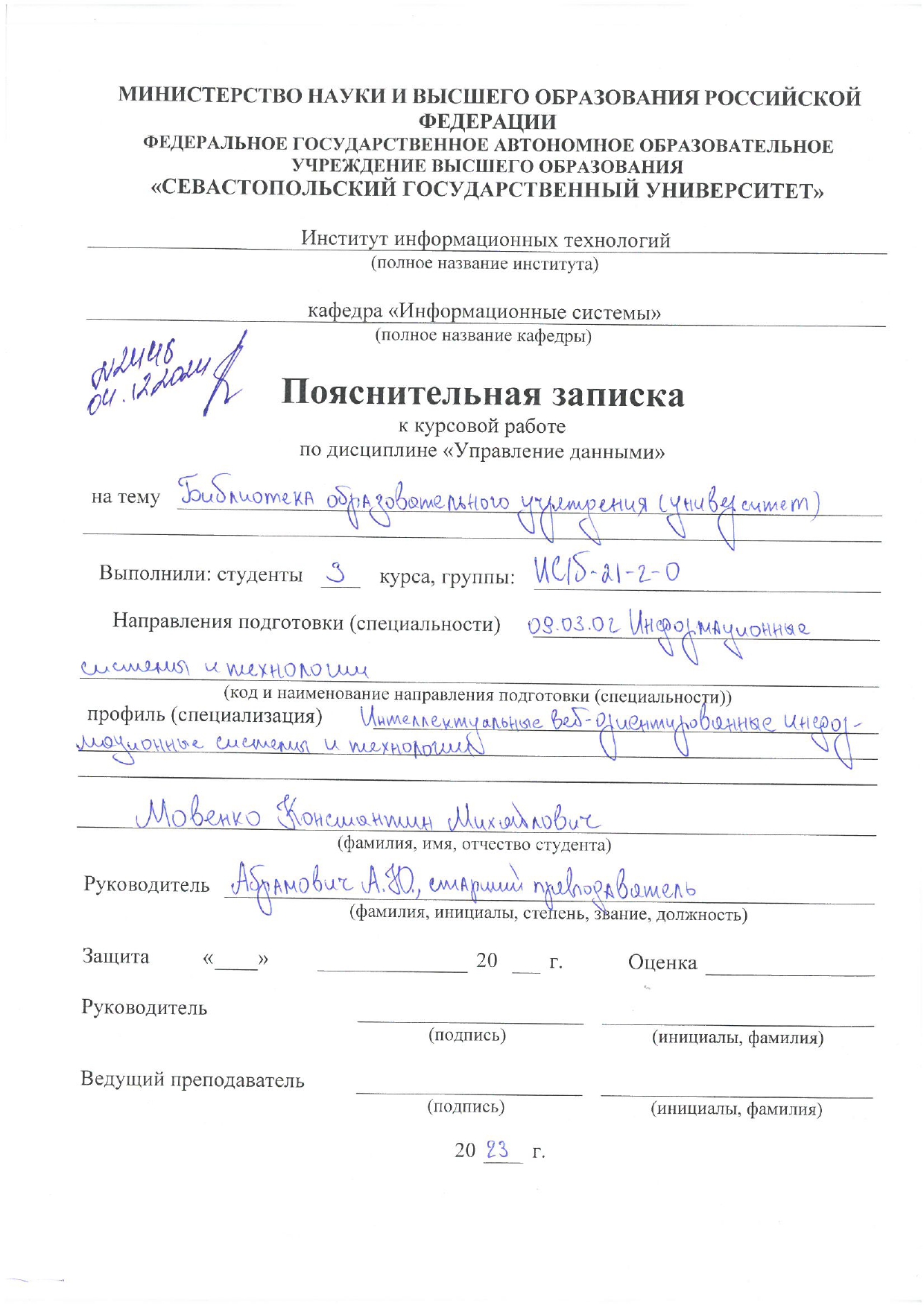
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | |  | | |  | | | | | | | | |  | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |  | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | |  | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | |  | | | | | | | | | | | | | | | | | | | | | |
|  | | |  | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | |  | |  | |  | | | | | | | |  | |  | |  |  | | | |  | |
|  | | | | | | |  | | | |  |  | | | | | | |  | | |  | | |
|  | | | | | | | | |  | | | | | | | | | | |  |  | | | |
|  | | | | | | | |  | | | | | | | | | | | |  |  | | | |
|  | | | | | | |  | | | |  |  | | | | | | |  | | |  | | |
|  | | | | | | | | |  | | | | | | | | | | |  |  | | | |
|  | | | | | | | |  | | | | | | | | | | | |  |  | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | |  | |  | | | | | | | | |

****

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc158305178)

[1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 4](#_Toc158305179)

[1.1 Анализ предметной области (концептуальное моделирование) 4](#_Toc158305180)

[1.2 Постановка задачи 4](#_Toc158305181)

[2 РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 6](#_Toc158305182)

[2.1 Построение диаграммы «сущность-связь» в нотации П. Чена 6](#_Toc158305183)

[2.2 Построение модели, основанной на ключах, и полной атрибутивной модели в нотации IDEF1X 7](#_Toc158305184)

[3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 9](#_Toc158305185)

[3.1 Выбор аппаратной и программной платформы и реализация БД 9](#_Toc158305186)

[3.2 Тестирование базы данных (создание и реализация запросов) 9](#_Toc158305187)

[3.3 Разграничение прав доступа 18](#_Toc158305188)

[4. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 19](#_Toc158305189)

[4.1 Обоснование выбора языка программирования 19](#_Toc158305190)

[4.2 Разработка интерфейса пользователя 19](#_Toc158305191)

[4.3 Алгоритм работы каждого из модулей 19](#_Toc158305192)

[4.4 Тестирование работы приложения 19](#_Toc158305193)

[ЗАКЛЮЧЕНИЕ 20](#_Toc158305194)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc158305195)

[ПРИЛОЖЕНИЕ А 22](#_Toc158305196)

[ПРИЛОЖЕНИЕ Б 25](#_Toc158305197)

# ВВЕДЕНИЕ

В настоящее время информационные системы являются неотъемлемой частью управления деятельностью множества организаций и учреждений. Создание эффективной базы данных позволяет автоматизировать процессы управления, повысить эффективность работы и сократить время на выполнение рутинных задач. В рамках данной работы была поставлена задача разработки базы данных для библиотеки образовательного учреждения.

Целью работы является проектирование и реализация базы данных, а также разработка пользовательского приложения для её использования. В базе данных должны храниться данные о хранимых книгах, читательских билетах и истории займа и возврата книг. Благодаря этому библиотека сможет эффективнее вести учёт имущества и своих клиентов.

В ходе работы будут рассмотрены логическая и физическая модели базы данных, а также способы реализации связей между таблицами. Будет создано приложение для управления базой данных и предоставления доступа к её функционалу. Результатом работы будет готовая база данных и работающее приложение, которые можно использовать для ведения учёта в библиотеке образовательного учреждения.

# 1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

## Анализ предметной области (концептуальное моделирование)

Студенческая библиотека – это один из видов библиотек, ориентированный на студентов высших учебных заведений. В стандартной библиотеке читатель может бесплатно взять на временное пользование книгу из специального архива, после чего он должен вернуть её обратно в библиотеку. Если читатель не возвращает книгу вовремя, на него накладываются санкции, в том числе материального характера.

Особенностью студенческой библиотеки является то, что её основные клиент – студент, который нуждается не столько в художественной, сколько в научной литературе.

Для разработки базы данных и приложения библиотеки были изучены документы, относящиеся к процессу обучения и организации работы библиотечных учреждений. Кроме того, было проведено общение со специалистами данной предметной области, в результате которого были получены ценные сведения о особенностях функционирования библиотек.

## Постановка задачи

Были определены основные объекты предметной области:

* Книги (название, автор, год, издатель, тематика и др.);
* Люди (ФИО, статус в университете, контакты и др.);
* Выдачи (читатель, книга, дата выдачи, дата возврата и др.);

Связи между объектами:

* Каждая запись о выдаче книги связывает соответствующую книгу с читателем;
* Каждой книге соответствует 1 или более авторов;

Были определены группы пользователей и их набор используемых ими операций:

1. Библиотекарь (администратор):

* Операции: учёт книг в базе данных, регистрация новых пользователей, оформление выдачи и возврата книг;
* Выводимые данные: вся информация о книгах, список читателей и должников, история выдачи и возврата книг;

1. Читатель:

* Операции: поиск книг по различным критериям, запрос на выдачу книги или продление срока возврата;
* Выводимые данные: информация о доступных книгах, информация о выданных читателю книгах, сроках возврата и задолженностях;

Сценарий пользователя: студента регистрирует библиотекарь в системе, после чего он может войти в свой аккаунт и увидеть все доступные в библиотеке книги. После этого он может очно подойти к ответственному за выдачу книг сотруднику библиотеки, который может назначить выдачу необходимой книги на определенный срок, который библиотекарь может скорректировать исходя из обстоятельств. Если за студентом есть долги в виде вовремя не сданных книг, то он не может взять новую. После успешно оформленной выдачи читатель может в любое время до её окончания сдать книгу обратно. Если он не сдает книгу вовремя, то на него накладываются санкции – он не сможет получить новую книгу, а также может получить предупреждение от деканата. На каждую книгу формируется своя выдача, после окончания (закрытия) выдачи, она остается в базе данных, что позволяет отслеживать историю как книг, так и читателей.

# 2 РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ

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

После проведения анализа предметной области и выделения ключевых сущностей стало возможным создание диаграммы «сущность-связь» в нотации П. Чена.

Перед этим были определены отношения между выделенными сущностями. Была разработана сложная сетевая структура, основанная на основных элементах предметной области (Рисунок 2.1).

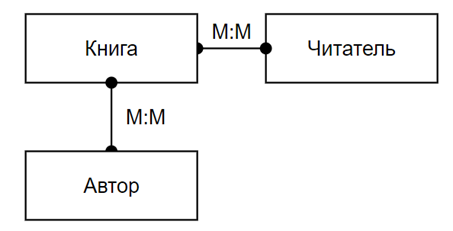


Рисунок 2.1 – Сложная сетевая структура

На основе неё была разработана упрощенная сетевая структура. Она была получена путём введения промежуточных таблиц для преобразования связи «многие ко многим» в связь «один ко многим» (Рисунок 2.2).

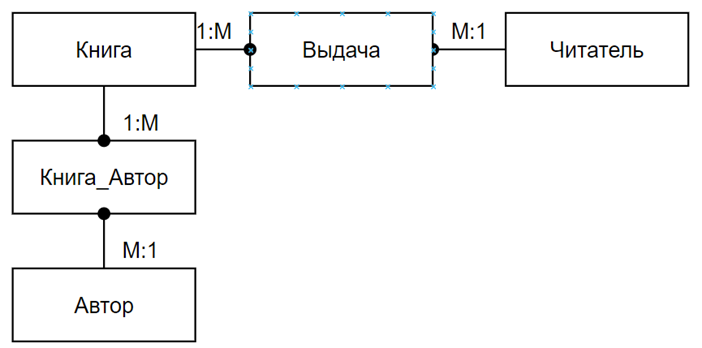


Рисунок 2.2 – Простая сетевая структура

На основе предыдущих таблиц была создана ER-диаграмма (Рисунок 2.3), которая иллюстрирует взаимосвязь между объектами и основные элементы их взаимодействия.

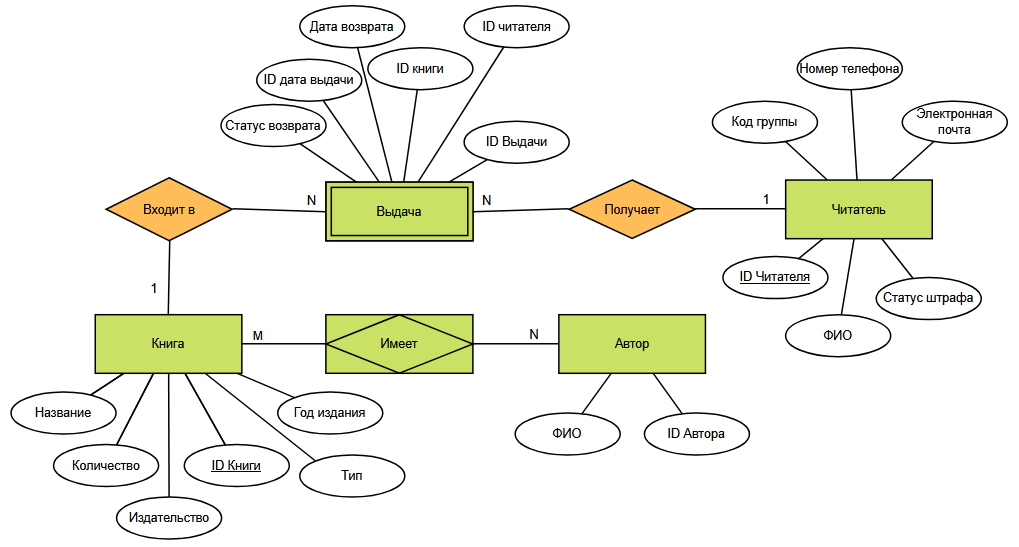


Рисунок 2.3 – ER**-**диаграмма в нотации П. Чена

## 2.2 Построение модели, основанной на ключах, и полной атрибутивной модели в нотации IDEF1X

Основной целью модели, основанной на ключах, является широкий обзор структур данных и ключей, нужных для поддержки определенной предметной области.

В модель были включены все упомянутые сущности, их первичные и вторичные ключи, а также атрибуты (Рисунок 2.4). Для достижения полной атрибутивной модели структура базы данных была приведена к третьей нормальной, была устранена избыточность данных.

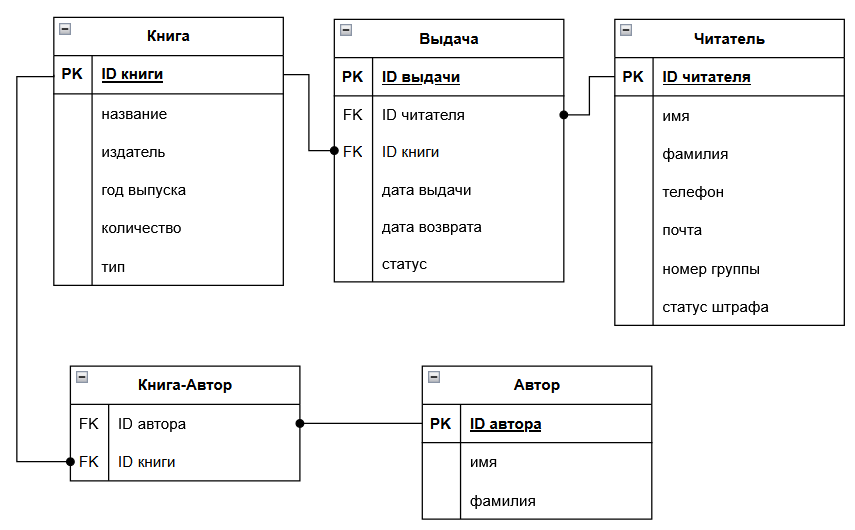


Рисунок 2.4 – Полная атрибутивная модель в нотации IDF1X

Т.к. для первой нормальной формы достаточно, чтобы все атрибуты были атомарными, имеющаяся структура удовлетворяет ей. Всё связи «многие ко многим» выполнены через ассоциативные сущности.

Был проведён анализ на соответствие второй нормальной форме. Первичные ключи в каждой сущности полностью определяют каждый не ключевой атрибут. Таким образом, база данных соответствует второй нормальной форме.

Продолжая анализ в соответствии с третьей нормальной формой, подтверждено отсутствие транзитивных зависимостей в каждой сущности. Это говорит о том, что база данных находится в третьей нормальной форме.

# 3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ

## 3.1 Выбор аппаратной и программной платформы и реализация БД

Для реализации базы данных библиотеки была выбрана СУБД PostgreSQL. Данный выбор сделан в результате сравнительного анализа с аналогичными СУБД: Firebird и MySQL.

PostgreSQL была выбрана, так как имеет ряд преимуществ по сравнению с другими СУБД, в частности, более широкие возможности по работе с большими объёмами данных, высокую производительность и продвинутый механизм обработки транзакций. Также был учтён положительный опыт работы в графическом клиенте pgAdmin, облегчающем администрирование и разработку баз данных PostgreSQL.

Для реализации физической схемы базы данных были использованы стандартные типы данных, поддерживаемые PostgreSQL: INTEGER, VARCHAR, DATE и др. Тип данных для каждого атрибута был выбран на основе его предназначения и возможностей, предоставляемых СУБД.

SQL запросы, использованные при создании физической модели базы данных, приведены в приложении А.

## 3.2 Тестирование базы данных (создание и реализация запросов)

В соответствии с предметной областью и структурой разработанной базы данных были составлены тестовые наборы данных (Рисунки 3.1-3.3).

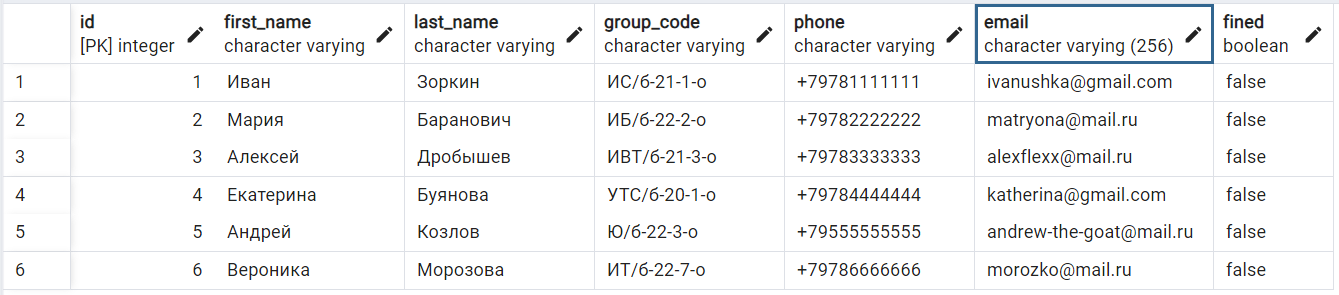


Рисунок 3.1 – Тестовые данные для таблицы «Читатели»

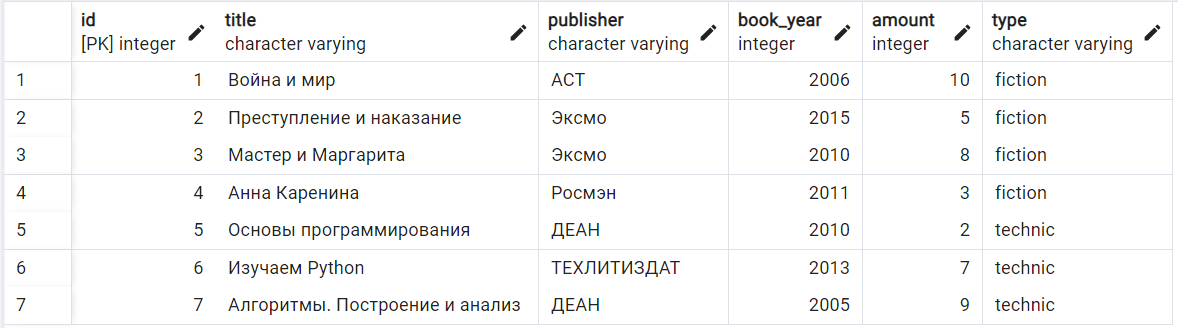


Рисунок 3.2 – Тестовые данные для таблицы «Книги»

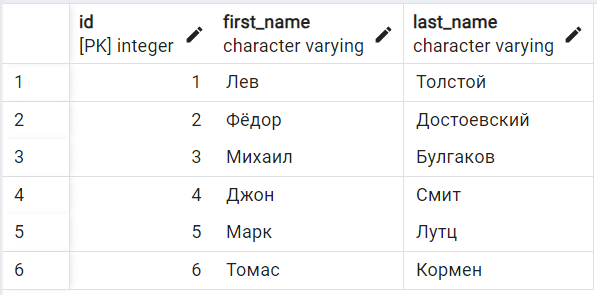


Рисунок 3.3 – Тестовые данные для таблицы «Авторы»

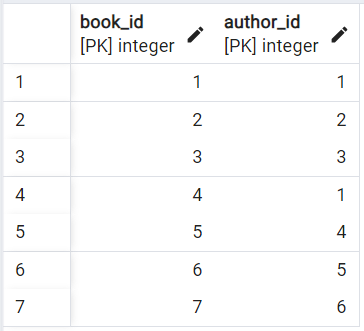


Рисунок 3.4 – Тестовые данные для таблицы «Книга-Автор»

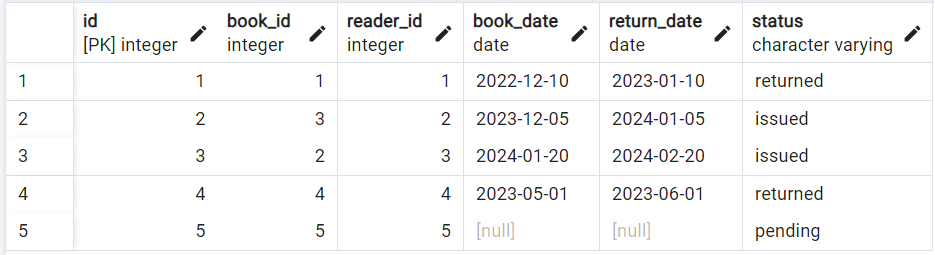


Рисунок 3.5 – Тестовые данные для таблицы «Выдача»

Полученная база данных была протестирована путём проведения 15-ти запросов.

Запрос 1: вывести имена и группы всех читателей библиотеки (текст запроса – Листинг 3.1, результат – Рисунок 3.6).

Листинг 3.1 – Текст запроса 1

select first\_name, last\_name, group\_code

from readers;

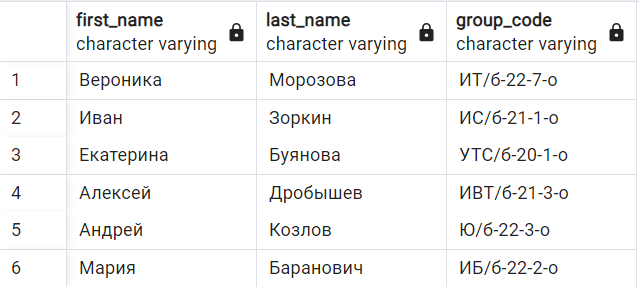


Рисунок 3.6 – Результат запроса 1

Запрос 2: вывести все книги, тип которых – художественная литература (текст запроса – Листинг 3.2, результат – Рисунок 3.7).

Листинг 3.2 – Текст запроса 2

select title, publisher, amount

from books

where type = 'fiction';

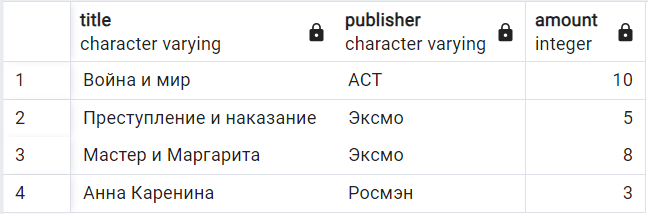


Рисунок 3.7 – Результат запроса 2

Запрос 3: подсчитать число книг, выпущенных до 2010 года (текст запроса – Листинг 3.3, результат – Рисунок 3.8).

Листинг 3.3 – Текст запроса 3

select count(\*) from books

where book\_year < 2010;

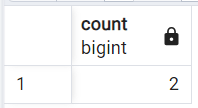


Рисунок 3.8 – Результат запроса 3

Запрос 4: заменить имя четвёртого автора в таблице «Авторы» на ‘Игорь Черпаков’ (текст запроса – Листинг 3.4, результат – Рисунок 3.9).

Листинг 3.4 – Текст запроса 4

update authors

set first\_name = 'Игорь',

last\_name = 'Черпаков'

where id = 4;



Рисунок 3.9 – Результат запроса 4

Запрос 5: вывести все записи, срок возврата которых истёк к моменту запроса (текст запроса – Листинг 3.5, результат – Рисунок 3.10).

Листинг 3.5 – Текст запроса 5

select \* from issuance

where return\_date < current\_date

and status != 'returned';



Рисунок 3.10 – Результат запроса 4

Запрос 6: установить штраф всем читателям, которые на текущий момент не вернули книгу в положенный срок (текст запроса – Листинг 3.6, результат – Рисунок 3.11).

Листинг 3.6 – Текст запроса 4

update readers

set fined = True

where id in

(select reader\_id

from issuance

where return\_date < current\_date

and status != 'returned');

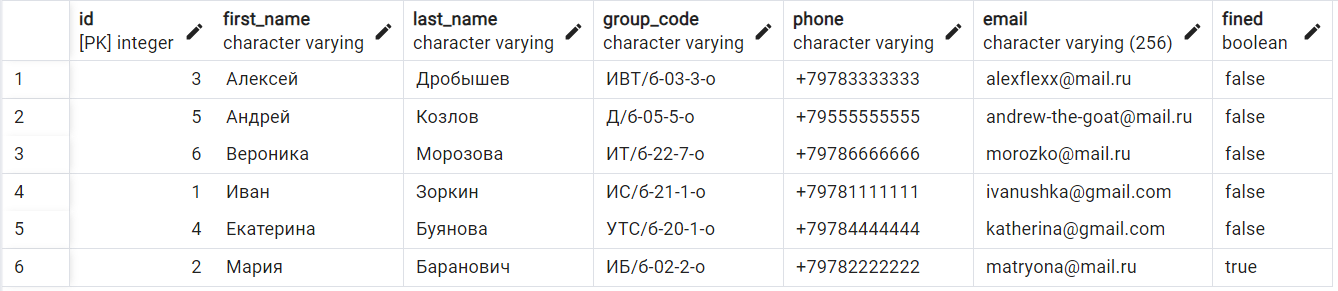


Рисунок 3.11 – Результат запроса 6

Запрос 7: вывести для каждой книги список её авторов (текст запроса – Листинг 3.7, результат – Рисунок 3.12).

Листинг 3.4 – Текст запроса 7

select books.title, string\_agg(authors.last\_name, ', ') as authors

from books

join book\_author on books.id = book\_author.book\_id

join authors on book\_author.author\_id = authors.id

group by books.title;

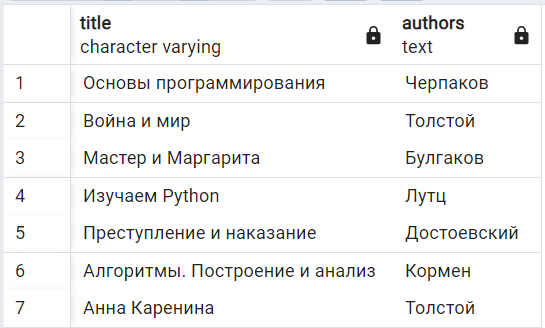


Рисунок 3.12 – Результат запроса 7

Запрос 8: рассмотреть имеющуюся заявку на выдачу – установить дату выдачи, дату возврата и обновить статус (текст запроса – Листинг 3.8, результат – Рисунок 3.13).

Листинг 3.4 – Текст запроса 8

update issuance

set book\_date = current\_date,

return\_date = current\_date + interval '1 month',

status = 'issued'

where id = 5;



Рисунок 3.13 – Результат запроса 8

Запрос 9: добавить в таблицу «Выдачи» заявку на выдачу читателю книги (текст запроса – Листинг 3.9, результат – Рисунок 3.14).

Листинг 3.4 – Текст запроса 9

insert into issuance (book\_id, reader\_id) values (5, 1);

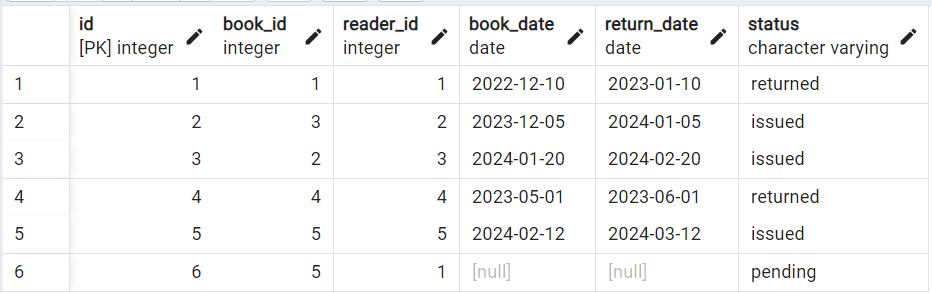


Рисунок 3.14 – Результат запроса 9

Запрос 10: вывести количество выданных книг типа «Художественная литература» (текст запроса – Листинг 3.10, результат – Рисунок 3.15).

Листинг 3.4 – Текст запроса 10

select count(\*)

from issuance

where status = 'issued'

and exists

(select \*

from books

where books.id = issuance.book\_id

and type = 'fiction');

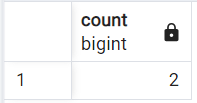


Рисунок 3.15 – Результат запроса 10

Запрос 11: вывести количество книг автора с ID=4, выданных на данный момент (текст запроса – Листинг 3.11, результат – Рисунок 3.16).

Листинг 3.4 – Текст запроса 11

select count(\*)

from issuance

where status = 'issued'

and exists

(select \*

from books

join book\_author as BA

on books.id = BA.book\_id

where BA.author\_id = 4

and books.id = issuance.book\_id

);

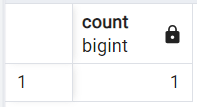


Рисунок 3.16 – Результат запроса 11

Запрос 12: вывести список книг с их общим количеством и количеством книг, доступных для выдачи на данный момент (текст запроса – Листинг 3.12, результат – Рисунок 3.17).

Листинг 3.4 – Текст запроса 12

select books.id, title, amount,

amount - coalesce(issued.total\_count, 0) as available

from books

left join (

select book\_id, count(\*) as total\_count

from issuance

where status = 'issued'

group by book\_id

) as issued

on books.id = issued.book\_id;

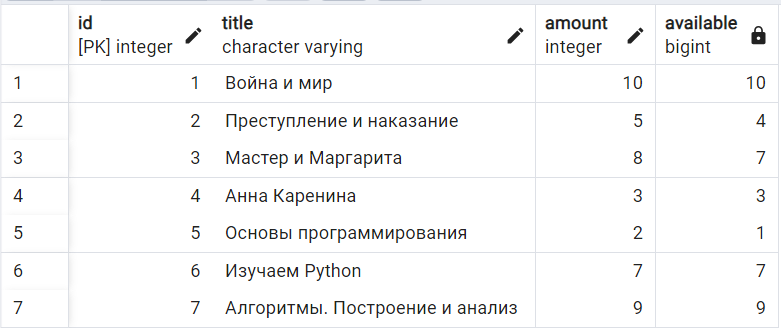


Рисунок 3.17 – Результат запроса 12

Запрос 13: вывести все книги, доступные для выдачи и никем не забронированные (текст запроса – Листинг 3.13, результат – Рисунок 3.18).

Листинг 3.13 – Текст запроса 13

select id, title, book\_year, publisher, type

from books

left join (

select book\_id, count(\*) as total\_count

from issuance

where status in ('issued', 'pending')

group by book\_id

) as issued

on books.id = issued.book\_id

where books.amount - coalesce(issued.total\_count, 0) > 0;



Рисунок 3.18 – Результат запроса 13

Запрос 14: вывести все невозвращённые книги вместе с именами их читателей (текст запроса – Листинг 3.14, результат – Рисунок 3.19).

Листинг 3.14 – Текст запроса 14

select first\_name, last\_name, group\_code,

book\_id, book\_date, return\_date

from issuance

join readers

on readers.id = issuance.reader\_id

where status = 'issued';

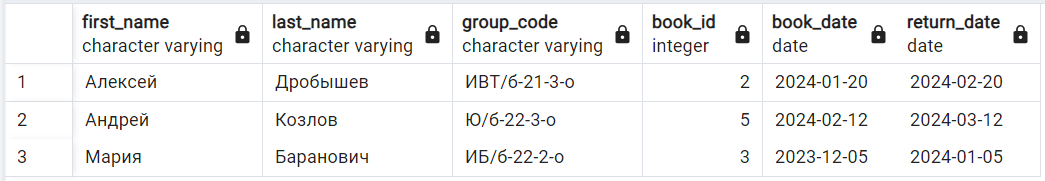


Рисунок 3.19 – Результат запроса 14

Запрос 15: вывести топ авторов по упоминанию их книг в таблице «Выдачи» (текст запроса – Листинг 3.4, результат – Рисунок 3.20).

Листинг 3.15 – Текст запроса 15

select first\_name, last\_name, count(issuance.id) as amount

from authors

join book\_author as ba on ba.author\_id = authors.id

join books on books.id = ba.book\_id

left join issuance on books.id = issuance.book\_id

group by authors.id

order by amount desc;

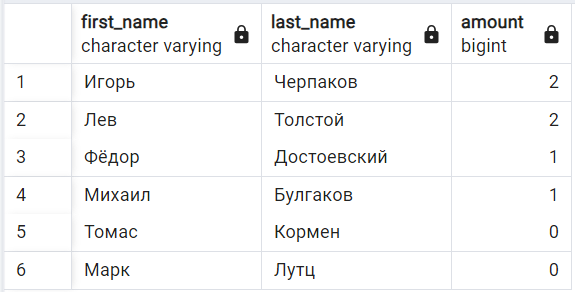


Рисунок 3.20 – Результат запроса 15

По результатам проведённых тестов можно сделать вывод, что таблицы в базе данных созданы правильно и работают корректно.

## 3.3 Разграничение прав доступа

В ходе анализа предметной роли были выделены 2 основные группы пользователей: Сотрудник и Читатель. Сотрудник может просматривать любые данные о книгах и пользователях, добавлять новые данные и обновлять старые. Для него была создана роль STAFF. Читатель может просматривать каталог книг и авторов, делать заявку на выдачу книги, но не имеет доступа к данным других читателей. Для него была создана роль READER.

Листинг 3.16 – Права доступа роли «Сотрудник»

CREATE ROLE staff;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO staff;

GRANT INSERT ON TABLE readers TO staff;

GRANT INSERT ON TABLE books TO staff;

GRANT INSERT ON TABLE authors TO staff;

GRANT INSERT ON TABLE book\_author TO staff;

GRANT UPDATE ON TABLE issuance TO staff;

GRANT UPDATE ON TABLE books TO staff;

GRANT UPDATE ON TABLE authors TO staff;

GRANT DELETE ON TABLE book\_author TO staff;

Листинг 3.17 – Права доступа роли «Читатель»

CREATE ROLE reader;

GRANT SELECT ON TABLE books, authors, book\_author TO reader;

GRANT INSERT ON TABLE issuance TO reader;

# 4. РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

## 4.1 Обоснование выбора языка программирования

Для разработки клиентского приложения была выбрана платформа Windows Forms фреймворка .NET. Она предоставляет удобные инструменты для создания интерфейсов клиентских приложений рабочего стола ОС Windows.

Несмотря на то, что платформа не кроссплатформенная, она обладает рядом важных преимуществ: широкий набор компонентов и библиотек, удобный графический интерфейс, большое количество обучающих материалов.

Windows Forms предусматривает возможность написания ПО на различных языках, однако основным для платформы является C#. В отличие от C++, он более высокоуровневый, простой в использовании и сильнее интегрирован с WinForms. Поэтому для разработки приложения был выбран именно он.

## 4.2 Разработка интерфейса пользователя

.

## 4.3 Алгоритм работы каждого из модулей

.

## 4.4 Тестирование работы приложения

.

# ЗАКЛЮЧЕНИЕ

.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# ПРИЛОЖЕНИЕ А

СОЗДАНИЕ ТАБЛИЦ ДЛЯ БАЗЫ ДАННЫХ

Листинг А.1 – Создание таблицы «Читатели»

CREATE TABLE IF NOT EXISTS public.readers

(

id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

first\_name character varying NOT NULL,

last\_name character varying NOT NULL,

group\_code character varying,

phone character varying COLLATE,

email character varying(256) COLLATE,

fined boolean NOT NULL DEFAULT false,

CONSTRAINT reader\_pkey PRIMARY KEY (id),

CONSTRAINT email CHECK (email ~\* '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'),

CONSTRAINT phone CHECK (phone ~ '^\+[0-9]{11,11}$'::text),

CONSTRAINT group\_code CHECK (group\_code ~\* '[А-Я]{1,3}/б-[0-9]{2}-[0-9]-[а-я]{1,2}$')

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.readers

OWNER to postgres;

Листинг А.2 – Создание таблицы «Книги»

CREATE TABLE IF NOT EXISTS public.books

(

id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

title character varying COLLATE NOT NULL,

publisher character varying COLLATE,

book\_year integer,

amount integer NOT NULL DEFAULT 0,

type character varying COLLATE,

CONSTRAINT books\_pkey PRIMARY KEY (id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.books

OWNER to postgres;

Листинг А.3 – Создание таблицы «Выдача»

CREATE TABLE IF NOT EXISTS public.issuance

(

id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

book\_id integer NOT NULL,

reader\_id integer NOT NULL,

book\_date date,

return\_date date,

status character varying NOT NULL DEFAULT 'pending',

CONSTRAINT issuance\_pkey PRIMARY KEY (id),

CONSTRAINT issuance\_book\_id\_fkey FOREIGN KEY (book\_id)

REFERENCES public.books (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT issuance\_reader\_id\_fkey FOREIGN KEY (reader\_id)

REFERENCES public.readers (id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.issuance

OWNER to postgres;

Листинг А.4 – Создание таблицы «Авторы»

CREATE TABLE IF NOT EXISTS public.authors

(

id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

first\_name character varying COLLATE pg\_catalog."default",

last\_name character varying COLLATE pg\_catalog."default",

CONSTRAINT authors\_pkey PRIMARY KEY (id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.authors

OWNER to postgres;

Листинг А.5 – Создание таблицы «Книга-Автор»

CREATE TABLE IF NOT EXISTS public.book\_author

(

book\_id integer NOT NULL,

author\_id integer NOT NULL,

CONSTRAINT book\_author\_pkey PRIMARY KEY (book\_id, author\_id),

CONSTRAINT book\_author\_author\_id\_fkey FOREIGN KEY (author\_id)

REFERENCES public.authors (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT book\_author\_book\_id\_fkey FOREIGN KEY (book\_id)

REFERENCES public.books (id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.book\_author

OWNER to postgres;

Листинг А.6 – Представление «Количество книг»

CREATE OR REPLACE VIEW public.books\_amount

AS

SELECT books.id,

books.title,

books.amount AS total\_amount,

COALESCE(issued.amount, 0) AS issued\_amount,

books.amount - COALESCE(issued.amount, 0) AS available\_amount

FROM books

LEFT JOIN ( SELECT issuance.book\_id,

count(\*) AS amount

FROM issuance

WHERE issuance.status = 'issued'

GROUP BY issuance.book\_id) issued ON books.id = issued.book\_id;

ALTER TABLE public.books\_amount

OWNER TO postgres;

# ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД КЛИЕНТСКОГО ПРИЛОЖЕНИЯ