

Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ
«МИФИ»

ИНСТИТУТ ЛАЗЕРНЫХ И ПЛАЗМЕННЫХ ТЕХНОЛОГИЙ
КАФЕДРА №31 ПРИКЛАДНАЯ МАТЕМАТИКА

Отчет

по лабораторным работам по предмету:

Базы данных (теоретические основы баз данных)

Выполнили: Есис А.И., Суслова И.И., группа Б20-215

Преподаватель: Павленко Дарья Александровна

Содержание

1	Введение	4
2	Схема базы данных	5
3	Создание базы данных	9
4	Генерация данных	17
5	Примеры запросов	22
6	Заключение	32

1 Введение

LiveLib.ru – русскоязычный интернет-проект, социальная сеть, посвящённая литературе. Сайт предоставляет информацию о книгах, писателях, издательствах, библиотеках, а также даёт возможность ведения читательских дневников, обсуждения книг; публикует рейтинги пользовательских предпочтений, информацию о книжных новинках и литературных новостях, произведениях и изданиях, авторах книг, издательствах и издательских сериях книг, библиотеках. Основной контент сайта создается пользователями. Пользователи могут общаться на форуме, играть в различные литературные игры, участвовать в конкурсах. Также благодаря данному сервису между читателями осуществляется обмен литературой.

В рамках данной лабораторной работы из всего многообразия функционала было выделено следующее:

1. Получение информации о книгах, авторах.
2. Обмен мнениями между пользователями: возможность добавить отзыв к книге или автору, написать историю или цитату к книге, создать обсуждение, отметить книгу понравившейся, прочитанной, недочитанной и остальное. А также некоторые дополнительные функции, такие как: добавление в друзья, получение достижений за различные действия на сайте.

2 Схема базы данных

Рассмотрим основные сущности, выделенные в рамках этой работы:

1. **Издательства.**
2. **Книги.**
3. **Авторы.**
4. **Жанры.**
5. **Цитаты** – цитаты из книги, добавленные пользователями.
6. **Истории** – истории из жизни пользователей, связанные с той или иной книгой.
7. **Отзывы.**
8. **Подборки.**
9. **Пользователи.**
10. **Друзья.**
11. **Достижения.**
12. **Оценки** – оценки к разным объектам на сайте (отзыву, цитате или истории).
13. **Обсуждения.**
14. **Статьи** – сообщения пользователей в каком-либо обсуждении.
15. **Тип отношения пользователь-книга** – пользователь может отметить книгу понравившейся, прочитанной, перечитанной, недочитанной, "читает сейчас" "хочет прочитать".

На Рис. 1 можно увидеть схему связей между этими сущностями – концептуальную модель базы данных.

Далее необходимо выделить для каждой сущности поля (поля created, deleted, updated содержат информацию о дате и времени создания, удаления и обновления записи и подразумеваются в каждой таблице):

1. **source** (картинки, файлы, какие-либо ресурсы) в файловой системе): путь в файловой системе.

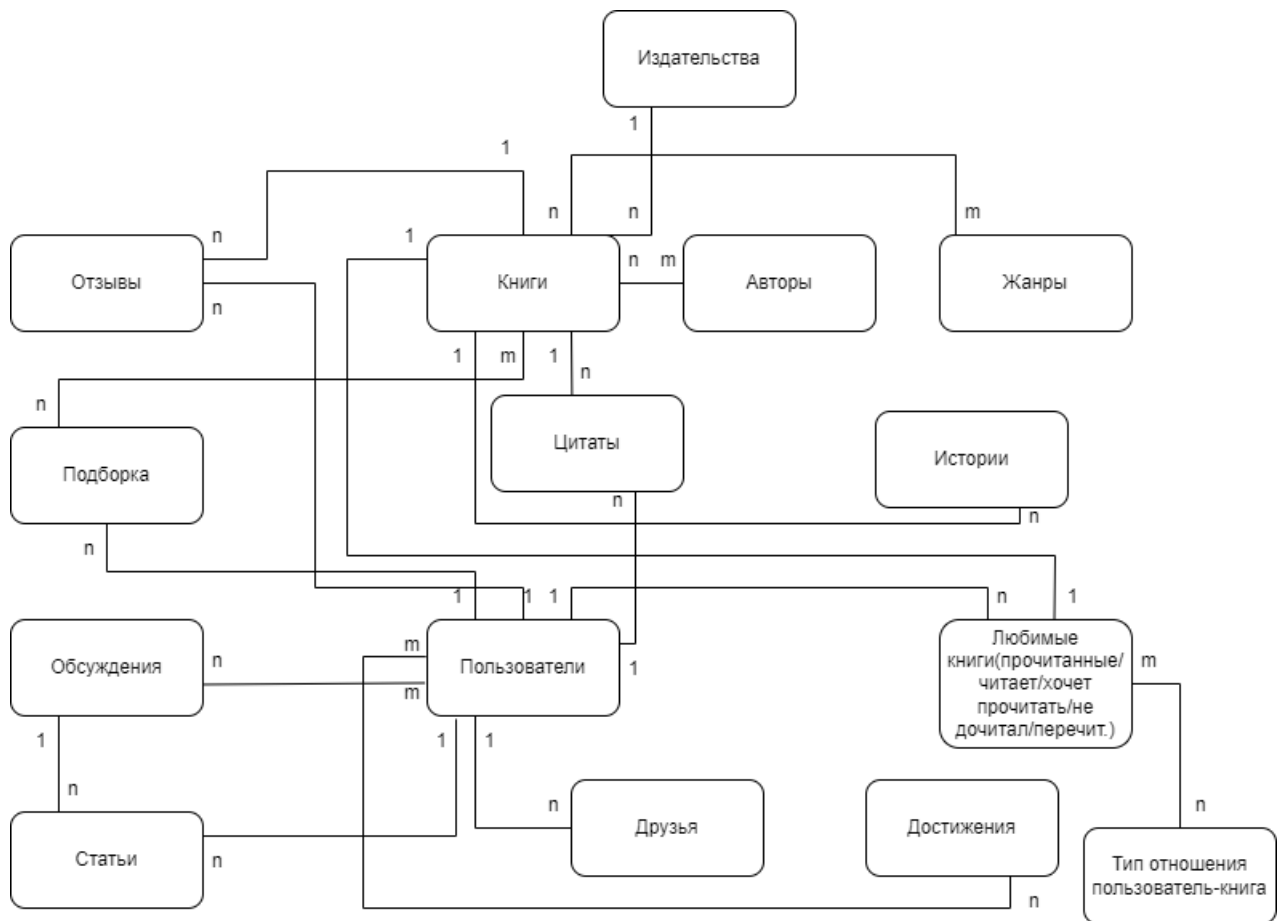


Рис. 1: Концептуальная модель базы данных

2. **Издательства:** publisher_id, название.
3. **Книги:** book_id, название, описание, дата выхода, publisher_id, серия, обложка.
4. **Авторы:** author_id, имя, фамилия, фото, описание, годы жизни.
5. **Книга – Автор:** book_id, author_id.
6. **Жанры:** genre_id, название.
7. **Книга – Жанр:** book_id, genre_id.
8. **Цитаты:** quotes_id, book_id, user_id, текст.
9. **Истории:** story_id, book_id, user_id, текст.
10. **Отзывы:** review_id, book_id, user_id, текст, рейтинг.
11. **Подборки:** user_id, user_id (автор), название, описание.

12. **Книга – Подборка:** book_id, list_id.
13. **Пользователи:** user_id, имя, фамилия, nick, пол, описание, фото, пароль.
14. **Друзья** (пользователь – пользователь): id, user_id_1, user_id_2.
15. **Достижения:** reward_id, название, картинка, описание (за что можно получить).
16. **Достижение – пользователь:** id_user, id_reward.
17. **Оценки:** mark_id, чему оценка (отзыв, цитата, история), оценка.
18. **Обсуждения:** dialog_id, user_id (автор), название, описание.
19. **Статьи:** article_id, user_id (автор), dialog_id (обсуждение), текст.
20. **Тип отношения пользователь-книга:** user_book_type_id, название.
21. **Пользователь – Книга:** id, user_book_type_id, book_id, user_id.

На Рис. 2 представлена логическая модель базы данных.

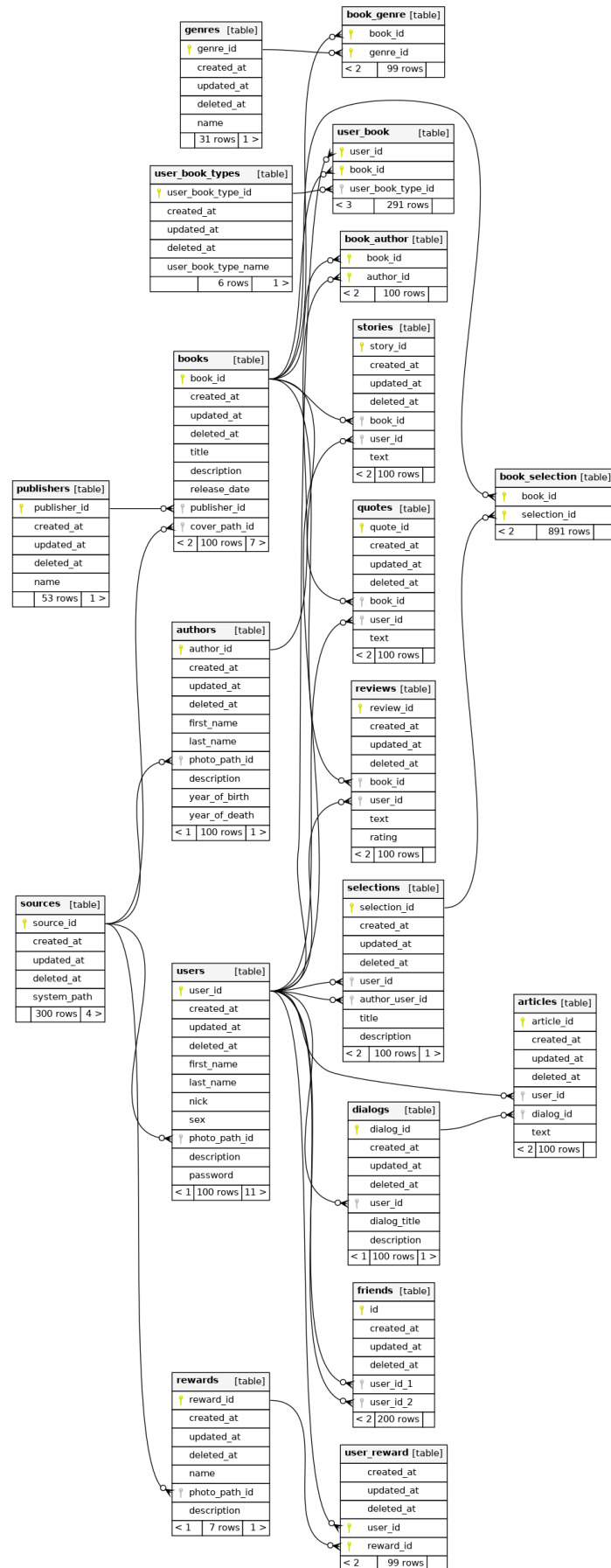


Рис. 2: Логическая модель базы данных

3 Создание базы данных

В соответствии со схемой был написан следующий *create*-запрос:

```
CREATE TABLE IF NOT EXISTS "sources" (  
    source_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    system_path TEXT NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "publishers" (  
    publisher_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ NULL,  
    name VARCHAR(128) UNIQUE NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "books" (  
    book_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    title VARCHAR(256) NOT NULL,  
    description TEXT DEFAULT NULL,  
    release_date TIMESTAMPTZ NOT NULL,  
    publisher_id INT NOT NULL,  
    CONSTRAINT publisher_fk FOREIGN KEY(publisher_id) REFERENCES  
        publishers(publisher_id) ON DELETE SET NULL,  
    cover_path_id INT NOT NULL,  
    CONSTRAINT cover_fk FOREIGN KEY(cover_path_id) REFERENCES  
        sources(source_id) ON DELETE SET NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "authors" (  
    author_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    first_name VARCHAR(128) NOT NULL,  
    last_name VARCHAR(128) NOT NULL,  
    photo_path_id INT DEFAULT NULL,  
    CONSTRAINT photo_path_fk FOREIGN KEY(photo_path_id) REFERENCES  
        sources(source_id) ON DELETE SET NULL,  
    description TEXT DEFAULT NULL,  
    year_of_birth TIMESTAMPTZ NOT NULL,  
    year_of_death TIMESTAMPTZ DEFAULT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "book_author" (  
    book_id INT REFERENCES books(book_id) ON UPDATE CASCADE ON DELETE  
        SET NULL,  
    author_id INT REFERENCES authors(author_id) ON UPDATE CASCADE,  
    CONSTRAINT book_author_pk PRIMARY KEY (book_id, author_id)  
);
```

```
CREATE TABLE IF NOT EXISTS "genres" (  
    genre_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    name VARCHAR(128) UNIQUE NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "book_genre" (  
    book_id INT REFERENCES books(book_id) ON UPDATE CASCADE ON DELETE
```

```

        SET NULL,
        genre_id INT REFERENCES genres(genre_id) ON UPDATE CASCADE,
        CONSTRAINT book_genre_pk PRIMARY KEY (book_id, genre_id)
    );

```

```

CREATE TABLE IF NOT EXISTS "users" (
    user_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,
    deleted_at TIMESTAMPTZ DEFAULT NULL,
    first_name VARCHAR(128) NOT NULL,
    last_name VARCHAR(128) NOT NULL,
    nick VARCHAR(128) NOT NULL,
    sex VARCHAR(6) DEFAULT NULL,
    photo_path_id INT DEFAULT NULL,
    CONSTRAINT photo_path_fk FOREIGN KEY(photo_path_id) REFERENCES
        sources(source_id) ON DELETE SET NULL,
    description TEXT DEFAULT NULL,
    password VARCHAR(128) NOT NULL
);

```

```

CREATE TABLE IF NOT EXISTS "quotes" (
    quote_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,
    deleted_at TIMESTAMPTZ DEFAULT NULL,
    book_id INT,
    CONSTRAINT book_fk FOREIGN KEY(book_id) REFERENCES books(book_id)
        ON DELETE SET NULL,
    user_id INT,
    CONSTRAINT user_fk FOREIGN KEY(user_id) REFERENCES users(user_id)
        ON DELETE SET NULL,
    text TEXT NOT NULL
);

```

);

```
CREATE TABLE IF NOT EXISTS "stories" (  
    story_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    book_id INT NOT NULL,  
    CONSTRAINT book_fk FOREIGN KEY(book_id) REFERENCES books(book_id)  
        ON DELETE CASCADE,  
    user_id INT NOT NULL,  
    CONSTRAINT user_fk FOREIGN KEY(user_id) REFERENCES users(user_id)  
        ON DELETE CASCADE,  
    text TEXT NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "reviews" (  
    review_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    book_id INT NOT NULL,  
    CONSTRAINT book_fk FOREIGN KEY(book_id) REFERENCES books(book_id)  
        ON DELETE CASCADE,  
    user_id INT NOT NULL,  
    CONSTRAINT user_fk FOREIGN KEY(user_id) REFERENCES users(user_id)  
        ON DELETE CASCADE,  
    text TEXT NOT NULL,  
    rating SMALLINT NOT NULL,  
    CONSTRAINT rating_from_0_to_5 CHECK (  
        rating >= 0  
        AND rating <= 5  
)
```

);

```
CREATE TABLE IF NOT EXISTS "friends" (  
    id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    user_id_1 INT REFERENCES users(user_id) ON UPDATE CASCADE  
        ON DELETE SET NULL,  
    user_id_2 INT REFERENCES users(user_id) ON UPDATE CASCADE  
);
```

```
CREATE TABLE IF NOT EXISTS "rewards" (  
    reward_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    name VARCHAR(128) UNIQUE NOT NULL,  
    photo_path_id INT DEFAULT NULL,  
    CONSTRAINT photo_path_fk FOREIGN KEY(photo_path_id)  
        REFERENCES sources(source_id) ON DELETE SET NULL,  
    description TEXT DEFAULT NULL  
);
```

```
CREATE TABLE user_reward (  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    user_id INT REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE  
        SET NULL,  
    reward_id INT REFERENCES rewards(reward_id) ON UPDATE CASCADE,  
    CONSTRAINT user_reward_pk PRIMARY KEY (user_id , reward_id)  
);
```

```
CREATE TABLE IF NOT EXISTS "user_book_types" (  
    user_book_type_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    user_book_type_name VARCHAR(128) UNIQUE NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "marks" (  
    mark_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    type_of_marked_object VARCHAR(128) NOT NULL,  
    value_of_mark INT DEFAULT 0 NOT NULL,  
    CONSTRAINT value_of_mark_from_0_to_5 CHECK (  
        value_of_mark >= 0  
        AND value_of_mark <= 5  
    )  
);
```

```
CREATE TABLE IF NOT EXISTS "dialogs" (  
    dialog_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    user_id INT NOT NULL,  
    CONSTRAINT user_id_fk FOREIGN KEY(user_id) REFERENCES  
        users(user_id) ON DELETE SET NULL,  
    dialog_title VARCHAR(128) NOT NULL,  
    description TEXT DEFAULT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "selections" (  
    selection_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    user_id INT NOT NULL,  
    CONSTRAINT user_fk FOREIGN KEY(user_id) REFERENCES users(user_id)  
        ON DELETE CASCADE,  
    author_user_id INT NOT NULL,  
    CONSTRAINT author_user_fk FOREIGN KEY(author_user_id) REFERENCES  
        users(user_id) ON DELETE CASCADE,  
    title TEXT NOT NULL,  
    description TEXT NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS "book_selection" (  
    book_id INT REFERENCES books(book_id) ON UPDATE CASCADE ON DELETE  
        CASCADE,  
    selection_id INT REFERENCES selections(selection_id) ON UPDATE  
        CASCADE,  
    CONSTRAINT book_selection_pk PRIMARY KEY (book_id, selection_id)  
);
```

```
CREATE TABLE IF NOT EXISTS "articles" (  
    article_id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    created_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    updated_at TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP NOT NULL,  
    deleted_at TIMESTAMPTZ DEFAULT NULL,  
    user_id INT NOT NULL,  
    CONSTRAINT user_id_fk FOREIGN KEY(user_id) REFERENCES users(user_id)  
        ON DELETE SET NULL,  
    dialog_id INT NOT NULL,
```

```

    CONSTRAINT dialog_id_fk FOREIGN KEY(dialog_id) REFERENCES
        dialogs(dialog_id) ON DELETE SET NULL,
    text TEXT DEFAULT NULL
);

```

```

CREATE TABLE "user_book" (
    user_id INT REFERENCES users(user_id) ON UPDATE CASCADE
        ON DELETE SET NULL,
    book_id INT REFERENCES books(book_id) ON UPDATE CASCADE
        ON DELETE SET NULL,
    CONSTRAINT user_book_pk PRIMARY KEY (user_id, book_id),
    user_book_type_id INT,
    CONSTRAINT user_book_type_id_fk FOREIGN KEY(user_book_type_id)
        REFERENCES user_book_types(user_book_type_id) ON DELETE SET NULL
);

```

4 Генерация данных

Для генерации данных была использована библиотека для *Python*: *mimesis*, позволяющая сгенерировать случайные имена, ники, даты, а также слова, предложения, цитаты. Рассмотрим несколько примеров генерации данных.

Генерация названий **файлов**:

```
n = 300
import random

sources = []
root = "root/"
extensions = [".jpg", ".png"]
def to_path(i):
    current = i
    path = ""
    cnt = 0;
    while current != 0:
        path += chr(current % (ord('z') - ord('a') + 1) + ord('a'))
        current //= (ord('z') - ord('a') + 1)
        cnt += 1
        if cnt == 1:
            path += '/'
        elif (cnt - 1) % 3 == 0:
            path += '/'
    while cnt < 7:
        path += 'a'
        cnt += 1
        if (cnt - 1) % 3 == 0:
            path += '/'
    return path[::-1]

for i in range(n):
    sources += [root[:-1] + to_path(i) +
                extensions[random.randint(0, len(extensions) - 1)]]
```


Данный код сгенерирует данные в следующем виде:

```
'root/aaa/aaa/a.jpg'
'root/aaa/aaa/b.jpg'
'root/aaa/aaa/c.png'
'root/aaa/aaa/d.jpg'
'root/aaa/aaa/e.jpg'
...
'root/aaa/aab/a.png'
'root/aaa/aab/b.jpg'
...
```

Данный способ позволяет сохранить до 26^7 объектов.

Генерация **цитат**. Генерация отзывов, историй, статей осуществляется похожим образом.

```
n = 100
min_sentences = 3
max_sentences = 15

import random
from mimesis import locales
from mimesis import Text
```

```
text = Text('ru')

quotes = []
for i in range(n):
    quotes.append(text.quote())
quotes[:min(n, 1)]
```

Цитаты создаются при помощи библиотеки *mimesis*, которая предоставляет случайную цитату при вызове соответствующей функции. Для отзывов и других текстовых полей используется функция для получения случайных предложений (пример далее, в генерации описания для пользователя).

Генерация **пользователей**:

```
min_sentences = 0
```

```
max_sentences = 3
```

```
from mimesis import locales, enums
```

```
from mimesis import Generic
```

```
import random
```

```
import hashlib
```

```
class User:
```

```
    def __init__(self):
```

```
        generic = Generic('ru')
```

```
        self.sex = random.choice([enums.Gender.MALE,
                                   enums.Gender.FEMALE])
```

```
        self.name = generic.person.name(gender=self.sex)
```

```
        self.last_name = generic.person.last_name(gender=self.sex)
```

```
        self.nick = generic.person.username()
```

```
        self.description = generic.text.text(quantity=random
                                                .randint(min_sentences, max_sentences))
```

```
        self.photo = random.choice(sources)
```

```
        self.password = hashlib.md5(generic.person.password()
                                       .encode('utf-8')).hexdigest()
```

```
        self.sex = 'male' if self.sex == enums.Gender.MALE
                       else 'female'
```

Данный код, при создании класса *User* создает один объект, заполненный необходимыми данными. Как видно из кода, пол определяется случайным выбором, далее в соответствии с полом генерируются русские имя и фамилия (так как сайт является русскоязычным), ник, описание (несколько случайных предложений в случайном количестве от 0 до 3). Фото является случайным элементом из списка ресурсов. Пароль храниться в виде хэша (*md-5*) от случайного пароля, сгенерированного библиотекой. Генерация авторов происходит аналогично.

Ряд полей (названия издательств, названия книг, типы отношения пользователь-книга были заполнены осмысленными данными вручную). Например, **достижения**:

```
import random
```

```

class Reward:
    def __init__(this, name, description):
        this.name = name
        this.source = random.choice(sources)
        this.description = description
    def __str__(this):
        return ' '.join([this.name, this.source, this.description])

rewards = [Reward("Книгочей", "1500+_прочитанных_книг"),
            Reward("Критик", "30+_рецензий_с_оценкой_50+"),
            Reward("Цитатчик", "1000+_цитат"),
            Reward("Комментатор", "3000+_комментариев"),
            Reward("Большой_друг", "250+_взаимных_друзей"),
            Reward("Рассказчик", "5_историй_с_оценкой_50+"),
            Reward("Старожил",
                  "1000_дней_на_сайте,_100+_прочитанных_книг"),
            ]

```

Названия и описание были взяты с сайта.

Сгенерированные данные были добавлены в соответствующую таблицу. Пример подключения и *insert*-запроса:

```

import psycopg2

conn = psycopg2.connect(database="lab",
                        host="localhost",
                        user="lab",
                        password="1111",
                        port="5432")

cursor = conn.cursor()
SQL = '''INSERT INTO books (
        title,
        description,
        release_date,

```

```

        publisher_id ,
        cover_path_id
    )
VALUES (
    'Martin Eden ',
    'The title character becomes a writer , hoping to acquire
the respectability sought by his society-girl sweetheart.',
    '1909-09-22 00:00:00-00 ',
    1,
    1
);
cursor.execute(SQL)
conn.commit()

```

5 Примеры запросов

Далее рассмотрим ряд запросов к данной базе для получения необходимой информации.

Запрос 1. Получение всех имен друзей пользователей.

```
SELECT u1.first_name ,  
       u1.last_name ,  
       u2.first_name ,  
       u2.last_name  
FROM friends  
      JOIN users u1 on u1.user_id = friends.user_id_1  
      JOIN users u2 on u2.user_id = friends.user_id_2  
ORDER BY u1.first_name ;
```

Результат (несколько первых записей) – Рис. 3.

	first_name character varying (128) 🔒	last_name character varying (128) 🔒	first_name character varying (128) 🔒	last_name character varying (128) 🔒
1	Абакар	Бурлачков	Кислица	Аплечеева
2	Абакар	Бурлачков	Жанета	Обзацова
3	Абакар	Бурлачков	Абдель	Патаенков
4	Абат	Мозжевитинов	Лорен	Бегченкова
5	Абдель	Патаенков	Зайда	Авдякова
6	Абдель	Патаенков	Ингрида	Жиббарова
7	Абдель	Патаенков	Юхан	Доломов
8	Абдель	Патаенков	Иштван	Ведешов
9	Абдель	Патаенков	Абакар	Бурлачков
10	Абдель	Патаенков	Казма	Минигулов

Рис. 3: Результат выполнения запроса 1.

Запрос 2. Получение пользователей, у которых больше трёх друзей.

```
SELECT u1.first_name ,  
       u1.last_name ,  
       COUNT(u2.nick)  
FROM friends  
      JOIN users u1 on u1.user_id = friends.user_id_1  
      JOIN users u2 on u2.user_id = friends.user_id_2  
GROUP BY u1.first_name ,
```

```

u1.last_name
HAVING COUNT(u2.nick) >= 4
ORDER BY COUNT(u2.nick) DESC;

```

Результат (несколько первых записей) – Рис. 4.

	first_name character varying (128) 🔒	last_name character varying (128) 🔒	count bigint 🔒
1	Абдель	Патаенков	6
2	Фримэн	Ушанаев	5
3	Иба	Минибаева	4
4	Газиза	Дудышева	4
5	Рамзан	Замолоцких	4
6	Лурдес	Исломова	4
7	Дара	Горбылева	4
8	Никора	Лемзинова	4

Рис. 4: Результат выполнения запроса 2.

Запрос 3. Получение информации о книге – название, автор, издательство.

```

SELECT b.title ,
       a.first_name ,
       a.last_name ,
       p.name
FROM book_author
      JOIN authors a on a.author_id = book_author.author_id
      JOIN books b on b.book_id = book_author.book_id
      JOIN publishers p on p.publisher_id = b.publisher_id;

```

Результат (несколько первых записей) – Рис. 5.

Запрос 4. Получение автора, прожившего дольше всех.

```

SELECT a.first_name ,
       a.last_name ,
       a.year_of_birth ,
       a.year_of_death ,
       (a.year_of_death - a.year_of_birth) as years
FROM authors a

```

	title character varying (256)	first_name character varying (128)	last_name character varying (128)	name character varying (128)
1	«Ласточки и амазонк...	Месроп	Джелепов	Издательство «Весь Мир»
2	«Лола Роза»	Жена	Раянова	Издательский Дом «Интеллект»
3	«Мрачный Жнец»	Махир	Золотоводов	Издательство «Новое литературное обозрение»
4	«Чародей»	Джия	Порхунова	Издательская группа «Дело и сервис»
5	«Тэсс из рода д'Эрбе...	Джоли	Краенова	Издательский Дом «Интеллект»
6	«Столпы Земли»	Карт	Коламеv	Издательство «ФИЗМАТЛИТ»
7	«Война и мир»	Собина	Ерашева	Художественная литература. Литературоведение
8	«Неуютная ферма»	Захар	Гродобоев	Издательство «Вече»
9	«Миддлмарч»	Принс	Ферябников	Издательство «Вече»
10	«Маленькие женщин...	Солон	Лукиллианов	Издательство Православного Свято - Тихоновского Гуманит...
11	«Эмма»	Малыга	Фалевич	Издательская группа «АСТ»

Рис. 5: Результат выполнения запроса 3.

WHERE a.year_of_death IS NOT NULL
ORDER BY years **DESC**;

Результат (несколько первых записей) – Рис. 6.

	first_name character varying (128)	last_name character varying (128)	year_of_birth timestamp with time zone	year_of_death timestamp with time zone	years interval
1	Казак	Гицелов	1522-10-26 00:00:00+02:...	1612-02-19 00:00:00+02:...	32623 days
2	Гамзат	Меркелов	1587-09-12 00:00:00+02:...	1676-11-29 00:00:00+02:...	32586 days
3	единственная	Хетагова	1880-12-17 00:00:00+02:...	1969-06-11 00:00:00+03	32316 days 23:30:17
4	Хорт	Струбщиков	1668-01-03 00:00:00+02:...	1754-07-25 00:00:00+02:...	31614 days
5	Юнна	Вертакова	1914-09-13 00:00:00+02:...	2000-12-12 00:00:00+03	31501 days 23:30:17
6	Рафиль	Чебоков	1613-08-21 00:00:00+02:...	1698-08-20 00:00:00+02:...	31045 days
7	Базука	Котоенков	1828-10-14 00:00:00+02:...	1913-01-17 00:00:00+02:...	30775 days
8	Фаиля	Долбенева	1574-09-17 00:00:00+02:...	1656-01-21 00:00:00+02:...	29711 days
9	Кания	Ошмарова	1588-07-06 00:00:00+02:...	1669-08-15 00:00:00+02:...	29625 days
10	Альтаграсия	Камзалова	1908-12-15 00:00:00+02:...	1988-05-15 00:00:00+04	29005 days 22:30:17
11	Сауда	Шемеманева	1587-11-25 00:00:00+02:...	1664-06-02 00:00:00+02:...	27949 days

Рис. 6: Результат выполнения запроса 4.

Запрос 5. Получение самого взрослого автора.

SELECT a.first_name ,
a.last_name ,
a.year_of_birth ,
(now() – a.year_of_birth) **as** years
FROM authors a
WHERE a.year_of_death IS NULL
ORDER BY years **DESC**;

Результат – Рис. 7.

	first_name character varying (128) 🔒	last_name character varying (128) 🔒	year_of_birth timestamp with time zone 🔒	years interval 🔒
1	Алеф	Завелов	1974-09-09 00:00:00+03	17597 days 15:17:...
2	Месроп	Джелепов	1975-06-16 00:00:00+03	17317 days 15:17:...
3	Алб	Тоцев	1977-02-26 00:00:00+03	16696 days 15:17:...
4	Саломон	Овсенников	1992-11-16 00:00:00+03	10954 days 15:17:...
5	Ильвера	Розова	1993-02-03 00:00:00+03	10875 days 15:17:...
6	Гарина	Исмайлова	1994-11-15 00:00:00+03	10225 days 15:17:...
7	Мигель	Чучкаев	1997-11-22 00:00:00+03	9122 days 15:17:3...
8	Гела	Цулева	1998-02-04 00:00:00+03	9048 days 15:17:3...
9	Несси	Бутраева	2003-12-31 00:00:00+03	6892 days 15:17:3...
10	Бая	Ходжибекова	2005-04-30 00:00:00+04	6406 days 16:17:3...

Рис. 7: Результат выполнения запроса 5.

Запрос 6. Получение информации о количестве сколько книг прочитал.

```

SELECT u.first_name ,
       u.last_name ,
       u.nick ,
       COUNT(user_book.book_id)
FROM user_book
      JOIN user_book_types ubt on
         ubt.user_book_type_id = user_book.user_book_type_id
      JOIN users u on user_book.user_id = u.user_id
WHERE ubt.user_book_type_name = 'Прочитала()'
GROUP BY u.user_id
ORDER BY COUNT(user_book.book_id) DESC;

```

Результат (несколько первых записей) – Рис. 8.

Далее рассмотрим последовательно несколько запросов, чтобы получить информацию о книгах, прочитанных только женщинами.

Запрос 7. Получение информации о поле людей, прочитавших какие-либо книги.

```

SELECT ub.user_id ,
       ub.book_id ,
       b.title ,
       u.sex

```


	first_name character varying (128)	last_name character varying (128)	nick character varying (128)	count bigint
1	Лиёпа	Мусюнов	narcotization.1895	3
2	Чики	Найдов-Железова	Improve.1809	2
3	Улун	Шохорова	phasianinae.2055	2
4	Джим	Куламазов	EnfeeblishBarong.1815	2
5	Газиза	Дудышева	Nonimposition_1965	2
6	Гия	Рева	Filamentiferous-1870	2
7	Нагар	Жерлов	SchuitDecafid-1863	2
8	Доступа	Дукаева	Waugh-1856	2
9	Катерина	Продаева	cauchy.2057	2
10	Ануля	Улесова	Earwig.1902	2

Рис. 8: Результат выполнения запроса 6.

```

FROM user_book AS ub
JOIN books b ON b.book_id = ub.book_id
JOIN users u ON ub.user_id = u.user_id
JOIN user_book_types ubt ON
    ub.user_book_type_id = ubt.user_book_type_id
JOIN book_genre bg ON bg.book_id = b.book_id
JOIN genres g ON g.genre_id = bg.genre_id
WHERE g.name = 'Бизнескниги-'
AND ubt.user_book_type_name = 'Прочитала()';

```

Результат – Рис. 9.

	user_id integer	book_id integer	title character varying (256)	sex character varying (6)
1	34	14	«Унесённые ветром»	female
2	59	14	«Унесённые ветром»	male
3	66	14	«Унесённые ветром»	male
4	71	19	«Сёгун»	female
5	75	19	«Сёгун»	female
6	16	40	«Белый пик»	male
7	71	40	«Белый пик»	female
8	82	48	«Большие надежды»	female

Рис. 9: Результат выполнения запроса 7.

Запрос 8. Получение книг, которые прочитал хотя бы один мужчина.

```

WITH r1 AS (

```

```

SELECT ub.user_id ,
       ub.book_id ,
       b.title ,
       u.sex
FROM user_book AS ub
      JOIN books b ON b.book_id = ub.book_id
      JOIN users u ON ub.user_id = u.user_id
      JOIN user_book_types ubt ON
          ub.user_book_type_id = ubt.user_book_type_id
      JOIN book_genre bg ON bg.book_id = b.book_id
      JOIN genres g ON g.genre_id = bg.genre_id
WHERE g.name = 'Бизнескниги-'
      AND ubt.user_book_type_name = 'Прочитала()'
)
SELECT DISTINCT r1.title
FROM r1
      JOIN r1 r2 ON r1.book_id = r2.book_id
WHERE r1.sex = 'male'
      OR r2.sex = 'male';

```

Результат – Рис. 10.


	title character varying (256) 
1	«Унесённые ветром»
2	«Белый пик»

Рис. 10: Результат выполнения запроса 8.

Запрос 9. Получение названий книг, прочитанных только женщинами.

```

WITH r1 AS (
    SELECT ub.user_id ,
           ub.book_id ,
           b.title ,
           u.sex
    FROM user_book AS ub
          JOIN books b ON b.book_id = ub.book_id

```

```

JOIN users u on ub.user_id = u.user_id
JOIN user_book_types ubt on
    ub.user_book_type_id = ubt.user_book_type_id
JOIN book_genre bg on bg.book_id = b.book_id
JOIN genres g on g.genre_id = bg.genre_id
WHERE g.name = 'Бизнескниги-'
    AND ubt.user_book_type_name = 'Прочитала()'
)
SELECT DISTINCT b.title
FROM user_book AS ub
    JOIN books b on b.book_id = ub.book_id
    JOIN users u on ub.user_id = u.user_id
    JOIN user_book_types ubt on
        ub.user_book_type_id = ubt.user_book_type_id
    JOIN book_genre bg on bg.book_id = b.book_id
    JOIN genres g on g.genre_id = bg.genre_id
WHERE g.name = 'Бизнескниги-'
    AND ubt.user_book_type_name = 'Прочитала()'
EXCEPT
SELECT DISTINCT r1.title
FROM r1
    JOIN r1 r2 on r1.book_id = r2.book_id
WHERE r1.sex = 'male'
    OR r2.sex = 'male';

```

Результат – Рис. 11.


	title character varying (256) 
1	«Сёгун»
2	«Большие надежды»

Рис. 11: Результат выполнения запроса 9.

Запрос 10. Выделение категорий (по количеству прочитанных книг.

```

SELECT u.first_name ,
       u.last_name ,

```

```

u.nick ,
COUNT(user_book.book_id) as books ,
CASE WHEN COUNT(user_book.book_id) BETWEEN 0 AND 1 THEN
    'мало_(<=1) '
    WHEN COUNT(user_book.book_id) BETWEEN 1 AND 2 THEN
    'средне_(<=2) '
    WHEN COUNT(user_book.book_id) >= 3 THEN
    'много_(>=3) '
END CASE
FROM user_book
JOIN user_book_types ubt on
    ubt.user_book_type_id = user_book.user_book_type_id
JOIN users u on user_book.user_id = u.user_id
WHERE ubt.user_book_type_name = 'Прочитала()'
GROUP BY u.user_id
ORDER BY books DESC;

```

Результат (несколько первых записей) – Рис. 12.

	first_name character varying (128) 🔒	last_name character varying (128) 🔒	nick character varying (128) 🔒	books bigint 🔒	case text 🔒
1	Либа	Мусюнов	narcotization.1895	3	много (>=3)
2	Ануля	Улесова	Earwig.1902	2	средне (<=2)
3	Чики	Найдов-Железова	Improve.1809	2	средне (<=2)
4	Шина	Бидердинова	Davidh_1922	2	средне (<=2)
5	Дау	Казиканова	Quizzify.2058	1	мало (<=1)
6	Фаридун	Сонцев	suez.1869	1	мало (<=1)
7	Зиля	Катараева	Northeners.1800	1	мало (<=1)
8	Аленик	Валерков	crumpler.1881	1	мало (<=1)
9	Айтан	Выгодчиков	Portment_1904	1	мало (<=1)

Рис. 12: Результат выполнения запроса 10.

Запрос 11. Рекурсивный поиск друзей.

```

WITH RECURSIVE r AS (SELECT user_id , nick , 1 AS level
FROM users
WHERE user_id = 1

```

UNION ALL

```
SELECT u.user_id, u.nick, r.level + 1 AS level
FROM friends f
      JOIN r ON f.user_id_1 = r.user_id
      JOIN users u on u.user_id = f.user_id_2
WHERE r.level < 5)
SELECT DISTINCT nick, user_id, MIN(level) as level FROM r
GROUP BY nick, user_id;
```

Результат (несколько первых записей) – Рис. 12.

	nick character varying (128)	user_id integer	level integer
1	missouri_1928	49	3
2	hughes_2058	58	4
3	elimination_2052	1	1
4	playlist_1911	82	4
5	venice_1898	81	2
6	trip_2022	27	5
7	holiday_1953	31	5
8	granted_1902	11	4
9	authority_2069	34	3
10	forwarding_2050	4	5

Рис. 13: Результат выполнения запроса 11.

Запрос 12. Теория 5 рукопожатий.

```
WITH RECURSIVE r AS (SELECT user_id, nick, 1 AS level
FROM users
WHERE user_id = 1
```

UNION ALL

```
SELECT u.user_id, u.nick, r.level + 1 AS level
FROM friends f
      JOIN r ON f.user_id_1 = r.user_id
      JOIN users u on u.user_id = f.user_id_2
WHERE r.level < 5)
SELECT COUNT(h) as handshake FROM (SELECT nick
FROM users
```

```
EXCEPT
SELECT DISTINCT r.nick
FROM r) as h;
```

Результат (несколько первых записей) – Рис. 12.

	handshake bigint 
1	73

Рис. 14: Результат выполнения запроса 12.

6 Заключение

Таким образом была спроектирована, создана и заполнена база данных, соответствующая некоторому функционалу книжного сайта *LiveLib.ru*, а также реализован ряд аналитических запросов к ней.