



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ
ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра системного програмування та спеціалізованих комп'ютерних систем

Лабораторна робота №2
з дисципліни **Бази даних і засоби управління**
на тему: «Створення додатку бази даних, орієнтованого на
взаємодію з СУБД PostgreSQL»

Виконав:
студент III курсу
групи КВ-92
Михайлов О.Г.
Перевірів:
Петрашенко А.В.

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

URL репозиторію з вихідним кодом:

https://github.com/AlexMykhallov/DataBase_KV-92.git

Результати виконання операції вилучення запису батьківської таблиці:
З таблиці Author:

```
-----  
| 1 to insert |  
| 2 to insert random |  
| 3 to update |  
| 4 to delete |  
| 5 to select table |  
| 6 to search |  
| 7 to end |  
-----  
  
Input: 4  
  
-----  
| Select a table: |  
| 1 = Author |  
| 2 = Book |  
| 3 = Subscription |  
-----  
  
Input: 1  
Num of id: 111  
  
[Data was successfully deleted]
```

Before:

	author_id [PK] integer	name character varying	origin character varying
1	111	Taras Shevchenko	Ukraine
2	4212	Eric Arthur Blair	India
3	46546	Joanne Rowling	England
4	213689	Agatha Mary Clarissa Miller	England

After:

	author_id [PK] integer	name character varying	origin character varying
1	4212	Eric Arthur Blair	India
2	46546	Joanne Rowling	England
3	213689	Agatha Mary Clarissa Miller	England

Результат, коли вилучення неможливе:

В Subscription:

```
| 1 to insert |
| 2 to insert random |
| 3 to update |
| 4 to delete |
| 5 to select table |
| 6 to search |
| 7 to end |
-----
Input: 4
-----
| Select a table: |
| 1 = Author |
| 2 = Book |
| 3 = Subscription |
-----
Input: 3
New of id: 2

[Error while working with PostgreSQL ORM/ORM: update and delete в таблиці "Subscription" порушує обмеження зовнішнього ключа "Book_sub_id_fkey" таблиці "Book"
DETAIL: На елемент (sub_id)=(2) вже є посилання в таблиці "Book".
]
```

Результати виконання операції вставки запису в дочірню таблицю:

В Book:

```
| 1 to insert |
| 2 to insert random |
| 3 to update |
| 4 to delete |
| 5 to select table |
| 6 to search |
| 7 to end |
-----
Input: 1
-----
| Select a table: |
| 1 = Author |
| 2 = Book |
| 3 = Subscription |
-----
Input: 2
Input new book id: 123
Input new sub id: 3
Input new author id: 4212
Input new title: Animal Farm
Input new genre: allegorical novella

[Data was successfully inserted]
```

Before:

	book_id [PK] integer	sub_id integer	author_id integer	title character varying	genre character varying
1	1	1	46546	Harry Potter and Goblet of Fire	fantasy
2	2	2	213689	Murder on the Orient Express	detective
3	4	2	4212	1984	dystopian
4	5	1	46546	NRGHZ	WPPVN

After:

	book_id [PK] integer	sub_id integer	author_id integer	title character varying	genre character varying
1	1	1	46546	Harry Potter and Goblet of Fire	fantasy
2	2	2	213689	Murder on the Orient Express	detective
3	4	2	4212	1984	dystopian
4	5	1	46546	NRGHZ	WPPVN
5	123	3	4212	Animal Farm	allegorical novella

Неможливість запису(в батьківській таблиці немає відповідного запису):

В Book:

```
| 1 to insert  
| 2 to insert random  
| 3 to update  
| 4 to delete  
| 5 to select table  
| 6 to search  
| 7 to end  
|-----|
```

Input: 1

```
| Select a table:  
| 1 = Author  
| 2 = Book  
| 3 = Subscription  
|-----|
```

Input: 2

Input new book id: 123

Input new sub id: 1

Input new author id: 7777

Input new title: wppv

Input new genre: wppv





```
[Error while working with PostgreSQL ПОМИЛКА: insert або update в таблиці "Book" порушує обмеження зовнішнього ключа "Book_author_id_fkey"  
DETAIL: Ключ (author_id)=(7777) не присутній в таблиці "Author".  
]
```

Ілюстрації фрагментів згенерованих таблиць з відповідними SQL-запитами:

В Author:

```
-----  
| 1 to insert |  
| 2 to insert random |  
| 3 to update |  
| 4 to delete |  
| 5 to select table |  
| 6 to search |  
| 7 to end |  
-----  
  
Input: 2  
  
-----  
| Select a table: |  
| 1 = Author |  
| 2 = Book |  
| 3 = Subscription |  
-----  
  
Input: 1  
Input amount of data to generate: 12  
  
[Random Data was successfully inserted]
```

Before:

	 author_id [PK] integer 	name character varying 	origin character varying 
1	4212	Eric Arthur Blair	India
2	46546	Joanne Rowling	England
3	213689	Agatha Mary Clarissa Miller	England

After:

	author_id [PK] integer	name character varying	origin character varying
1	4	RQXFV	CYLLU
2	5	DTZIF	CHKCZ
3	6	LNEE	TPIEN
4	7	FWPNU	KJGSK
5	8	NUHBH	BZXDE
6	9	EXYWM	JLLEZ
7	10	NHHRM	EKXLQ
8	11	LDZCK	FLWZV
9	12	YDEOU	TVKFT
10	13	HLIKI	YWGOD
11	14	DVWRI	IYISV
12	15	QNL PY	USEFS
13	4212	Eric Arthur Blair	India
14	46546	Joanne Rowling	England
15	213689	Anatha Mary Clarissa Miller	England

```
cursor.execute("""INSERT INTO "Author"(author_id, name, origin)
VALUES(
(SELECT(select count(author_id) from "Author")+1::int),
(SELECT (chr(ascii('B') + (random() * 25)::int) ||
chr(ascii('B') + (random() * 25)::int) ||
chr(ascii('B') + (random() * 25)::int) ||
chr(ascii('B') + (random() * 25)::int)
)),
(SELECT (chr(ascii('B') + (random() * 25)::int) ||
chr(ascii('B') + (random() * 25)::int) ||
chr(ascii('B') + (random() * 25)::int) ||
chr(ascii('B') + (random() * 25)::int)
))));"
```

Ілюстрації результатів пошукових запитів з відповідними SQL-запитами:

В Author:

Пошук за author_id:

```
-----
| 1 to insert      |
| 2 to insert random |
| 3 to update      |
| 4 to delete      |
| 5 to select table |
| 6 to search      |
| 7 to end         |
-----

Input: 6

-----
| Select a table:   |
| 1 = Author       |
| 2 = Book          |
| 3 = Subscription |
-----

Input: 1
1 = to search author_id
2 = to search name or origin
Input: 1
Input start: 1
Input finish: 15
```

```
author_id = 13
name = HLIKI
origin = YWGOD

author_id = 14
name = DVWRI
origin = IYISV

author_id = 15
name = QNLPY
origin = USEFS

Search duration = 0.0018529891967773438
```

```
start = time.time()
cursor.execute("""SELECT * FROM "Author" WHERE %s <= author_id and author_id <= %s;""", (a, b,))
finish = time.time()
```

Пошук за name та origin:

Input: 0

```
-----  
| Select a table: |  
| 1 = Author     |  
| 2 = Book       |  
| 3 = Subscription |  
-----
```

Input: 1

1 = to search author_id

2 = to search name or origin

Input: 2

Input text: *Joanne Rowling England*

author_id = 46546

name = Joanne Rowling

origin = England

Search duration = 0.0009996891021728516

```
cursor.execute("""SELECT * FROM "Author" WHERE  
    to_tsvector(name) || to_tsvector(origin)  
    @@ plainto_tsquery(%s);""", (txt,))  
finish = time.time()
```

B Book:

Пошук за book_id:

Input: 2

1 = to search book_id

2 = to search sub_id

3 = to search author_id

4 = to search title or genre

Input: 1

Input start: 1

Input finish: 10

book_id = 1

sub_id = 1

```
book_id = 4
sub_id = 2
author_id = 4212
title = 1984
genre = dystopian
```

```
book_id = 5
sub_id = 1
author_id = 46546
title = NRGHZ
genre = WPPVN
```

```
Search duration = 0.0014832019805908203
```

```
start = time.time()
cursor.execute("""SELECT * FROM "Book" WHERE %s <= book_id and book_id <= %s;""", (a, b,))
finish = time.time()
```

Помык за title та genre:

```
4 = to search title or genre
```

```
Input:4
```

```
Input text: 1984 dystopian
```

```
book_id = 4
sub_id = 2
author_id = 4212
title = 1984
genre = dystopian
```

```
Search duration = 0.0017328262329101562
```

```
cursor.execute("""SELECT * FROM "Book" WHERE
    to_tsvector(title) || to_tsvector(genre)
    @@ plainto_tsquery(%s);""", (txt,))
```

Ілюстрації програмного коду модуля “Model”:

Функція insert:

```
def insert(table, inp):
    connection = connect()
    if table == 1:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Author"(author_id, name, origin) VALUES(%s, %s, %s);""",
                            (inp[0], inp[1], inp[2],))

    elif table == 2:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Book"(book_id, sub_id, author_id, title, genre)
                            VALUES(%s, %s, %s, %s, %s);""",
                            (inp[0], inp[1], inp[2], inp[3], inp[4],))

    elif table == 3:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Subscription"(sub_id, name, email)
                            VALUES(%s, %s, %s);""", (inp[0], inp[1], inp[2],))
```

Забезпечує можливість введення даних до таблиць баз даних.

Функція insert_rand:

```
def insert_rand(table, count):
    connection = connect()
    if table == 1:
        with connection.cursor() as cursor:
            for i in range(0, count):
                cursor.execute("""INSERT INTO "Author"(author_id, name, origin)
                                VALUES(
                                    (SELECT(select count(author_id) from "Author")+1::int),
                                    (SELECT (chr(ascii('A') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int)
                                    )),
                                    (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int)
                                    )));""")

    elif table == 2:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Book"(book_id, sub_id, author_id, title, genre)
                            VALUES(
                                (SELECT(select count(book_id) from "Book")+1::int),
                                (SELECT sub_id FROM "Subscription" OFFSET
                                floor(random()*(select count(sub_id) from "Subscription")) LIMIT 1),
                                (SELECT author_id FROM "Author" OFFSET
                                floor(random()*(select count(author_id) from "Author")) LIMIT 1),
                                (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int)
                                )),
                                (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int)
                                )));""")
```



```

def update(table, inp):
    connection = connect()

    n_id = inp[0]
    column = inp[1]
    new = inp[2]

    if table == 1:
        if column == 1:
            with connection.cursor() as cursor:
                cursor.execute("""update "Author" set name = %s where author_id = %s;""", (new, n_id,))

        elif column == 2:
            with connection.cursor() as cursor:
                cursor.execute("""update "Author" set origin = %s where author_id = %s;""", (new, n_id,))

    elif table == 2:
        if column == 1:
            with connection.cursor() as cursor:
                cursor.execute("""update "Book" set sub_id = %s where book_id = %s;""", (new, n_id,))

        elif column == 2:
            with connection.cursor() as cursor:
                cursor.execute("""update "Book" set author_id = %s where book_id = %s;""", (new, n_id,))

        elif column == 3:
            with connection.cursor() as cursor:
                cursor.execute("""update "Book" set title = %s where book_id = %s;""", (new, n_id,))

        elif column == 4:
            with connection.cursor() as cursor:
                cursor.execute("""update "Book" set genre = %s where book_id = %s;""", (new, n_id,))

    elif table == 3:
        if column == 1:
            with connection.cursor() as cursor:
                cursor.execute("""update "Subscription" set name = %s where sub_id = %s;""", (new, n_id,))

        elif column == 2:
            with connection.cursor() as cursor:
                cursor.execute("""update "Subscription" set email = %s where sub_id = %s;""", (new, n_id,))

```

Забезпечує можливість оновлення потрібних даних в таблиці баз даних.

Функція delete:

```

def delete(table, nid):
    connection = connect()

    if table == 1:
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Author" WHERE author_id = %s;""", (nid,))

    elif table == 2:
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Book" WHERE book_id = %s;""", (nid,))

    elif table == 3:
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Subscription" WHERE sub_id = %s;""", (nid,))

```

Забезпечує можливість видалення потрібних таблиць в таблиці баз даних.

Функція select_table:

```
def select_table(table, inp):
    connection = connect()

    string = inp[0]

    if table == 1:
        if string == 1:
            n_id = inp[1]
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Author" where author_id = %s;""", (n_id,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Author";""")
                return cursor.fetchall()

    elif table == 2:
        if string == 1:
            n_id = inp[1]
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Book" where book_id = %s;""", (n_id,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Book";""")
                return cursor.fetchall()

    elif table == 3:
        if string == 1:
            n_id = inp[1]
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Subscription" where sub_id = %s;""", (n_id,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Subscription";""")
                return cursor.fetchall()
```

Забезпечує можливість виведення потрібних таблиць баз даних, де ми можемо вивести, як всі таблиці, так і одну

Функція search:

```
def search(table, inp):
    connection = connect()

    column = inp[0]

    if table == 1:
        if column == 1:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Author" WHERE %s <= author_id and author_id <= %s;""", (a, b))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
            return ret

        elif column == 2:
            txt = inp[1]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Author" WHERE
                                to_tsvector(name) || to_tsvector(origin)
                                @@ plainto_tsquery(%s);""", (txt,))
                finish = time.time()
                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
            return ret

    elif table == 2:
        if column == 1:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()

                cursor.execute("""SELECT * FROM "Book" WHERE %s <= book_id and book_id <= %s;""", (a, b))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
            return ret

        elif column == 2:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Book" WHERE %s <= sup_id and sup_id <= %s;""", (a, b))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
            return ret

        elif column == 3:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Book" WHERE %s <= author_id and author_id <= %s;""", (a, b))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
            return ret
```

```

elif column == 4:
    txt = inp[1]
    with connection.cursor() as cursor:
        start = time.time()
        cursor.execute("""SELECT * FROM "Book" WHERE
            to_tsvector(title) || to_tsvector(genre)
            @@ plainto_tsquery(%s);""", (txt,))
        finish = time.time()

        search_duration = finish - start
        f_a = cursor.fetchall()
        ret = [f_a, search_duration]
        return ret

elif table == 3:
    if column == 1:
        a = inp[1]
        b = inp[2]
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Subscription" WHERE %s <= sub_id and sub_id <= %s;""", (a, b,))
            finish = time.time()

            search_duration = finish - start
            f_a = cursor.fetchall()
            ret = [f_a, search_duration]
            return ret

    elif column == 2:
        txt = inp[1]
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Subscription" WHERE to_tsvector(name) || to_tsvector(email)
                @@ plainto_tsquery(%s);""", (txt,))
            finish = time.time()

            search_duration = finish - start
            f_a = cursor.fetchall()
            ret = [f_a, search_duration]
            return ret

```

Забезпечує можливість пошуку потрібних даних по всі базі даних.

Модель «сутність-зв’язок»:

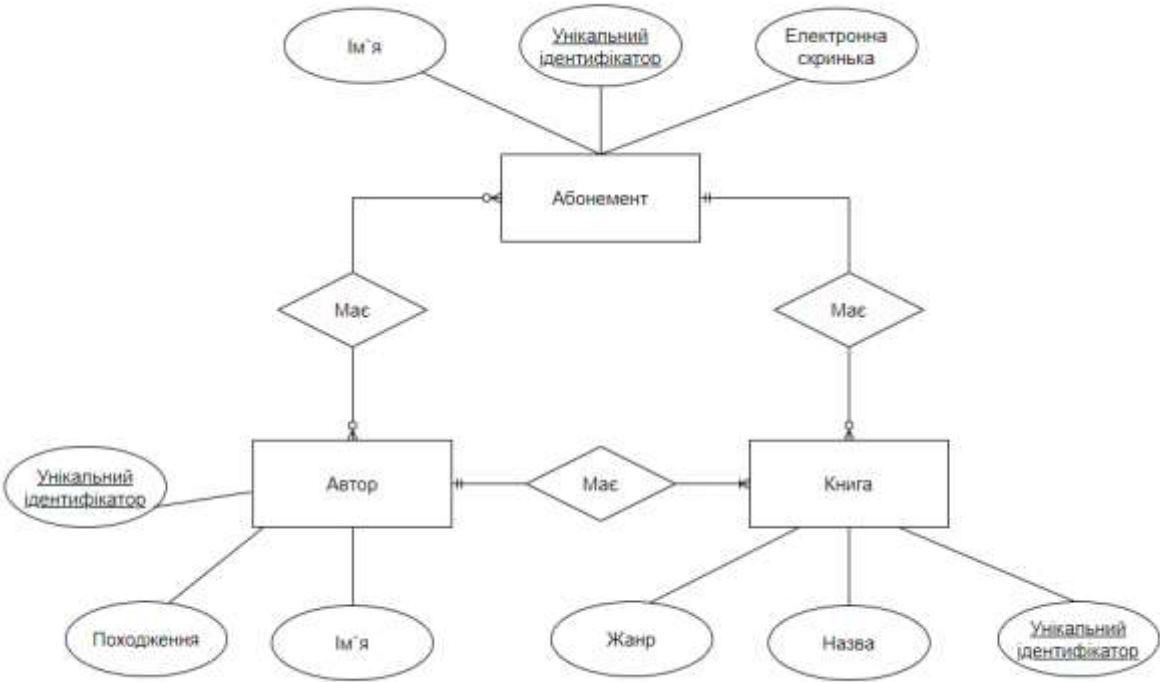


Рис1. ER-діаграма за нотацією Чена
Схеми бази даних:

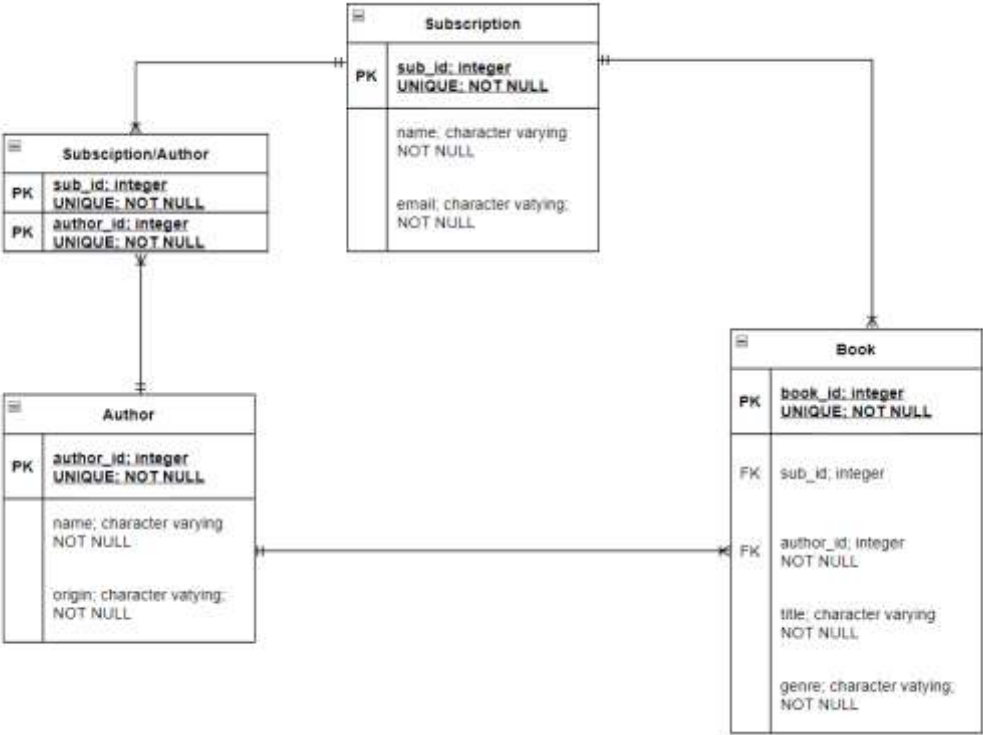


Рис2. Схема бази даних

Сутність	Атрибут	Тип атрибуту
Subscription – містить інформацію про абонемент(людини)	sub_id – первинний ключ, унікальний ідентифікатор абонемента name – ім`я людини email – прив`язана електронна пошта	integer character varying character varying
Book – містить інформацію про книгу	book_id – первинний ключ, що містить значення унікального ідентифікатора книги sub_id – ідентифікатор абонемента, який бере книгу author_id – ідентифікатор автора, котрий написав цю книгу title – назва книги genre - жанр	integer integer character varying character varying
Author – містить інформацію про автора	author_id – первинний ключ, що містить значення унікального ідентифікатора автора name – ім`я людини origin - походження	integer character varying character varying
Subscription/Author – містить інформацію про зв'язок між абонементом і автором	sub-id – первинний ключ, унікальний ідентифікатор абонемента author-id - первинний ключ, унікальний ідентифікатор автора	Integer integer

Опис зв'язків:

Багато абонементаів можуть вподобати нуль або одних й тих самих авторів тому тут утворюється зв'язок М:N. Так як всі книги містять унікальний ідентифікатор, хоча й можуть бути однаковими але все рівно тільки один абонемент може володіти цією/цими книгами, тому сутність «Абонемент» утворює із сутністю «Книги» зв'язок 1:N. І так як автор може написати безліч книжок, або лише одну(мінімум для того, щоб він вважався автором), але вони були написані саме цією однією людиною, звідси маємо зв'язок 1:N.

Опис меню:

Головне меню:

```
-----  
| 1 to insert |  
| 2 to insert random |  
| 3 to update |  
| 4 to delete |  
| 5 to select table |  
| 6 to search |  
| 7 to end |  
-----  
Input:
```

- 1 – введення даних в вибрану таблицю
- 2 – ввід випадкових даних у таблицю
- 3 – оновлення даних в вибраній таблиці
- 4 – видалення даних в вибраній таблиці
- 5 – вивід вибраної таблиці видалення
- 6 – пошук по таблицями бази даних
- 7 – вихід з меню

Після вибору одного з пунктів 1 – 6, ми побачимо наступне меню для вибору таблиці бази даних:

```
-----  
| Select a table: |  
| 1 = Author |  
| 2 = Book |  
| 3 = Subscription |  
-----  
Input: |
```

Якщо, в першому меню ми вибрали пункт 1 та в другому вибрали таблицю, то побачимо:

```
Input new book id: 100  
Input new sub id: 1  
Input new author id: 46546  
Input new title: The Ickabog  
Input new genre: Fairy tale
```

меню для вводу нових даних.

Якщо, в першому меню ми вибрали пункт 2 та в другому вибрали таблицю, то побачимо:

- Для таблиці 1

```
Input: 1
```

```
Input amount of data to generate: |
```

меню для вводу кількості даних

для генерації

- Для інших таблиць буде проведена одна генерація

Якщо, в першому меню ми вибрали пункт 3 та в другому вибрали таблицю, то побачимо:

```
Num of author id: 12
```

```
1 = to upd name
```

```
2 = to upd origin
```

```
Input:|
```

меню для вибору даних для оновлення.

Якщо, в першому меню ми вибрали пункт 4 та в другому вибрали таблицю, то побачимо:

```
Num of id: 1|
```

меню де вводимо id для видалення рядка.

Якщо, в першому меню ми вибрали пункт 5 та в другому вибрали таблицю, то побачимо:

```
1 = to one str
```

```
2 = to all str
```

```
Input:|
```

меню для вибору вивести один рядок чи всі,

```
Input:1
```

```
Num of id:
```

якщо, вибираємо 1 то вводимо id рядка для виводу.

Якщо, в першому меню ми вибрали пункт 6 та в другому вибрали таблицю, то побачимо меню для вибору атрибуту, та пошуку конкретних даних, для таблиці “Author”

```
1 = to search author_id
```

```
2 = to search name or origin
```

```
Input:
```

Для таблиці “Book”

```
1 = to search book_id
2 = to search sub_id
3 = to search author_id
4 = to search title or genre
Input:
```

Для таблиці “Subscription”

```
1 = to search sub_id
2 = to search name or email
Input:
```

Код програми: main.py

```
from Controller import menu

menu()
```

Model.py

```
import time
import psycopg2

def connect():
    connection = psycopg2.connect(
        database="Library",
        user="postgres",
        password="ndeene28082002",
        host="127.0.0.1")
    connection.autocommit = True
    return connection

def close():
    connection = connect()
    connection.close()

def insert(table, inp):
    connection = connect()
    if table == 1:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Author"(author_id, name, origin)
VALUES(%s, %s, %s);""",
                            (inp[0], inp[1], inp[2],))

    elif table == 2:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Book"(book_id, sub_id, author_id,
title, genre)
VALUES(%s, %s, %s, %s, %s);""",
                            (inp[0], inp[1], inp[2], inp[3], inp[4],))

    elif table == 3:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Subscription"(sub_id, name, email)
```

```

VALUES(%s, %s, %s);""" , (inp[0], inp[1], inp[2],))

def insert_rand(table, count):
    connection = connect()
    if table == 1:
        with connection.cursor() as cursor:

            for i in range(0, count):
                cursor.execute("""INSERT INTO "Author"(author_id, name,
origin)
                                VALUES (
                                    (SELECT(select count(author_id) from "Author")+1::int),
                                    (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int)
                                    )),
                                    (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int) ||
                                        chr(ascii('B') + (random() * 25)::int)
                                    )))""")

    elif table == 2:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Book"(book_id, sub_id, author_id,
title, genre)
                            VALUES (
                                (SELECT(select count(book_id) from "Book")+1::int),
                                (SELECT sub_id FROM "Subscription" OFFSET
                                floor(random()*(select count(sub_id) from "Subscription"))
LIMIT 1),
                                (SELECT author_id FROM "Author" OFFSET
                                floor(random()*(select count(author_id) from "Author"))
LIMIT 1),
                                (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int)
                                )),
                                (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int)
                                )))""")

    elif table == 3:
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Subscription"(sub_id, name, email)
                            VALUES (
                                (SELECT(select count(sub_id) from "Subscription")+1::int),
                                (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int)),
                                (SELECT (chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int) ||
                                    chr(ascii('B') + (random() * 25)::int)
                                )))""")

```

```

        chr(ascii('B') + (random() * 25)::int) ||
        chr(ascii('B') + (random() * 25)::int) ||
        '@email.com'
    ))
);""" , )

def update(table, inp):
    connection = connect()

    n_id = inp[0]
    column = inp[1]
    new = inp[2]

    if table == 1:
        if column == 1:
            with connection.cursor() as cursor:
                cursor.execute("""update "Author" set name = %s where
author_id = %s;""", (new, n_id,))

            elif column == 2:
                with connection.cursor() as cursor:
                    cursor.execute("""update "Author" set origin = %s where
author_id = %s;""", (new, n_id,))

        elif table == 2:
            if column == 1:
                with connection.cursor() as cursor:
                    cursor.execute("""update "Book" set sub_id = %s where book_id
= %s;""", (new, n_id,))

            elif column == 2:
                with connection.cursor() as cursor:
                    cursor.execute("""update "Book" set author_id = %s where
book_id = %s;""", (new, n_id,))

            elif column == 3:
                with connection.cursor() as cursor:
                    cursor.execute("""update "Book" set title = %s where book_id
= %s;""", (new, n_id,))

            elif column == 4:
                with connection.cursor() as cursor:
                    cursor.execute("""update "Book" set genre = %s where book_id
= %s;""", (new, n_id,))

        elif table == 3:
            if column == 1:
                with connection.cursor() as cursor:
                    cursor.execute("""update "Subscription" set name = %s where
sub_id = %s;""", (new, n_id,))

            elif column == 2:
                with connection.cursor() as cursor:
                    cursor.execute("""update "Subscription" set email = %s where
sub_id = %s;""", (new, n_id,))

def delete(table, nid):
    connection = connect()

    if table == 1:
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Author" WHERE author_id = %s;""",

```

```

(nid,))

    elif table == 2:
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Book" WHERE book_id = %s;""",
(nid,))

    elif table == 3:
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Subscription" WHERE sub_id =
%s;""", (nid,))

def select_table(table, inp):
    connection = connect()

    string = inp[0]

    if table == 1:
        if string == 1:
            n_id = inp[1]
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Author" where author_id =
%s;""", (n_id,))
            return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Author";""")
            return cursor.fetchall()

    elif table == 2:
        if string == 1:
            n_id = inp[1]
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Book" where book_id =
%s;""", (n_id,))
            return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Book";""")
            return cursor.fetchall()

    elif table == 3:
        if string == 1:
            n_id = inp[1]
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Subscription" where sub_id =
%s;""", (n_id,))
            return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Subscription";""")
            return cursor.fetchall()

def search(table, inp):
    connection = connect()

    column = inp[0]

    if table == 1:

```



```

        if column == 1:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Author" WHERE %s <=
author_id and author_id <= %s;""", (a, b,))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
                return ret

        elif column == 2:
            txt = inp[1]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Author" WHERE
                to tsvector(name) || to tsvector(origin)
                @@ plainto_tsquery(%s);""", (txt,))
                finish = time.time()
                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
                return ret

    elif table == 2:

        if column == 1:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Book" WHERE %s <= book_id
and book_id <= %s;""", (a, b,))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
                return ret

        elif column == 2:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Book" WHERE %s <= sub_id and
sub_id <= %s;""", (a, b,))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
                return ret

        elif column == 3:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Book" WHERE %s <= author_id
and author_id <= %s;""", (a, b,))

```

```

        finish = time.time()

        search_duration = finish - start
        f_a = cursor.fetchall()
        ret = [f_a, search_duration]
        return ret

    elif column == 4:
        txt = inp[1]
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Book" WHERE
                to_tsvector(title) || to_tsvector(genre)
                @@ plainto_tsquery(%s);""", (txt,))
            finish = time.time()

            search_duration = finish - start
            f_a = cursor.fetchall()
            ret = [f_a, search_duration]
            return ret

    elif table == 3:
        if column == 1:
            a = inp[1]
            b = inp[2]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Subscription" WHERE %s <=
sub_id and sub_id <= %s;""", (a, b,))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
                return ret

        elif column == 2:
            txt = inp[1]
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Subscription" WHERE
to_tsvector(name) || to_tsvector(email)
                @@ plainto_tsquery(%s);""", (txt,))
                finish = time.time()

                search_duration = finish - start
                f_a = cursor.fetchall()
                ret = [f_a, search_duration]
                return ret

```

Controller.py

```

from Model import *
from View import *

def menu():
    task = start_menu()

    if task == 1:
        table = tables()
        inp = input_insert(table)

```

```

        try:
            insert(table, inp)
        except Exception as _ex:
            err_except(_ex)
        data(task)

    elif task == 2:
        table = tables()
        if table == 1:
            count = count_rand()
            try:
                insert_rand(table, count)
            except Exception as _ex:
                err_except(_ex)
        else:
            try:
                insert_rand(table, None)
            except Exception as _ex:
                err_except(_ex)
        data(task)

    elif task == 3:
        table = tables()
        inp = input_update(table)
        try:
            update(table, inp)
        except Exception as _ex:
            err_except(_ex)
        data(task)

    elif task == 4:
        table = tables()
        n_id = del_id()
        try:
            delete(table, n_id)
        except Exception as _ex:
            err_except(_ex)
        data(task)

    elif task == 5:
        table = tables()
        inp = input_select()

        try:
            f_t = select_table(table, inp)
            fetch(table, f_t)
        except Exception as _ex:
            err_except(_ex)

    elif task == 6:
        table = tables()
        inp = input_search(table)
        try:
            f_t = search(table, inp)
            fetch(table, f_t[0])
            search_time(f_t[1])
        except Exception as _ex:
            err_except(_ex)

    elif task == 7:
        close()
        print_close()
        return None

```

```
menu()
```

View.py

```
def start_menu():
    sm = int(input("
    | 1 to insert      |\n"
    | 2 to insert random |\n"
    | 3 to update      |\n"
    | 4 to delete       |\n"
    | 5 to select table |\n"
    | 6 to search       |\n"
    | 7 to end          |\n"
    |-----|\n"
    "Input: "))

    if sm < 1 or sm > 7:
        err()
        start_menu()
    else:
        return sm

def tables():
    choice_table = int(input("
    | Select a table:   |\n"
    | 1 = Author        |\n"
    | 2 = Book          |\n"
    | 3 = Subscription  |\n"
    |-----|\n"
    "Input: "))

    if choice_table < 1 or choice_table > 3:
        err()
        tables()
    else:
        return choice_table

def input_insert(table):
    if table == 1:
        id_a = int(input("Input new author id: "))
        name = str(input("Input new author name: "))
        origin = str(input("Input new origin: "))

        inp = [id_a, name, origin]
        return inp

    elif table == 2:
        id_b = int(input("Input new book id: "))
        id_s = int(input("Input new sub id: "))
        id_a = int(input("Input new author id: "))
        title = str(input("Input new title: "))
        genre = str(input("Input new genre: "))

        inp = [id_b, id_s, id_a, title, genre]
        return inp

    elif table == 3:
        id_s = int(input("Input new sub id: "))
        name = str(input("Input new name: "))
        email = str(input("Input new email: "))

        inp = [id_s, name, email]
```

```

        return inp

def count_rand():
    count = int(input("Input amount of data to generate: "))
    return count

def input_update(table):
    new = None
    if table == 1:
        id_c = int(input("Num of author id: "))
        column = int(input("1 = to upd name\n"
                           "2 = to upd origin\n"
                           "Input:"))
        if column == 1:
            new = str(input("New author name: "))
        elif column == 2:
            new = str(input("New origin: "))
        else:
            err()
            input_update(table)

        inp = [id_c, column, new]
        return inp

    elif table == 2:
        id_h = int(input("Num of book id: "))
        column = int(input("1 = to upd sub id\n"
                           "2 = to upd author id\n"
                           "3 = to upd title\n"
                           "4 = to upd genre\n"
                           "Input:"))
        if column == 1:
            new = int(input("New sub id: "))
        elif column == 2:
            new = int(input("New author id: "))
        elif column == 3:
            new = str(input("New title: "))
        elif column == 4:
            new = str(input("New genre: "))
        else:
            err()
            input_update(table)

        inp = [id_h, column, new]
        return inp

    elif table == 3:
        id_cr = int(input("Num of id: "))
        column = int(input("1 = to upd name\n"
                           "2 = to upd email\n"
                           "Input:"))
        if column == 1:
            new = str(input("New name: "))
        elif column == 2:
            new = str(input("New email: "))
        else:
            err()
            input_update(table)

        inp = [id_cr, column, new]
        return inp

```

```

def del_id():
    nid = int(input("Num of id: "))
    return nid

def input_select():
    string = int(input("1 = to one str\n"
                       "2 = to all str\n"
                       "Input:"))

    if string == 1:
        nid = int(input("Num of id: "))
        inp = [string, nid]
        return inp

    elif string == 2:
        inp = [string]
        return inp

    else:
        err()
        input_select()

def input_search(table):
    if table == 1:
        column = int(input("1 = to search author_id\n"
                           "2 = to search name or origin\n"
                           "Input:"))

        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            inp = [column, a, b]
            return inp

        elif column == 2:
            txt = str(input("Input text: "))
            inp = [column, txt]
            return inp

        else:
            err()
            input_search(table)

    elif table == 2:
        column = int(input("1 = to search book_id\n"
                           "2 = to search sub_id\n"
                           "3 = to search author_id\n"
                           "4 = to search title or genre\n"
                           "Input:"))

        if column == 1 or column == 2 or column == 3:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            inp = [column, a, b]
            return inp

        elif column == 4:
            txt = str(input("Input text: "))
            inp = [column, txt]
            return inp

        else:
            err()

```

```

        input_search(table)

    elif table == 3:
        column = int(input("1 = to search sub_id\n"
                           "2 = to search name or email\n"
                           "Input:"))

        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            inp = [column, a, b]
            return inp

        elif column == 2:
            txt = str(input("Input text: "))
            inp = [column, txt]
            return inp

        else:
            err()
            input_search(table)

def fetch(table, f_table):
    if table == 1:
        for i in f_table:
            print("\n"author_id =", i[0])
            print("name =", i[1])
            print("origin =", i[2], "\n")
    elif table == 2:
        for i in f_table:
            print("\n"book_id =", i[0])
            print("sub_id =", i[1])
            print("author_id =", i[2])
            print("title =", i[3])
            print("genre =", i[4], "\n")
    elif table == 3:
        for i in f_table:
            print("\n"sub_id =", i[0])
            print("name =", i[1])
            print("email =", i[2], "\n")

def data(task):
    if task == 1:
        print("\n[Data was successfully inserted]\n")
    elif task == 2:
        print("\n[Random Data was successfully inserted]\n")
    elif task == 3:
        print("\n[Data was successfully updated]\n")
    elif task == 4:
        print("\n[Data was successfully deleted]\n")

def search_time(t):
    print("\nSearch duration = ", t, "\n")

def err():
    print("\n[Error input, try again!]\n")

def print_close():
    print("[PostgreSQL connection closed]")

```

```
def err_except(_ex):  
    print("\n[Error while working with PostgreSQL", _ex, "]\n")
```

Мова програмування – Python 3.8

Середовище розробки програмного забезпечення – PyCharm Community Edition.

Середовище для відлагодження SQL-запитів до бази даних – PgAdmin4.

Використані бібліотеки:

psycopg – для роботи з PostgreSQL

time – для роботи з часом, а саме визначення швидкодії пошуку по базі даних