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-01
Рябенко Б. Ю.
Перевірив:
Павловський В. І.

Київ – 2022

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).

Деталізоване завдання:

1. Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати **вилучення** рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.
2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не мовою програмування, а відповідним SQL-запитом**!

3. Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

4. Програмний код організувати згідно шаблону Model-View-Controller(MVC). При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

Логічна модель бази даних

Нижче (Рисунок 1) наведено логічну модель бази даних:

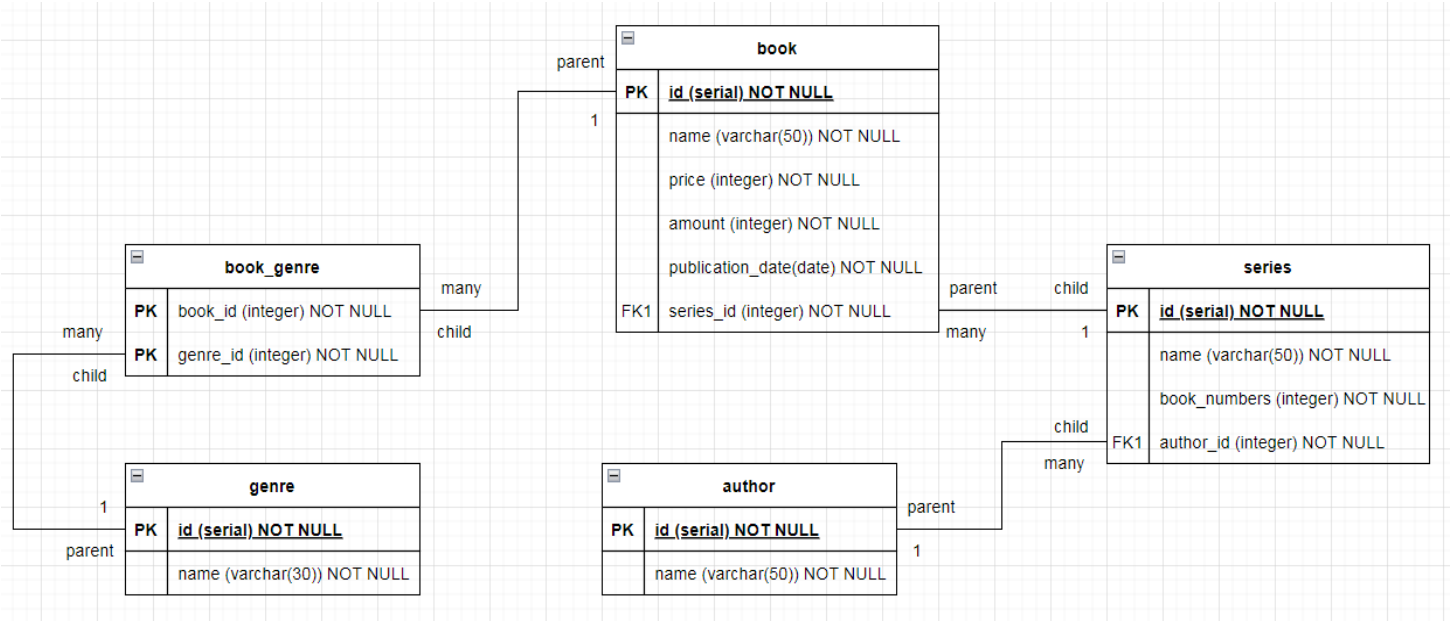

Рисунок 1 – Логічна модель бази даних

Середовище розробки та налаштування підключення до бази даних

Для розробки використовувалась мова програмування Python, середовище розробки PyCharm Community Edition, а також стороння бібліотека, що надає API для доступу до PostgreSQL – psycopg2.

Шаблон проектування

MVC - Шаблон проектування, який використаний у програмі.

Model – представляє клас, що описує логіку використовуваних даних. Згідно компоненту моделі, у моїй програмі відповідають всі компоненти які знаходять у папці Models.

View – в нашому випадку консольний інтерфейс з яким буде взаємодіяти наш користувач. Згідно компоненту представлення, то їй відповідають такі компоненти, згідно яким користувач бачить необхідні дані, що є представленням даних у вигляді консольного інтерфейсу.

Controller – представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує вводяться користувачем

дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання.

Структура програми та її опис

На рисунку 2 відображено деревовидну структуру програми:

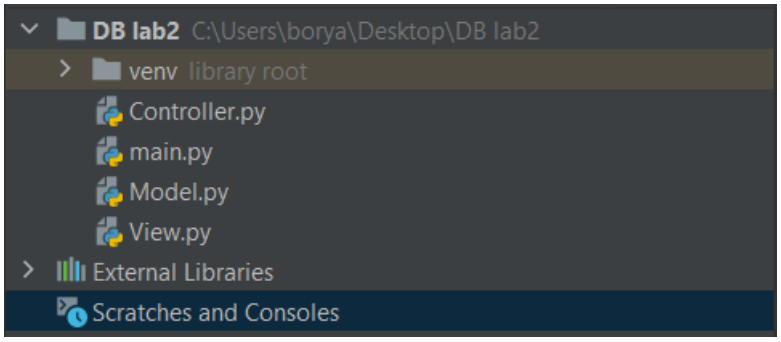

Рисунок 2 – структура програми

Програма поділена на 4 файли: main.py, Controller.py, Model.py, View.py.

У файлі View.py описані функції, що виводять дані у консоль.

У файлі Model.py описаний клас, який реалізує методи управління базою даних

У файлі Controller.py описаний клас, який використовує функції та методи файлів View.py та Model.py.

Структура меню програми

```
What do you want to do?
    1. Insert data in the database
    2. Remove data from database
    3. Edit data in the database
    4. Generate random data in the database
    5. Search data in the database
```

Рисунок 3 – Початкове меню програми

Перший пункт пропонує внесення даних. Користувач повинен вибрати до якої таблиці вносити дані, а потім ввести дані, які будуть додані до таблиці.

Другий пункт пропонує видалення даних. Користувач повинен вибрати з якої таблиці видалити дані, а потім ввести номер рядку.

Третій пункт пропонує редагування даних. Користувач повинен вибрати яку таблицю редагувати та номер рядку яка буде редагуватися, після цього користувач може вводити дані.

Четвертий пункт пропонує заповнення даних випадковим чином. Користувач повинен вибрати яку таблицю заповнювати, а потім скільки всього потрібно даних згенерувати.

П’ятий пункт пропонує пошук за атрибутами з декількох таблиць. Користувач має вибрати, який запит він хоче виконувати, а тоді вже здійснювати відповідний пошук. Після введення необхідних атрибутів він побачить результати пошуку та час самого пошуку.

Лістинги програм

Лістинг фрагменту програми для внесення даних

```
def insert_data(self, table_name, values, column):
        line = '('
        columns = '('
        for key in range(len(values)):
            if values[key]:
                line += column[key] + ','
                columns += f"'{values[key]}'" + ','
        columns = columns[:-1] + ')'
        line = line[:-1] + ')'
        try:
            self.__cursor.execute(
                sql.SQL('INSERT INTO {} {} VALUES
{}').format(sql.Identifier(table_name),
                                    sql.SQL(line), sql.SQL(columns)))
            self.__context.commit()
        except Exception as e:
            print(e)
```

Лістинг фрагменту програми для видалення даних

```
def delete_data(self, table_name, parameter, cond):
        string = f"{parameter}={cond}"
        self.__cursor.execute(
            sql.SQL('DELETE FROM {} WHERE
{}').format(sql.Identifier(table_name), sql.SQL(string)))
        self.__context.commit()
```

Лістинг фрагменту програми для зміни даних

```
def change_data(self, table_name, values, column, cond, id_name):
        string = f"{id_name} = '{cond}'"
        columns = ''
        for key in range(len(values)):
            if values[key]:
                columns += f"{column[key]} = '{values[key]}'" + ','
        columns = columns[:-1] + ''
        try:
            self.__cursor.execute(
                sql.SQL('UPDATE {} SET {} WHERE
{}').format(sql.Identifier(table_name),
                                    sql.SQL(columns), sql.SQL(string)))
            self.__context.commit()
```

```
        except Exception as e:
            print(e)
```

Лістинг фрагменту програми для пошуку даних у таблицях

```
def find_data(self, num):
        if num == 1:
            string = f"SELECT * FROM book WHERE amount > {int(input('Write
min. amount: '))}"
            self.__cursor.execute(string)
            return self.__cursor.fetchall()
        elif num == 2:
            string = f"SELECT * FROM author WHERE name = '{input('Write name:
')}'"
            self.__cursor.execute(string)
            return self.__cursor.fetchall()
        elif num == 3:
            string = f"SELECT * FROM genre WHERE name = '{input('Write name:
')}'"
            self.__cursor.execute(string)
            return self.__cursor.fetchall()
```

Лістинг фрагменту програми для генерації випадкових даних у таблицях

```
def generate_data(self, table_name, count):
        types = self.get_column_types(table_name)
        fk_array = self.get_foreign_key_info(table_name)
        select_subquery = ""
        insert_query = "INSERT INTO " + table_name + " ("
        for i in range(1 if "id" in self.get_table_data(table_name)[0] else
0, len(types)):
            t = types[i]
            name = t[0]
            kind = t[1]
            fk = [x for x in fk_array if x[0] == name]
            if fk:
                select_subquery += ('(SELECT {} FROM {} ORDER BY RANDOM(),
ser LIMIT 1)'.format(fk[0][2], fk[0][1]))
            elif kind == 'integer':
                select_subquery += 'trunc(random()*100)::INT'
            elif kind == 'character varying':
                select_subquery += 'chr(trunc(65 + random()*25)::INT) ||
chr(trunc(65 + random() * 25)::INT)'
            elif kind == 'date':
                select_subquery += """ date(timestamp '2014-01-10' + random()
*(timestamp '2020-01-20' - timestamp '2014-01-10'))"""
            else:
                continue
            insert_query += name
            if i != len(types) - 1:
                select_subquery += ','
                insert_query += ','
            else:
                insert_query += ') '

        self.__cursor.execute(
            insert_query + "SELECT " + select_subquery + "FROM
generate_series(1," + str(count) + ") as ser")
        self.__context.commit()
```

Для генерування випадкових даних спочатку, для кожного типу даних в обраній таблиці, за спеціальним шаблоном генерується рядок, який буде генерувати набір даних певного типу. Потім рядки для кожного стовпчика поєднуються, і на виході маємо готовий SQL-запит для генерації даних.

Дані фрагменти програми, які наведені вище, відповідають за додавання, видалення, зміну, генерування, та пошук даних у базі даних. Ці методи реалізовані у файлі Model.py, які потім використовує файл Controller.py.

Результати роботи програми

```
What do you want to do?
    1. Insert data in the database
    2. Remove data from database
    3. Edit data in the database
    4. Generate random data in the database
    5. Search data in the database
1
1. author
2. series
3. book
4. genre
5. book_genre
Write number of which table you want to change:3
Enter value for column name: book1
Enter value for column price: 23
Enter value for column amount: 109
Enter value for column publication_date: 1500-09-20
Enter value for column series_id: 2
```

Рисунок 4 – Введення даних

	id [PK] integer	name character varying (50)	price integer	amount integer	publication_date date	series_id integer
1	9	The Two Dead Girls	512	60	1996-03-01	1
2	10	The Lightning Thief	513	35	2005-07-01	2
3	11	Harry Potter and the Half-Blood Prin...	456	184	2005-07-16	4
4	12	The Shining	634	89	1977-01-28	6
5	14	Night Journey	570	72	1996-07-25	1
6	15	The Adventure of the Speckled Band	99	12	2010-10-29	3
7	16	The Mark of Athena	312	56	2012-10-02	5
8	17	book1	23	109	1500-09-20	2

Рисунок 5 – Таблиця ‘book’ після внесення до неї даних

```
What do you want to do?
    1. Insert data in the database
    2. Remove data from database
    3. Edit data in the database
    4. Generate random data in the database
    5. Search data in the database
2
1. author
2. series
3. book
4. genre
5. book_genre
Write number of which table you want to change: 4
1.  1   Detective fiction
2.  2   Short stories
3.  3   Fantasy
4.  4   Horror
5.  5   Gothic
6.  6   Dark Fantasy
7.  7   Magic Realism
8.  8   Greek mythology
Choose number:5
```

Рисунок 6 – Видалення даних

	id [PK] integer	name character varying (30)
1	1	Detective fiction
2	2	Short stories
3	3	Fantasy
4	4	Horror
5	6	Dark Fantasy
6	7	Magic Realism
7	8	Greek mythology

Рисунок 7 – Таблиця ‘genre’ після видалення даних

	book_id [PK] integer	genre_id [PK] integer
1	11	3
2	10	8
3	10	3
4	9	7
5	9	6
6	9	5
7	12	5
8	14	7
9	14	6
10	12	4
11	14	5
12	16	8
13	16	3
14	15	2
15	15	1

	book_id [PK] integer	genre_id [PK] integer
1	11	3
2	10	8
3	10	3
4	9	7
5	9	6
6	14	7
7	14	6
8	12	4
9	16	8
10	16	3
11	15	2
12	15	1

Рисунок 8 – Таблиця ‘genre_book’ до та після видалення даних у таблиці ‘genre’

```
What do you want to do?
    1. Insert data in the database
    2. Remove data from database
    3. Edit data in the database
    4. Generate random data in the database
    5. Search data in the database
3
1. author
2. series
3. book
4. genre
5. book_genre
Write number of which table you want to change: 3
1.  9   The Two Dead Girls  512 60  1996-03-01  1
2.  10  The Lightning Thief 513 35  2005-07-01  2
3.  11  Harry Potter and the Half-Blood Prince  456 184 2005-07-16  4
4.  12  The Shining 634 89  1977-01-28  6
5.  14  Night Journey   570 72  1996-07-25  1
6.  15  The Adventure of the Speckled Band  99  12  2010-10-29  3
7.  16  The Mark of Athena  312 56  2012-10-02  5
8.  17  book1   23  109 1500-09-20  2
Choose number:8
Enter value for column name: book2
Enter value for column price: 80
Enter value for column amount: 901
Enter value for column publication_date: 1000-02-19
Enter value for column series_id: 5

Process finished with exit code 0
```

Рисунок 9 – Зміна даних

	id [PK] integer	name character varying (50)	price integer	amount integer	publication_date date	series_id integer
1	9	The Two Dead Girls	512	60	1996-03-01	1
2	10	The Lightning Thief	513	35	2005-07-01	2
3	11	Harry Potter and the Half-Blood Prin...	456	184	2005-07-16	4
4	12	The Shining	634	89	1977-01-28	6
5	14	Night Journey	570	72	1996-07-25	1
6	15	The Adventure of the Speckled Band	99	12	2010-10-29	3
7	16	The Mark of Athena	312	56	2012-10-02	5
8	17	book2	80	901	1000-02-19	5

Рисунок 10 – Таблиця ‘book’ після зміни даних

```
What do you want to do?
    1. Insert data in the database
    2. Remove data from database
    3. Edit data in the database
    4. Generate random data in the database
    5. Search data in the database
4
1. author
2. series
3. book
4. genre
5. book_genre
Write number of which table you want to change: 3
Write how much to generate: 100000

Process finished with exit code 0
```

Рисунок 11 – Генерація випадкових значень

	id [PK] integer	name character varying (50)	price integer	amount integer	publication_date date	series_id integer
7720	7729	FR	29	15	2017-04-26	6
7721	7730	MA	47	7	2016-06-20	5
7722	7731	JS	41	58	2018-10-02	3
7723	7732	LF	22	61	2016-08-30	3
7724	7733	OG	36	85	2015-02-16	6
7725	7734	IC	67	10	2015-12-31	6
7726	7735	NR	28	87	2014-01-28	1
7727	7736	XY	82	93	2016-12-03	3
7728	7737	HK	48	82	2019-02-17	3
7729	7738	GX	62	22	2019-08-16	3
7730	7739	TF	41	2	2014-07-29	6
7731	7740	DE	1	49	2017-09-25	6
7732	7741	TP	21	43	2019-03-13	6
7733	7742	AV	89	8	2015-02-12	3
7734	7743	NO	47	83	2019-12-15	6
7735	7744	CL	82	66	2014-05-15	6
7736	7745	GS	44	34	2019-10-02	3
7737	7746	PR	33	58	2018-03-12	1
7738	7747	AT	97	77	2014-06-09	3
7739	7748	XG	51	45	2015-11-17	2
7740	7749	UE	90	78	2018-07-17	6

Рисунок 12 – таблиця ‘book’ після генерації випадкових значень

```
What do you want to do?
    1. Insert data in the database
    2. Remove data from database
    3. Edit data in the database
    4. Generate random data in the database
    5. Search data in the database
5
Choose what to search:
    1. Search books which amount more than N
    2. Search authors with name N
    3. Search genres with name N
1
Write min. amount: 70
```

```
28906.  99959   AE  40  91  2017-01-18  2
28907.  99961   IV  44  95  2017-05-29  6
28908.  99966   VF  63  92  2014-05-18  5
28909.  99967   SW  36  78  2019-12-14  2
28910.  99971   EO  56  96  2015-12-06  2
28911.  99973   DT  30  77  2015-06-11  4
28912.  99974   PE  19  95  2018-04-04  2
28913.  99981   IE  85  91  2019-01-18  6
28914.  99982   YK  34  71  2018-09-16  2
28915.  99983   BU  84  83  2015-10-31  1
28916.  99991   OI  64  98  2017-08-17  3
28917.  99994   MU  69  95  2016-12-29  4
28918.  99997   MI  98  95  2016-02-03  6
28919.  99999   MK  12  86  2014-08-21  6
28920.  100004  SU  51  86  2019-09-02  3
28921.  100008  ON  41  95  2015-02-08  6
28922.  100011  OX  6   95  2015-10-20  5
28923.  100013  AA  73  88  2016-02-15  6
Time of searching: 13.71915430000081 milliseconds
```

Рисунок 13 – Пошук даних

Текст програми

Main.py

```
from Controller import *
a = Controller()
```

View.py

```
def tables_print(tables):
    for i in range(len(tables)):
        print(f"{i+1}.", tables[i])
    return tables[int(input("Write number of which table you want to change:
"))-1]


def table_print(rows):
    number = 1
    for line in rows:
        string = f"{number}.\t"
        for elem in range(len(line)):
            string += str(line[elem]) + "\t"
        print(string)
        number += 1
    return number


def menu():
    print("""What do you want to do?
    1. Insert data in the database
    2. Remove data from database
    3. Edit data in the database
    4. Generate random data in the database
    5. Search data in the database""")
    n = int(input())
    if 1 <= n <= 5:
        return n
    else:
        print("Wrong parameter")


def find_data_menu():
    print("""Choose what to search:
    1. Search books which amount more than N
    2. Search authors with name N
    3. Search genres with name N""")
```

Model.py

```
import psycopg2
from psycopg2 import sql


class Model:
    def __init__(self, dbname, user_name, password, host):
        try:
            self.__context = psycopg2.connect(host=host, user=user_name,
password=password, database=dbname)
            self.__cursor = self.__context.cursor()
            self.__table_names = None
        except Exception as _ex:
```

```
            print("[INFO] Error while working with PostgresSQL", _ex)

    def __del__(self):
        if self.__context:
            self.__cursor.close()
            self.__context.close()

    def get_table_names(self):
        if self.__table_names is None:
            self.__cursor.execute("""SELECT table_name
                                     FROM information_schema.tables
                                     WHERE table_schema = 'public'""")
            self.__table_names = [table[0] for table in self.__cursor]
        return self.__table_names

    def get_column_types(self, table_name):
        self.__cursor.execute("""SELECT column_name, data_type
            FROM information_schema.columns
           WHERE table_schema = 'public' AND table_name = %s
           ORDER BY table_schema, table_name""", (table_name,))
        return self.__cursor.fetchall()

    def get_column_names(self, table_name):
        self.__cursor.execute("""
            SELECT column_name FROM information_schema.columns
            WHERE table_schema = 'public' AND table_name = %s
            ORDER BY table_schema, table_name""", (table_name,))
        return [x[0] for x in self.__cursor.fetchall()]

    def get_table_data(self, table_name):
        id_column = self.get_column_types(table_name)[0][0]
        cursor = self.__cursor
        try:
            cursor.execute(
                sql.SQL('SELECT * FROM {} ORDER BY {}
ASC').format(sql.Identifier(table_name), sql.SQL(id_column)))
        except Exception as e:
            return str(e)
        return [col.name for col in cursor.description], cursor.fetchall()

    def get_foreign_key_info(self, table_name):
        self.__cursor.execute("""
           SELECT kcu.column_name, ccu.table_name AS
           foreign_table_name, ccu.column_name AS foreign_column_name
           FROM information_schema.table_constraints AS tc
              JOIN information_schema.key_column_usage AS kcu
                 ON tc.constraint_name = kcu.constraint_name
                 AND tc.table_schema = kcu.table_schema
                JOIN information_schema.constraint_column_usage AS ccu
                 ON ccu.constraint_name = tc.constraint_name
                 AND ccu.table_schema = tc.table_schema
           WHERE tc.constraint_type = 'FOREIGN KEY' AND
                        tc.table_name=%s;""", (table_name,))
        return self.__cursor.fetchall()

    def insert_data(self, table_name, values, column):
        line = '('
        columns = '('
        for key in range(len(values)):
```

```
            if values[key]:
                line += column[key] + ','
                columns += f"'{values[key]}'" + ','
        columns = columns[:-1] + ')'
        line = line[:-1] + ')'
        try:
            self.__cursor.execute(
                sql.SQL('INSERT INTO {} {} VALUES
{}').format(sql.Identifier(table_name),
                                                                sql.SQL(line),
sql.SQL(columns)))
            self.__context.commit()
        except Exception as e:
            print(e)

    def delete_data(self, table_name, parameter, cond):
        string = f"{parameter}={cond}"
        self.__cursor.execute(
            sql.SQL('DELETE FROM {} WHERE
{}').format(sql.Identifier(table_name), sql.SQL(string)))
        self.__context.commit()

    def change_data(self, table_name, values, column, cond, id_name):
        string = f"{id_name} = '{cond}'"
        columns = ''
        for key in range(len(values)):
            if values[key]:
                columns += f"{column[key]} = '{values[key]}'" + ','
        columns = columns[:-1] + ''
        try:
            self.__cursor.execute(
                sql.SQL('UPDATE {} SET {} WHERE
{}').format(sql.Identifier(table_name),
                                                            sql.SQL(columns),
sql.SQL(string)))
            self.__context.commit()
        except Exception as e:
            print(e)

    def generate_data(self, table_name, count):
        types = self.get_column_types(table_name)
        fk_array = self.get_foreign_key_info(table_name)
        select_subquery = ""
        insert_query = "INSERT INTO " + table_name + " ("
        for i in range(1 if "id" in self.get_table_data(table_name)[0] else
0, len(types)):
            t = types[i]
            name = t[0]
            kind = t[1]
            fk = [x for x in fk_array if x[0] == name]
            if fk:
                select_subquery += ('(SELECT {} FROM {} ORDER BY RANDOM(),
ser LIMIT 1)'.format(fk[0][2], fk[0][1]))
            elif kind == 'integer':
                select_subquery += 'trunc(random()*100)::INT'
            elif kind == 'character varying':
                select_subquery += 'chr(trunc(65 + random()*25)::INT) ||
chr(trunc(65 + random() * 25)::INT)'
            elif kind == 'date':
```

```
                select_subquery += """ date(timestamp '2014-01-10' + random()
*
                (timestamp '2020-01-20' - timestamp '2014-01-10'))"""
            else:
                continue
            insert_query += name
            if i != len(types) - 1:
                select_subquery += ','
                insert_query += ','
            else:
                insert_query += ') '

        self.__cursor.execute(
            insert_query + "SELECT " + select_subquery + "FROM
generate_series(1," + str(count) + ") as ser")
        self.__context.commit()

    def find_data(self, num):
        if num == 1:
            string = f"SELECT * FROM book WHERE amount > {int(input('Write
min. amount: '))}"
            self.__cursor.execute(string)
            return self.__cursor.fetchall()
        elif num == 2:
            string = f"SELECT * FROM author WHERE name = '{input('Write name:
')}'"
            self.__cursor.execute(string)
            return self.__cursor.fetchall()
        elif num == 3:
            string = f"SELECT * FROM genre WHERE name = '{input('Write name:
')}'"
            self.__cursor.execute(string)
            return self.__cursor.fetchall()
```

Controller.py

```
from View import *
from Model import *
import time


def insert(model):
    table = tables_print(model.get_table_names())
    table_data = model.get_table_data(table)
    if 'id' in table_data[0]:
        table_data[0].remove('id')
    values = []
    for i in range(len(table_data[0])):
        values.append(input(f"Enter value for column {table_data[0][i]}: "))
    model.insert_data(table, values, table_data[0])


def delete(model):
    table = tables_print(model.get_table_names())
    table_data = model.get_table_data(table)
    table_print(table_data[1])
    model.delete_data(table, table_data[0][0],
table_data[1][int(input('Choose number:')) - 1][0])
```

```
def change(model):
    table = tables_print(model.get_table_names())
    table_data = model.get_table_data(table)
    table_print(table_data[1])
    id_name = table_data[0][0]
    num = table_data[1][int(input('Choose number:')) - 1][0]
    if 'id' in table_data[0]:
        table_data[0].remove('id')
    values = []
    for i in range(len(table_data[0])):
        values.append(input(f"Enter value for column {table_data[0][i]}: "))
    model.change_data(table, values, table_data[0], num, id_name)


def random(model):
    table = tables_print(model.get_table_names())
    model.generate_data(table, int(input("Write how much to generate: ")))


def find(model):
    find_data_menu()
    n = int(input())
    data = model.find_data(n) if 1 <= n <= 3 else print("Wrong parameter")
    table_print(data)


class Controller:
    def __init__(self):
        self.__operate = menu()
        if self.__operate:
            self.operation()

    def operation(self):
        model = Model("book_shop", "postgres", "qwerty", "127.0.0.1")
        if self.__operate == 1:
            insert(model)
        elif self.__operate == 2:
            delete(model)
        elif self.__operate == 3:
            change(model)
        elif self.__operate == 4:
            random(model)
        elif self.__operate == 5:
            t1 = time.perf_counter()
            find(model)
            t2 = time.perf_counter()
            print(f"Time of searching: {t2-t1} milliseconds")
```