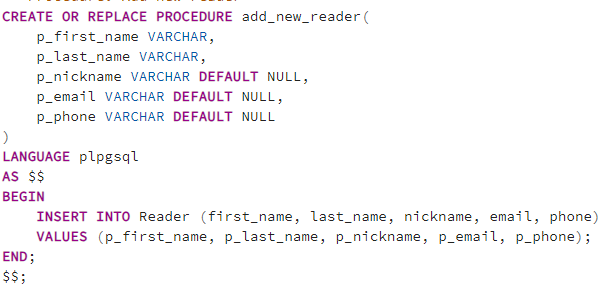


Рисунок 4.2 – Реализация хранимой процедуры add\_new\_reader

Процедура add\_new\_reader позволяет добавлять новых читателей в таблицу простым вызовом процедуры, передавая название и описание читателя.

**4.2.3** Хранимая процедура get\_authors\_of\_book

Хранимая процедура get\_authors\_of\_book реализована для получения всех авторов книги. Реализация данной процедуры представлена на рисунке 4.3.

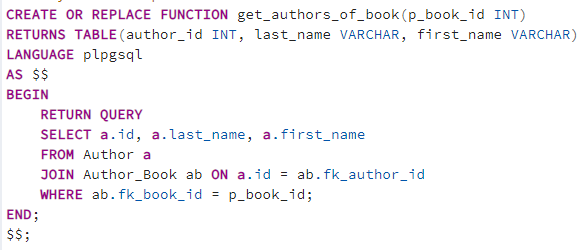


Рисунок 4.3 – Реализация хранимой процедуры get\_authors\_of\_book

Данная процедура реализована для удобного поиска всех авторов книги, так как авторы и книги связаны через таблицу-посредника.

**4.2.4** Хранимая процедура get\_books\_of\_author

Хранимая процедура get\_books\_of\_author реализована для получения всех книг, написанных автором. Реализация хранимой процедуры add\_agent\_or\_admin представлена на рисунке 4.4.

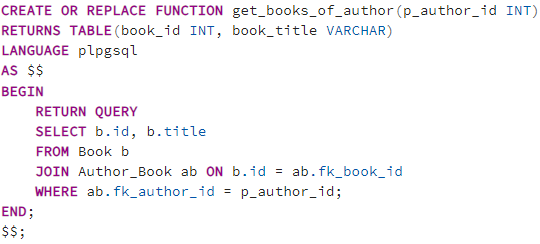


Рисунок 4.4 – Реализация хранимой процедуры get\_books\_of\_author

Данная процедура реализована для удобного поиска всех книг, написанных автором, так как авторы и книги связаны через таблицу-посредника.

**4.2.5** Хранимая процедура get\_languages\_of\_book

Хранимая процедура get\_languages\_of\_book реализована для возможности получить все языки, на которых книга есть на сайте. Реализация данной хранимой процедуры представлена на рисунке 4.5.

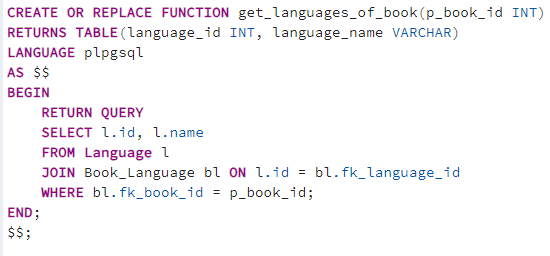


Рисунок 4.5 – Реализация хранимой процедуры get\_languages\_of\_book

Так как связь языка и книги реализована через таблицу-посредника, то вместо того, чтобы каждый раз писать запрос большого размера, можно воспользоваться данной процедурой. Аналогичная процедура есть для получения всех тегов, привязанных к книге.

**4.2.6** Хранимая процедура get\_books\_of\_tag

Хранимая процедура get\_books\_of\_tag реализована для возможности получить информацию об всех книгах, привязанных к тегу. Реализация хранимой процедуры get\_books\_of\_tag представлена на рисунке 4.6.

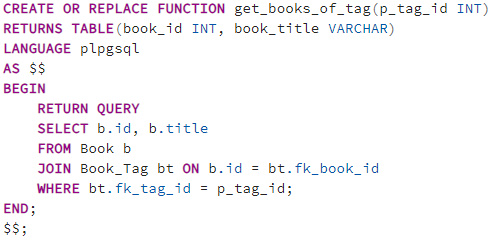


Рисунок 4.6 – Реализация хранимой процедуры get\_books\_of\_tag

Чтобы не писать каждый раз длинный запрос, соединяющий данные через таблицу-посредника, можно воспользоваться данной процедурой. Аналогичная процедура есть для получения всех книг, имеющихся на сайте на определенном языке.

**4.2.7** Хранимая процедура delete\_publisher

Хранимая процедура delete\_publisher реализована для удаления издателя по заданному идентификатору, что упрощает удаление издателей, ведь вместо целого запроса нужно просто вызвать процедуру и передать в нее идентификатор. Реализация хранимой процедуры delete\_publisher представлена на рисунке 4.7.

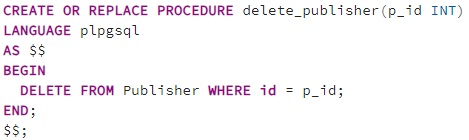


Рисунок 4.7 – Реализация хранимой процедуры delete\_publisher

Огромное количество экземпляров сущностей в БД удаляется одинаково, меняется лишь идентификатор и смысл самой сущности. В коде также присутствуют идентичные хранимые процедуры для других таблиц, что упрощает удаление экземпляров сущностей и работу с ними.

Таким образом хранимые процедуры помогают заменить набор инструкций одной строкой для гибкой и упрощенной работы с кодом.

## 4.3 Создание триггеров

Триггер – это блок кода на PL/pgSQL, который сохраняется в базе данных и автоматически выполняется при наступлении определённого события. Он может быть привязан к таблице, представлению, схеме (для её владельца) или всей базе данных (для всех пользователей).

Для удобства написания основная логика, выполняемая триггером, находится в функции, а сам триггер вызывает соответствующую функцию в нужный момент.

**4.3.1** Триггер borrow\_dates\_check

Триггер borrow\_dates\_check реализован для невозможности взять из библиотеки один и тот же экземпляр в одинаковый или пересекающийся срок. Реализация данного триггера представлена на рисунке 4.8.

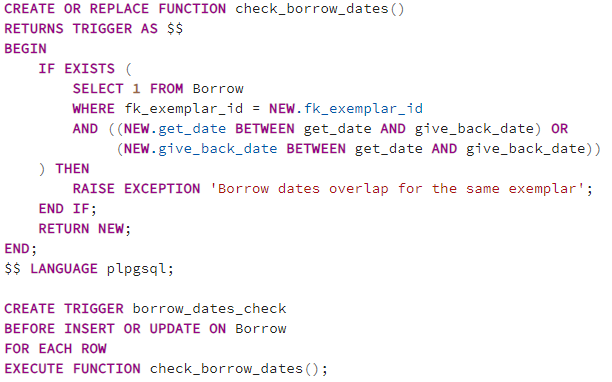


Рисунок 4.8 – Реализация триггера borrow\_dates\_check

Чтобы взять экземпляр, нужно подождать, пока другой читатель вернет его или взять другой экземпляр.

**4.3.2** Триггер valid\_rating\_score\_check

Триггер valid\_rating\_score\_check реализован для предотвращения выставления рейтингов книгам, не предусмотренным системой. Реализация данного триггера представлена на рисунке 4.9.

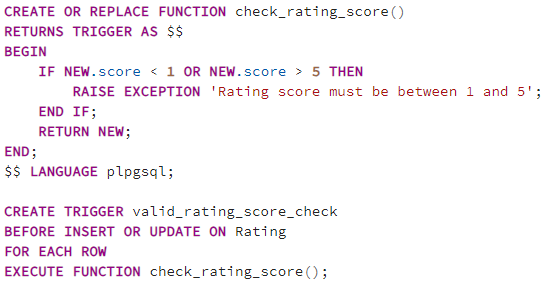


Рисунок 4.9 – Реализация триггера valid\_rating\_score\_check

Оценка книги должна находиться в диапазоне от 1 до 5.

**4.3.3** Триггер positive\_subscription\_price\_check

Триггер positive\_subscription\_price\_check реализован для того, чтобы отслеживать, что значение стоимости подписки не может быть меньше нуля. Реализация данного триггера представлена на рисунке 4.10.

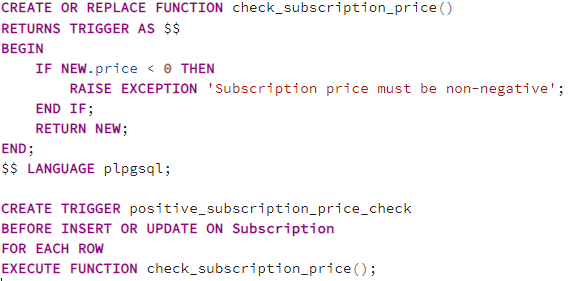


Рисунок 4.10 – Реализация триггера positive\_subscription\_price\_check

Наличие данного триггера поможет предотвратить случайные ошибки, связанные с выставлениями стоимостей за подписки на сайт.

**4.3.4** Триггер trg\_check\_active\_subscription

Триггер trg\_check\_active\_subscription позволяет автоматически активна ли подписка читателя перед взятием книги. Реализация данного триггера представлена на рисунке 4.11.

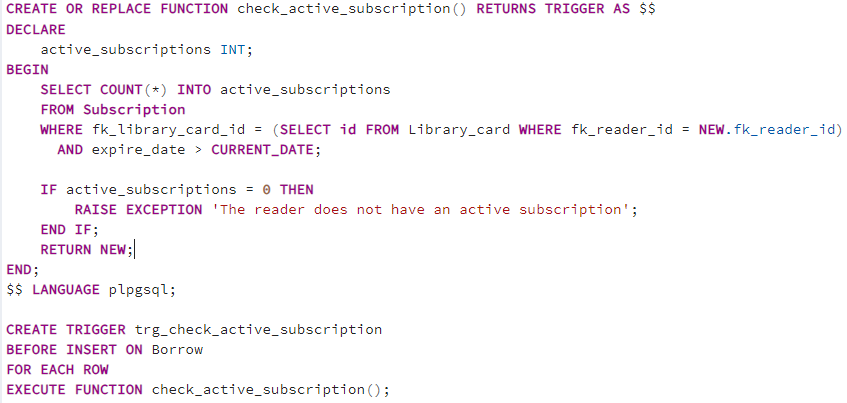


Рисунок 4.11 – Реализация триггера trg\_check\_active\_subscription

Данный триггер осуществляет автоматическую проверку наличия актуальной подписки у читателя, что позволяет постоянно не перепроверять это вручную при выдаче книг.

## 4.4 Создание функций

Функция в PostgreSQL – это именованный блок кода, выполняющий задачу, принимающий (необязательно) параметры, возвращающий (необязательно) результат, написанный на SQL, PL/pgSQL или других языках, и предназначенный для переиспользования кода.

**4.4.1** Функция check\_unique\_isbn

Функция check\_unique\_isbn предназначена для того, чтобы проверить уникальность ISBN номеров для экземпляров книг. Реализация данной функции представлено на рисунке 4.13.

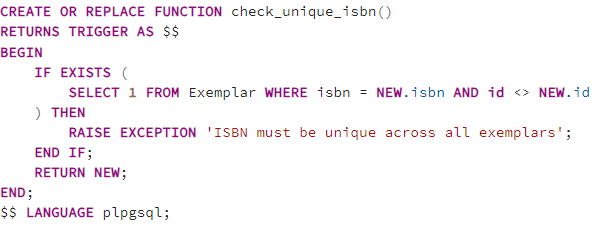
****

Рисунок 4.13 – Реализация функции check\_unique\_isbn

Функция check\_unique\_isbn повышает читаемость кода и понижает вероятность ошибки при необходимости проверить уникальность ISBN. Также эта функция применяется совместно с триггером, срабатывающим на вставку и обновление таблицы экземпляров, что позволяет дополнительно автоматизировать процесс проверки уникальности ISBN.

**4.4.2** Функция update\_fine\_on\_overdue

Функция update\_fine\_on\_overdue реализована с целью добавления штрафа в случае, если просрочена дата возврата книги. Реализация данной функции представлена на рисунке 4.14.

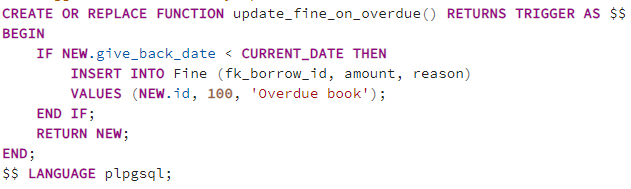


Рисунок 4.14 – Реализация функции update\_fine\_on\_overdue

Функция update\_fine\_on\_overdue реализована также с целью автоматизации начисления штрафов, поэтому она также работает совместно с триггером, который вызывает ее в нужный момент.

## 4.5 Создание индексов

Для таблиц book, reader, exemplar, reader и borrow были созданы следующие индексы, представленные на рисунке 4.15.

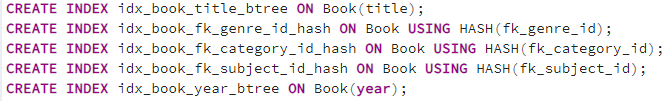


Рисунок 4.15 – Реализация индексов для таблиц, связанных с книгой

Поскольку принципы работы и реализации всех индексов схожи, и они выполняют одну и ту же роль, описание остальных индексов не является необходимым.

# Тестирование работоспособности базы              данных

После реализации физической модели БД путем написания SQL-запросов, необходимо протестировать базу данных и убедиться, что все работает корректно. Для этого запустим некоторые процедуры.

Все тесты были выполнены с использованием SQL-запросов, ошибки или успешное выполнение которых зафиксированы на рисунках:

При некорректном вводе оценки книги выводится сообщение об ошибке. Демонстрация ошибки представлена на рисунке 5.1.



Рисунок 5.1 – Ошибка при некорректном вводе оценки книги

При вводе отрицательного числа в стоимость подписки также выводится ошибка. Демонстрация ошибки представлена на рисунке 5.2.



Рисунок 5.2 – Ошибка при некорректном вводе стоимости подписки

Добавление новых оценок на книги при помощи процедур проходит успешно. Демонстрация успешного добавления новых оценок представлена на рисунке 5.3.

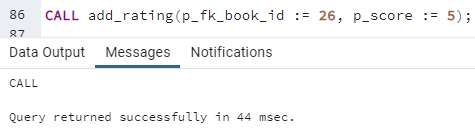


Рисунок 5.3 – Демонстрация успешного добавления новых оценок

Успешное появление добавленных оценок продемонстрировано на рисунке 5.4.

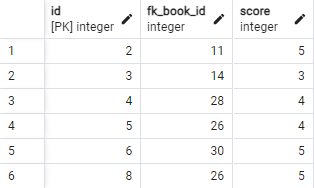


Рисунок 5.4 – Успешное добавление оценок в таблицу

Если в таблице уже есть экземпляр с определенным номером ISBN, то добавить еще один с таким же номером не представляется возможным, так как ISBN уникальны для каждого экземпляра. Демонстрация ошибки представлена на рисунке 5.5.



Рисунок 5.5 – Ошибка при попытке добавить экземпляр с уже существующим ISBN

Доступность представляет собой количество экземпляров, хранимых в библиотеке, поэтому его атрибут количества не может быть отрицательным. Демонстрация ошибки при попытке добавления отрицательного значения в сущность доступности представлена на рисунке 5.6.



Рисунок 5.6 – Демонстрация ошибки при добавлении отрицательного значения в доступность

В результате тестирования базы данных была выполнена проверка всех таблиц, триггеров, функций и индексов. База данных была заполнена тестовыми данными, и были проведены проверки работы всех компонентов. Все операции прошли успешно, что подтверждает готовность базы данных к эксплуатации.

Заключение

В рамках курсового проекта была разработана база данных, предназначенная для эффективного управления библиотечными данными. Актуальность данного проекта обусловлена возрастающими объемами информации, которыми оперируют библиотеки, и необходимостью обеспечения их эффективной обработки и управления для улучшения обслуживания читателей и оптимизации внутренних процессов.

В результате проведенных исследований и проектирования была создана реляционная база данных, удовлетворяющая требованиям к хранению и обработке данных о книгах, авторах, читателях, библиотечных фондах и связанных сущностях.

В соответствии с поставленными задачами, были выполнены следующие ключевые этапы:

1 Определены основные сущности предметной области (книги, авторы, жанры, читатели, экземпляры и т.д.) и установлены их взаимосвязи.

2 Проведена нормализация БД до третьей нормальной формы (3NF), что обеспечивает устранение избыточности данных и минимизацию рисков, связанных с аномалиями вставки, обновления и удаления данных.

3 Сформирован набор SQL-запросов для выборки, вставки, обновления и удаления данных, обеспечивающих полноценное взаимодействие с БД.

4 Разработаны индексы для наиболее часто используемых таблиц, что способствует ускорению выполнения поисковых запросов и повышению общей производительности системы.

5 Созданы триггеры и хранимые процедуры, автоматизирующие логику и обеспечивающие целостность данных при их изменении, например, при выдаче или возврате книг.

Реализация поставленных задач позволила создать функциональный инструмент для эффективного управления данными в библиотеке. Разработанная база данных предоставляет удобный и надежный механизм для хранения и обработки информации, обеспечивая возможности для поиска, учета и анализа данных, повышая эффективность библиотечных операций и удобство для читателей.

Проект демонстрирует потенциал использования современных технологий управления данными для повышения качества обслуживания в библиотеках, эффективности их работы и удовлетворенности читателей.

Результаты исследования и разработанная база данных могут служить основой для дальнейшей разработки информационной системы, обеспечивающей автоматизацию ключевых процессов сети библиотек, включая управление каталогизацию, обслуживание читателей и отчетность.

Список литературных источников

[1] СТП 01-2024 – Стандарт предприятия. Дипломные проекты (работы) общие требования. – Минск: БГУИР, 2024. – 175с.

[2] Основные функции и объекты СУБД [Электронный ресурс] – Режим   
доступа : https://sky.pro/wiki/sql/osnovnye-funkcii-i-obuekty-subd/. – Дата доступа : 15.09.2024

[3] Классификация СУБД [Электронный ресурс] – Режим доступа : https://lms.kgeu.ru/mod/book/view.php?id=77952&chapterid=3854. – Дата доступа : 16.09.2024

[4] Разница между физической и логической моделью данных [Электронный ресурс] – Режим доступа : https://aws.amazon.com/compare/the-difference-between-logical-and-physical-data-model/. – Дата доступа : 16.09.2024

[5] Нормализация отношений. Шесть нормальных форм [Электронный ресурс] – Режим доступа : https://habr.com/ru/articles/254773/. – Дата доступа : 19.09.2024

[6] Национальная библиотека Республики Беларусь [Электронный ресурс] – Режим доступа : https://www.nlb.by/. – Дата доступа : 19.09.2024

[7] LibGen [Электронный ресурс] – Режим доступа : https://libgen.is/. – Дата доступа : 19.09.2024

[8] Библиотека БГУИР [Электронный ресурс] – Режим доступа : https://library.bsuir.by/. – Дата доступа : 19.09.2024

[9] WorldCat [Электронный ресурс] – Режим доступа : https://search.worldcat.org/. – Дата доступа : 19.09.2024

[10] Виды, особенности, примеры баз данных [Электронный ресурс] – Режим доступа : https://blog.ishosting.com/ru/types-and-features-of-databases. – Дата доступа : 27.09.2024

[11] Базы данных «ключ-значение» [Электронный ресурс] – Режим доступа : https://habr.com/ru/companies/vk/articles/685402/. – Дата доступа : 30.09.2024

[12] Графовые базы данных [Электронный ресурс] – Режим доступа : https://habr.com/ru/companies/neoflex/articles/654267/. – Дата доступа : 30.09.2024

[13] Реляционные БД [Электронный ресурс] – Режим доступа : https://itglobal.com/ru-ru/company/glossary/relyaczionnaya-baza-dannyh/. – Дата доступа : 10.10.2024

[14] PostgreSQL [Электронный ресурс] – Режим доступа : https://www.ibm.com/topics/postgresql. – Дата доступа : 11.10.2024

[15] Миграция данных [Электронный ресурс] – Режим доступа : https://www.astera.com/ru/type/blog/database-migration-what-it-is-and-how-it-is-done/. – Дата доступа : 19.10.2024

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг кода

Листинг А.1 – SQL-код файла tables\_reation.sql

CREATE TABLE Genre (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

-- Table: Category

CREATE TABLE Category (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

-- Table: Subject

CREATE TABLE Subject (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

-- Table: Publisher

CREATE TABLE Publisher (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

city VARCHAR(255),

country VARCHAR(255),

email VARCHAR(255)

);

-- Table: Author

CREATE TABLE Author (

id SERIAL PRIMARY KEY,

last\_name VARCHAR(255) NOT NULL,

first\_name VARCHAR(255) NOT NULL,

birthdate DATE,

country VARCHAR(255)

);

-- Table: Book

CREATE TABLE Book (

id SERIAL PRIMARY KEY,

fk\_subject\_id INT NOT NULL REFERENCES Subject(id),

fk\_genre\_id INT NOT NULL REFERENCES Genre(id),

fk\_category\_id INT NOT NULL REFERENCES Category(id),

title VARCHAR(255) NOT NULL,

page\_count INT,

year INT

);

-- Table: Author\_Book

CREATE TABLE Author\_Book (

id SERIAL PRIMARY KEY,

fk\_book\_id INT NOT NULL REFERENCES Book(id),

fk\_author\_id INT NOT NULL REFERENCES Author(id)

);

-- Table: Form

CREATE TABLE Form (

id SERIAL PRIMARY KEY,

fk\_book\_id INT NOT NULL REFERENCES Book(id),

name VARCHAR(255) NOT NULL

);

-- Table: Format

CREATE TABLE Format (

id SERIAL PRIMARY KEY,

fk\_form\_id INT NOT NULL REFERENCES Form(id),

name VARCHAR(255) NOT NULL

);

-- Table: Language

CREATE TABLE Language (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

code VARCHAR(255)

);

-- Table: Book\_Language

CREATE TABLE Book\_Language (

id SERIAL PRIMARY KEY,

fk\_book\_id INT NOT NULL REFERENCES Book(id),

fk\_language\_id INT NOT NULL REFERENCES Language(id)

);

-- Table: Tag

CREATE TABLE Tag (

id SERIAL PRIMARY KEY,

name VARCHAR NOT NULL

);

-- Table: Book\_Tag

CREATE TABLE Book\_Tag (

id SERIAL PRIMARY KEY,

fk\_book\_id INT NOT NULL REFERENCES Book(id),

fk\_tag\_id INT NOT NULL REFERENCES Tag(id)

);

-- Table: Availability

CREATE TABLE Availability (

id SERIAL PRIMARY KEY,

quantity INT NOT NULL

);

-- Table: Exemplar

CREATE TABLE Exemplar (

id SERIAL PRIMARY KEY,

fk\_book\_id INT NOT NULL REFERENCES Book(id),

fk\_availability\_id INT NOT NULL REFERENCES Availability(id),

fk\_publisher\_id INT NOT NULL REFERENCES Publisher(id),

isbn INT UNIQUE NOT NULL,

year INT

);

-- Table: Library

CREATE TABLE Library (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

open\_hours VARCHAR(255),

address VARCHAR(255) NOT NULL,

phone VARCHAR(255)

);

-- Table: Position

CREATE TABLE Position (

id SERIAL PRIMARY KEY,

fk\_library\_id INT NOT NULL REFERENCES Library(id),

fk\_availability\_id INT NOT NULL REFERENCES Availability(id),

section INT,

rack INT,

shelf INT

);

-- Table: Reading\_room

CREATE TABLE Reading\_room (

id SERIAL PRIMARY KEY,

fk\_library\_id INT NOT NULL REFERENCES Library(id),

name VARCHAR(255) NOT NULL,

open\_hours VARCHAR(255),

seats\_amount INT,

phone VARCHAR(255)

);

-- Table: Identification\_card

CREATE TABLE Identification\_card (

id SERIAL PRIMARY KEY,

personal\_number INT UNIQUE NOT NULL,

job\_title VARCHAR(255),

expire\_date DATE

);

-- Table: Librarian

CREATE TABLE Librarian (

id SERIAL PRIMARY KEY,

fk\_identity\_card\_id INT NOT NULL REFERENCES Identification\_card(id),

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL

);

-- Table: Reader

CREATE TABLE Reader (

id SERIAL PRIMARY KEY,

nickname VARCHAR(255),

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

email VARCHAR(255),

phone VARCHAR(255)

);

-- Table: Privilege

CREATE TABLE Privilege (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT

);

-- Table: Library\_card

CREATE TABLE Library\_card (

id SERIAL PRIMARY KEY,

fk\_reader\_id INT NOT NULL REFERENCES Reader(id),

fk\_privilege\_id INT NOT NULL REFERENCES Privilege(id),

card\_number INT UNIQUE NOT NULL,

receipt\_date DATE NOT NULL

);

-- Table: Type

CREATE TABLE Type (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT

);

-- Table: Subscription

CREATE TABLE Subscription (

id SERIAL PRIMARY KEY,

fk\_library\_card\_id INT NOT NULL REFERENCES Library\_card(id),

fk\_type\_id INT NOT NULL REFERENCES Type(id),

name VARCHAR(255) NOT NULL,

price INT,

expire\_date DATE

);

-- Table: Borrow

CREATE TABLE Borrow (

id SERIAL PRIMARY KEY,

fk\_exemplar\_id INT NOT NULL REFERENCES Exemplar(id),

fk\_reader\_id INT NOT NULL REFERENCES Reader(id),

fk\_librarian\_id INT NOT NULL REFERENCES Librarian(id),

get\_date DATE NOT NULL,

give\_back\_date DATE NOT NULL

);

-- Table: Fine

CREATE TABLE Fine (

id SERIAL PRIMARY KEY,

fk\_borrow\_id INT NOT NULL REFERENCES Borrow(id),

amount INT NOT NULL,

reason VARCHAR(255)

);

-- Table: Rating

CREATE TABLE Rating (

id SERIAL PRIMARY KEY,

fk\_book\_id INT NOT NULL REFERENCES Book(id),

score INT NOT NULL

);

-- Table: Review

CREATE TABLE Review (

id SERIAL PRIMARY KEY,

fk\_reader\_id INT NOT NULL REFERENCES Reader(id),

fk\_rating\_id INT NOT NULL REFERENCES Rating(id),

date DATE NOT NULL,

text TEXT

);

Листинг А.2 – SQL-код файла usage.sql

-- Вывести список книг с жанром и категорией

SELECT

Book.title AS "Название книги",

Genre.name AS "Жанр",

Category.name AS "Категория"

FROM

Book

JOIN Genre ON Book.fk\_genre\_id = Genre.id

JOIN Category ON Book.fk\_category\_id = Category.id;

--Посчитать общее количество разных (не кол-ва доступных экземпляров) книг в каждой библиотеке

SELECT

Library.name AS "Библиотека",

COUNT(Position.id) AS "Количество книг"

FROM

Library

JOIN Position ON Library.id = Position.fk\_library\_id

GROUP BY

Library.name;

--Вывести информацию о читателях, их книгах и библиотекарях

SELECT

Reader.first\_name AS "Имя читателя",

Reader.last\_name AS "Фамилия читателя",

Book.title AS "Книга",

Librarian.first\_name AS "Имя библиотекаря",

Librarian.last\_name AS "Фамилия библиотекаря"

FROM

Borrow

JOIN Reader ON Borrow.fk\_reader\_id = Reader.id

JOIN Exemplar ON Borrow.fk\_exemplar\_id = Exemplar.id

JOIN Book ON Exemplar.fk\_book\_id = Book.id

JOIN Librarian ON Borrow.fk\_librarian\_id = Librarian.id;

SELECT

Book.title AS "Название книги",

Publisher.name AS "Издатель",

Language.name AS "Язык"

FROM

Book

JOIN Exemplar ON Book.id = Exemplar.fk\_book\_id

JOIN Publisher ON Exemplar.fk\_publisher\_id = Publisher.id

JOIN Book\_Language ON Book.id = Book\_Language.fk\_book\_id

JOIN Language ON Book\_Language.fk\_language\_id = Language.id

WHERE

Book.year > 2000;

WITH ReaderFines AS (

SELECT

Reader.id AS reader\_id,

Reader.first\_name,

Reader.last\_name,

SUM(Fine.amount) AS total\_fine

FROM

Reader

JOIN Borrow ON Reader.id = Borrow.fk\_reader\_id

JOIN Fine ON Borrow.id = Fine.fk\_borrow\_id

GROUP BY Reader.id

)

SELECT

first\_name AS "Имя",

last\_name AS "Фамилия",

total\_fine AS "Общая сумма штрафов"

FROM

ReaderFines

WHERE

total\_fine > 100;

SELECT \* FROM review;

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема базы данных

ПРИЛОЖЕНИЕ В

(обязательное)

Ведомость документов