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 СУБДPostgreSQL "

Виконав:

студент III курсу

групи КВ-93

Гаращук Б.В

Перевірив:

Київ – 2021

Постановка задачі

Метою роботи є здобуття вмінь програмування прикладних додатків базданих PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення данних 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 дыапазону дат.
4. Програмний код виконати згыдно шаблону MVC (модель – подання - контролер).

Інформація про програму

Посилання на репозиторій GitHub з вихідним кодом програми та звітом: https://github.com/Bodgen/database/tree/main/Lab2

Використана мова програмування: Python 3.7.2

Використані бібліотеки: psycopg2 (для зв’язку з СУБД), time (для виміру часу запиту пошуку завдання 3), sys (для реалізації консольного інтерфейсу).

Відомості про обрану предметну галузь з лабораторної роботи №1

Обрана предметна галузь передбачає можливість користувачеві вибрати зі списку товар і додати його у свою корзину.

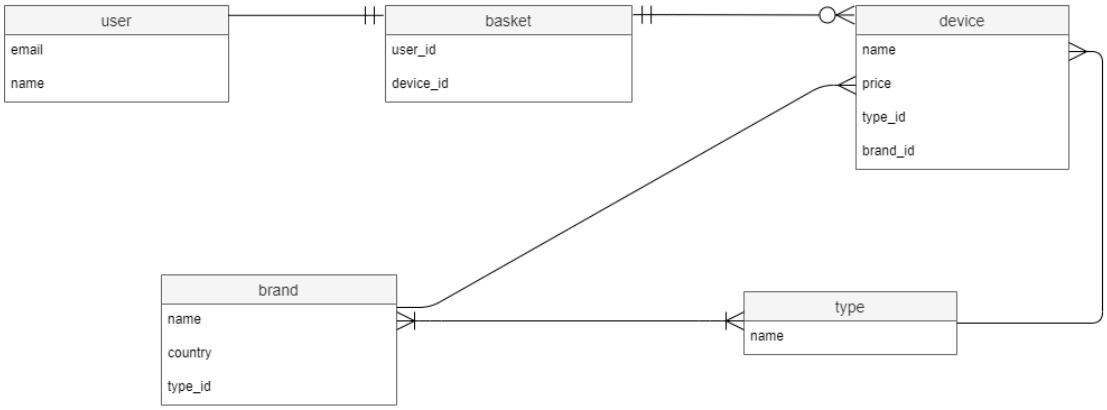
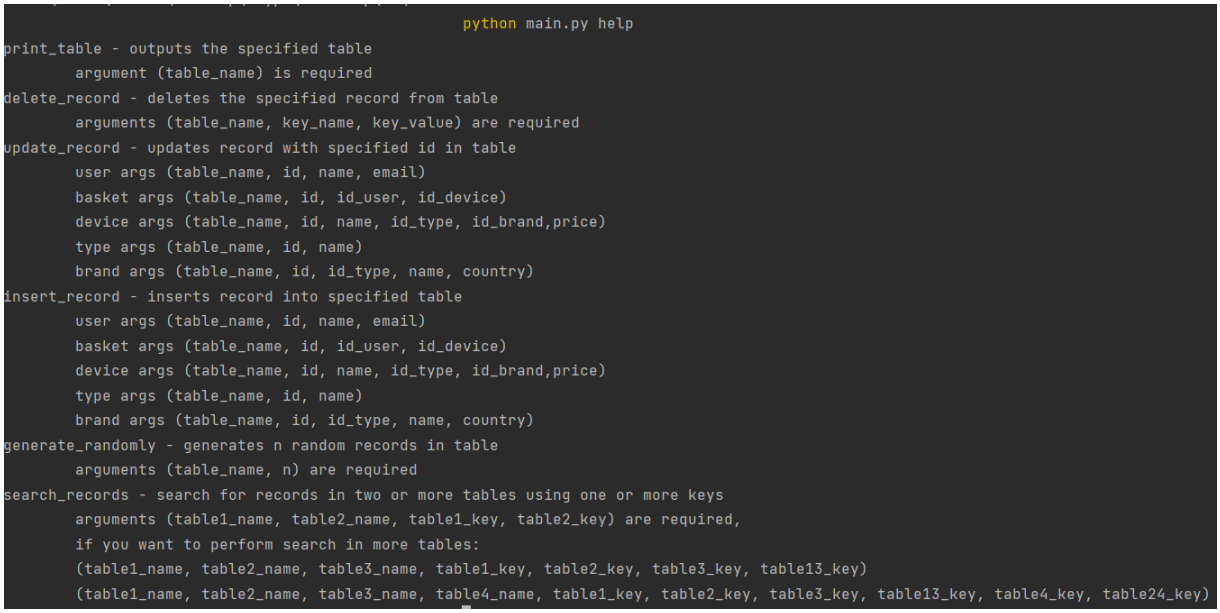


Рисунок 1. Схема бази даних

Таблиця 1. Опис структури БД

Відношення	Атрибут	Тип (розмір)
Відношення "user" Вміщує інформацію про користувача	**id** - унікальний ID користувача **email** - електронна пошта користувача **name** – ім'я користувача	Serial primary key Текстовий (255) Текстовий (255)
Відношення "basket" Вміщує інформацію про те,який товар купив користувач	**id** - унікальний ID кошика **user _ id** – атрибут,який посилається на ID користувача **device_ id** - атрибут,який посилається на ID товару	Serial primary key Числовий Числовий
Відношення "device" Вміщує інформацію про товар	**id** - унікальний ID товару **name** – назва товару **price –** вартість товару **type_id** - атрибут, який посилається на ID типу товару **brand_id –** атрибут,який посилається на ID бренда	Serial primary key Текстовий (255) Числовий Числовий Числовий
Відношення "brand" Вміщує інформацію про бренд	**id** - унікальний ID бренда **type_id** - атрибут, який посилається на ID типу товару **name** – назва бренду **country** – країна виробник	Serial primary key Числовий Текстовий (255) Текстовий (255)
Відношення "type" Вміщує інформацію про тип товару	**id** - унікальний ID типу товара **name** – назва типу товара	Serial primary key Текстовий (255)

Схема меню користувача



На знімку екрану терміналупродемонстровано використання команди help, що показує усі доступні користувачу команди, коротко описує їх та надає список обов'язкових аргументів. Кожна команда запускає відповідний метод об'єкту класа Controller, який реалізує передачу аргументів у клас View на перевірку і за умови їх коректності, Controller далі передає ці аргументи у клас Model, що здійснює запит до бази даних.

Методи реалізовані до пункту 1 завдання лабораторної роботи:

print_table – за умови, коректності імені таблиці виводисть вміст цієї таблиці, у вікно терміналу. Аргументом може бути одне із імен:

user, basket, device, type, brand.

delete_record – за умови правильності введених аргументів, наявності відповідного запису (з вказаним значенням первинного клоюча) та залежності інших таблиць від цього запису (до цього запису немає зовнішнього ключа з іншої таблиці), видаляє запис з вказаним первинним ключем. Аргументами є:

table_name, key_name, key_value.

update_record – за умови правильності введених аргументів, наявності відповідного запису (з вказаним значенням первинного ключа) та записів інших таблиць (на які хочемо змінити поточні), змінює усі поля, окрім первинного ключа у обраному записі. Аргументи різні для кожної таблиці:

user: id(int), name(str), email(str)

basket: id(int), device_id(int), user_id(int)

device: id(int), price(int), type_id(int), brand_id(int), name(str)

type: id(int), name(str)

brand: id(int), type_id(int), name(str), country(str)

insert_record – за умови правильності введених аргументів, відсутності відповідного запису (з вказаним значенням первинного ключа) та наявності записів інших таблиць (на які хочемо посилатись зі створеного запису), вставляє новий рядок у таблицю з обраними значеннями полів. Аргументи різні для кожної таблиці:

user: id(int), name(str), email(str)

basket: id(int), device_id(int), user_id(int)

device: id(int), price(int), type_id(int), brand_id(int), name(str)

type: id(int), name(str)

brand: id(int), type_id(int), name(str), country(str)

Метод реалізований до пункту 2 завдання лабораторної роботи:

generate_randomly – за умови введення правильного імені таблиці та числа n відмінного від нуля, здійснюється генерування n псевдорандомізованих записів у обраній таблиці. Аргументами є ім’я таблиці та число записів, що мають бути створені.

Метод реалізований до пункту 3 завдання лабораторної роботи:

search_records – за умови введення потрібної кількості аргументів та правильного задання умов пошуку, реалізує пошук за 1 та більше атрибутами з вказаних таблиць (від двох до п’яти) і виводить у вікно терміналу результат пошуку (або нічого, якщо пошук не дав результатів) та час, за який було проведено запит. Початково потрібно вказати аргументи.

Після вказання цієї інформації потрібно буде вказати кількість атрибутів для пошук, та тип пошуку, ім’я (обов’язково з вказанням до якої таблиці з перелічених аргументів він відноситься: one.key_name, two.key_name, three.key_name, four.key_name або five.key_name), та значення (спочатку лівий кінець інтервалу, потім правий для числового пошуку та пошуку за датою, або рядок для пошуку за ключовим словом). Спочатку вказуються всі дані для першого атрибуту, потім для другого і т.д. до введеної кількості атрибутів.

Завдання 1

Запит на видалення

Для видалення роботи розглянемо запити на видалення дочірньої таблиці user та basket.

Таблиця basket до видалення запису 4:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table basket
SELECT * FROM public."basket"
Basket table:
id: 3   device_id: 2    id_user: 1
-------------------------------------
id: 4   device_id: 1    id_user: 1
-------------------------------------
id: 5   device_id: 7    id_user: 2
-------------------------------------
id: 6   device_id: 3    id_user: 2
-------------------------------------
id: 7   device_id: 2    id_user: 2
-------------------------------------
id: 8   device_id: 7    id_user: 1
-------------------------------------
id: 9   device_id: 4    id_user: 2
-------------------------------------
```

Таблиця basket після видалення запису 4:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table basket
SELECT * FROM public."basket"
Basket table:
id: 3   device_id: 2    id_user: 1
-------------------------------------
id: 4   device_id: 1    id_user: 1
-------------------------------------
id: 5   device_id: 7    id_user: 2
-------------------------------------
id: 6   device_id: 3    id_user: 2
-------------------------------------
id: 7   device_id: 2    id_user: 2
-------------------------------------
id: 8   device_id: 7    id_user: 1
-------------------------------------
id: 9   device_id: 4    id_user: 2
-------------------------------------
```

У даній програмній реалізації видалення запису з батьківської таблиці, який зв’язаний з дочірньою таблицею, не буде здійснено, а буде видано повідомлення про помилку.

Таблиця user до видалення запису 4:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table user
SELECT * FROM public."user"
User table:
id: 1   name: Pavlo     email: pavlo@gmail.com
-------------------------------------
id: 2   name: Feofan    email: Feofan@gmail.com
-------------------------------------
id: 3   name: Alina     email: Alina@gmail.com
-------------------------------------
id: 4   name: Vova      email: vova@gmail.com
-------------------------------------
```

Спроба видалення запису 4 з таблиці basket:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py delete_record user id 1
select count(*) from public."user" where id=1
select count(*) from public."basket" where user_id=1
this record is connected with another table, deleting will throw error
```

Запит на вставку поля

Для перевірки роботи розглянемо запити на вставки в дочірню таблицю device. Спочатку коректний, потім з неіснуючим значеням зовнішнього ключа батьківської таблиці brand.

Таблиця device до вставки запису 4:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table user
SELECT * FROM public."user"
User table:
id: 1   name: Pavlo     email: pavlo@gmail.com
-------------------------------------
id: 2   name: Feofan    email: Feofan@gmail.com
-------------------------------------
id: 3   name: Alina     email: Alina@gmail.com
-------------------------------------
id: 4   name: Vova      email: vova@gmail.com
-------------------------------------
```

Таблиця device після вставки запису 4:

```
-----------------------------------
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py insert_record device 4 1599 2 5 device_5
select count(*) from public."device" where id=4
select count(*) from public."type" where id=2
select count(*) from public."brand" where id=5
insert into public."device" (id, price, type_id,brand_id,name ) VALUES (4, '1599', 2, '5','device_5');
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table device
SELECT * FROM public."device"
Device table:
id: 1   price: 500      id_type: 1      id_brand: 1     name: device_1
-----------------------------------
id: 2   price: 7000     id_type: 2      id_brand: 3     name: device_2
-----------------------------------
id: 3   price: 100      id_type: 1      id_brand: 2     name: device_3
-----------------------------------
id: 5   price: 1000     id_type: 1      id_brand: 2     name: device_4
-----------------------------------
id: 4   price: 1599     id_type: 2      id_brand: 5     name: device_5
-----------------------------------
```

Записи у батьківській таблиці brand:

```
-----------------------------------
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py insert_record device 4 1599 2 5 device_5
select count(*) from public."device" where id=4
select count(*) from public."type" where id=2
select count(*) from public."brand" where id=5
insert into public."device" (id, price, type_id,brand_id,name ) VALUES (4, '1599', 2, '5','device_5');
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table device
SELECT * FROM public."device"
Device table:
id: 1   price: 500      id_type: 1      id_brand: 1     name: device_1
-----------------------------------
id: 2   price: 7000     id_type: 2      id_brand: 3     name: device_2
-----------------------------------
id: 3   price: 100      id_type: 1      id_brand: 2     name: device_3
-----------------------------------
id: 5   price: 1000     id_type: 1      id_brand: 2     name: device_4
-----------------------------------
id: 4   price: 1599     id_type: 2      id_brand: 5     name: device_5
-----------------------------------
```

Спроба вставки запису у дочірню таблицю device з неіснуючим зовнішнім ключем 7:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py insert_record device 5 1599 2 8 device_8
select count(*) from public."device" where id=5
select count(*) from public."type" where id=2
select count(*) from public."brand" where id=8
Something went wrong (record with such id exists or inappropriate foreign key values)
```

Запит на зміну полів

Для перевірки роботи розглянемо запити на зміну значення в дочірній таблиці device. Спочатку коректний, потім з неіснуючим значенням зовнішнього ключа батьківської таблиці type.

Таблиця device до зміни запису 3:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table device
SELECT * FROM public."device"
Device table:
id: 1   price: 500      id_type: 1      id_brand: 1     name: device_1
-----------------------------------
id: 2   price: 7000     id_type: 2      id_brand: 3     name: device_2
-----------------------------------
id: 3   price: 100      id_type: 1      id_brand: 2     name: device_3
-----------------------------------
id: 5   price: 1000     id_type: 1      id_brand: 2     name: device_4
-----------------------------------
```

Таблиця device після зміни запису 4:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py update_record device 3 2099 3 4 device_6
select count(*) from public."device" where id=3
select count(*) from public."type" where id=3
select count(*) from public."brand" where id=4
UPDATE public."device" SET price='2099', type_id=3, brand_id='4',name ='device_6' WHERE id=3;
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table device
SELECT * FROM public."device"
Device table:
id: 1   price: 500      id_type: 1      id_brand: 1     name: device_1
-----------------------------------
id: 2   price: 7000     id_type: 2      id_brand: 3     name: device_2
-----------------------------------
id: 5   price: 1000     id_type: 1      id_brand: 2     name: device_4
-----------------------------------
id: 3   price: 2099     id_type: 3      id_brand: 4     name: device_6
-----------------------------------
```

Записи у батьківські таблиці type:

```
SELECT * FROM public."type"
Type table:
id: 1   name: type_1
-------------------------------------
id: 2   name: type_2
-------------------------------------
id: 3   name: play
-------------------------------------
id: 4   name: laptop
-------------------------------------
id: 5   name: ctuqeotbhlpdjrnp
-------------------------------------
id: 6   name: blfvcxggfwpnsqvdqjjlnst
-------------------------------------
```

Спроба зміни запису у дочірній таблиці device з неіснуючим зовнішнім ключем 5:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py update_record device 5 2099 7 4 device_6
select count(*) from public."device" where id=5
select count(*) from public."type" where id=7
select count(*) from public."brand" where id=4
Something went wrong (record with such id does not exist or inappropriate foreign key value)
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2>
```

Завдання 2

Вставки 5 псевдорандомызованих записів у кожну з таблиць

Початкова таблиця user:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table user
SELECT * FROM public."user"
User table:
id: 1   name: Pavlo     email: pavlo@gmail.com
-------------------------------------
id: 2   name: Feofan    email: Feofan@gmail.com
-------------------------------------
id: 3   name: Alina     email: Alina@gmail.com
-------------------------------------
id: 4   name: Oleg      email: oleg@email.com
-------------------------------------
```

Запит:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py generate_randomly user 5
insert into public."user"select (SELECT MAX(id)+1 FROM public."user"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(
25-10)+10):: integer)), ''), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), '');
insert into public."user"select (SELECT MAX(id)+1 FROM public."user"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(
25-10)+10):: integer)), ''), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), '');
insert into public."user"select (SELECT MAX(id)+1 FROM public."user"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(
25-10)+10):: integer)), ''), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), '');
insert into public."user"select (SELECT MAX(id)+1 FROM public."user"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(
25-10)+10):: integer)), ''), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), '');
insert into public."user"select (SELECT MAX(id)+1 FROM public."user"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(
25-10)+10):: integer)), ''), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), '');
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2>
```

Модифікована таблиця user:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table user
SELECT * FROM public."user"
User table:
id: 1   name: Pavlo     email: pavlo@gmail.com
-------------------------------------
id: 2   name: Feofan    email: Feofan@gmail.com
-------------------------------------
id: 3   name: Alina     email: Alina@gmail.com
-------------------------------------
id: 4   name: Oleg      email: oleg@email.com
-------------------------------------
id: 5   name: smjpgvuwzbtfldgup         email: wldqwinfu
-------------------------------------
id: 6   name: knrkxwewjkysyzxtyqsatls   email: fknjxxrf
-------------------------------------
id: 7   name: yswnhvpfraxk      email: tdew
-------------------------------------
id: 8   name: feoohgqqeopsww    email: pmmdtv
-------------------------------------
id: 9   name: hfehxldtkpvfndsbskdpucwz  email: biae
-------------------------------------
```

Початкова таблиця basket:

```
SELECT * FROM public."basket"
Basket table:
id: 1   device_id: 2    id_user: 1
-------------------------------------
id: 2   device_id: 3    id_user: 4
-------------------------------------
id: 3   device_id: 1    id_user: 5
-------------------------------------
```

Запит:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py generate_randomly basket 5
insert into public."basket" select (SELECT (MAX(id)+1) FROM public."basket"), (SELECT id FROM public."device" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."device")-
1)))),(SELECT id FROM public."user" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."user")-1))));
insert into public."basket" select (SELECT (MAX(id)+1) FROM public."basket"), (SELECT id FROM public."device" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."device")-
1)))),(SELECT id FROM public."user" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."user")-1))));
insert into public."basket" select (SELECT (MAX(id)+1) FROM public."basket"), (SELECT id FROM public."device" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."device")-
1)))),(SELECT id FROM public."user" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."user")-1))));
insert into public."basket" select (SELECT (MAX(id)+1) FROM public."basket"), (SELECT id FROM public."device" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."device")-
1)))),(SELECT id FROM public."user" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."user")-1))));
insert into public."basket" select (SELECT (MAX(id)+1) FROM public."basket"), (SELECT id FROM public."device" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."device")-
1)))),(SELECT id FROM public."user" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."user")-1))));
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2>
```

Модифікована таблиця basket:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table basket
SELECT * FROM public."basket"
Basket table:
id: 1   device_id: 2    id_user: 1
-------------------------------------
id: 2   device_id: 3    id_user: 4
-------------------------------------
id: 3   device_id: 1    id_user: 5
-------------------------------------
id: 4   device_id: 3    id_user: 2
-------------------------------------
id: 5   device_id: 5    id_user: 9
-------------------------------------
id: 6   device_id: 2    id_user: 3
-------------------------------------
id: 7   device_id: 1    id_user: 8
-------------------------------------
id: 8   device_id: 1    id_user: 5
-------------------------------------
```

Початкова таблиця device:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table device
SELECT * FROM public."device"
Device table:
id: 1   price: 500      id_type: 1      id_brand: 1     name: device_1
-------------------------------------
id: 2   price: 7000     id_type: 2      id_brand: 3     name: device_2
-------------------------------------
id: 3   price: 2099     id_type: 3      id_brand: 4     name: device_6
-------------------------------------
id: 5   price: 2099     id_type: 1      id_brand: 2     name: device_6
-------------------------------------
```

Запит:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py generate_randomly device 5
insert into public."device" select (SELECT MAX(id)+1 FROM public."device"), FLOOR(RANDOM()*(100000-1)+1),(SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id
) FROM public."type")-1)))), (SELECT id FROM public."brand" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."brand")-1)))),array_to_string(ARRAY(SELECT chr((97 + round(
random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '')  ;
insert into public."device" select (SELECT MAX(id)+1 FROM public."device"), FLOOR(RANDOM()*(100000-1)+1),(SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id
) FROM public."type")-1)))), (SELECT id FROM public."brand" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."brand")-1)))),array_to_string(ARRAY(SELECT chr((97 + round(
random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '')  ;
insert into public."device" select (SELECT MAX(id)+1 FROM public."device"), FLOOR(RANDOM()*(100000-1)+1),(SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id
) FROM public."type")-1)))), (SELECT id FROM public."brand" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."brand")-1)))),array_to_string(ARRAY(SELECT chr((97 + round(
random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '')  ;
insert into public."device" select (SELECT MAX(id)+1 FROM public."device"), FLOOR(RANDOM()*(100000-1)+1),(SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id
) FROM public."type")-1)))), (SELECT id FROM public."brand" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."brand")-1)))),array_to_string(ARRAY(SELECT chr((97 + round(
random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '')  ;
insert into public."device" select (SELECT MAX(id)+1 FROM public."device"), FLOOR(RANDOM()*(100000-1)+1),(SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id
) FROM public."type")-1)))), (SELECT id FROM public."brand" LIMIT 1 OFFSET (round(random() * ((SELECT COUNT(id) FROM public."brand")-1)))),array_to_string(ARRAY(SELECT chr((97 + round(
random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '')  ;
```

Модифікована таблиця device:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table device
SELECT * FROM public."device"
Device table:
id: 1    price: 500      id_type: 1      id_brand: 1     name: device_1
-------------------------------------
id: 2    price: 7000     id_type: 2      id_brand: 3     name: device_2
-------------------------------------
id: 3    price: 2099     id_type: 3      id_brand: 4     name: device_6
-------------------------------------
id: 5    price: 2099     id_type: 1      id_brand: 2     name: device_6
-------------------------------------
id: 6    price: 53216    id_type: 5      id_brand: 7     name: fetikgwjtebpett
-------------------------------------
id: 7    price: 75296    id_type: 2      id_brand: 1     name: vzzkywtpkvxvjqogqj
-------------------------------------
id: 8    price: 52362    id_type: 2      id_brand: 5     name: okogubskzohixutyzxyng
-------------------------------------
id: 9    price: 21534    id_type: 2      id_brand: 4     name: pymuypqnitkcfiqjvhmlzjbw
-------------------------------------
id: 10   price: 20421    id_type: 5      id_brand: 5     name: werjwankgomuxuepefd
-------------------------------------
```

Початкова таблиця type:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table type
SELECT * FROM public."type"
Type table:
id: 1   name: type_1
-------------------------------------
id: 2   name: type_2
-------------------------------------
id: 3   name: play
-------------------------------------
id: 4   name: laptop
-------------------------------------
id: 5   name: ctuqeotbhlpdjrnp
-------------------------------------
id: 6   name: blfvcxggfwpnsqvdqjjlnst
-------------------------------------
```

Запит:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py generate_randomly type 5
insert into public."type" select (SELECT MAX(id)+1 FROM public."type"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*
(25-10)+10):: integer)), '');
insert into public."type" select (SELECT MAX(id)+1 FROM public."type"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*
(25-10)+10):: integer)), '');
insert into public."type" select (SELECT MAX(id)+1 FROM public."type"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*
(25-10)+10):: integer)), '');
insert into public."type" select (SELECT MAX(id)+1 FROM public."type"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*
(25-10)+10):: integer)), '');
insert into public."type" select (SELECT MAX(id)+1 FROM public."type"), array_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*
(25-10)+10):: integer)), '');
```

Модифікована таблиця type:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table type
SELECT * FROM public."type"
Type table:
id: 1   name: type_1
-------------------------------------
id: 2   name: type_2
-------------------------------------
id: 3   name: play
-------------------------------------
id: 4   name: laptop
-------------------------------------
id: 5   name: ctuqeotbhlpdjrnp
-------------------------------------
id: 6   name: blfvcxggfwpnsqvdqjjlnst
-------------------------------------
id: 7   name: ujhrfjsdxcwuiuswzkltk
-------------------------------------
id: 8   name: akkwfshfrelhvuvqe
-------------------------------------
id: 9   name: mvnnxmpwpfyprfeu
-------------------------------------
id: 10  name: jkmoypmstknufmoky
-------------------------------------
id: 11  name: rhjgcgsfbozwpuuchpwytgt
-------------------------------------
```

Початкова таблиця brand:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table brand
SELECT * FROM public."brand"
Brand table:
id: 2   type_id: 1      name: brand_2   country: country_2

-------------------------------------
id: 3   type_id: 3      name: brand_3   country: country_3

-------------------------------------
id: 4   type_id: 1      name: brand_4   country: RU

-------------------------------------
id: 1   type_id: 2      name: apple     country: USA

-------------------------------------
id: 5   type_id: 2      name: qptjfxwmdpvigqylf         country: rupqvjvefqyoninmp

-------------------------------------
id: 6   type_id: 3      name: ycwkgfdhbdrfeiqvjuwvy     country: lxssaouxwfkeicbdynpx

-------------------------------------
id: 7   type_id: 4      name: brand_7   country: UA

-------------------------------------
```

Запит:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py generate_randomly brand 5
insert into public."brand" select (SELECT MAX(id)+1 FROM public."brand"), (SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."type")-1)))), ar
ray_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                     FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), ''),array_to_string(ARRA
Y(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '');
insert into public."brand" select (SELECT MAX(id)+1 FROM public."brand"), (SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."type")-1)))), ar
ray_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                     FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), ''),array_to_string(ARRA
Y(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '');
insert into public."brand" select (SELECT MAX(id)+1 FROM public."brand"), (SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."type")-1)))), ar
ray_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                     FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), ''),array_to_string(ARRA
Y(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '');
insert into public."brand" select (SELECT MAX(id)+1 FROM public."brand"), (SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."type")-1)))), ar
ray_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                     FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), ''),array_to_string(ARRA
Y(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '');
insert into public."brand" select (SELECT MAX(id)+1 FROM public."brand"), (SELECT id FROM public."type" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."type")-1)))), ar
ray_to_string(ARRAY(SELECT chr((97 + round(random() * 25)) :: integer)                     FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), ''),array_to_string(ARRA
Y(SELECT chr((97 + round(random() * 25)) :: integer) FROM generate_series(1, FLOOR(RANDOM()*(25-10)+10):: integer)), '');
```

Модифікована таблиця brand:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py print_table brand
SELECT * FROM public."brand"
Brand table:
id: 2   type_id: 1      name: brand_2   country: country_2

-------------------------------------
id: 3   type_id: 3      name: brand_3   country: country_3

-------------------------------------
id: 4   type_id: 1      name: brand_4   country: RU

-------------------------------------
id: 1   type_id: 2      name: apple     country: USA

-------------------------------------
id: 5   type_id: 2      name: qptjfxwmdpvigqylf         country: rupqvjvefqyoninmp

-------------------------------------
id: 6   type_id: 3      name: ycwkgfdhbdrfeiqvjuwvy     country: lxssaouxwfkeicbdynpx

-------------------------------------
id: 7   type_id: 4      name: brand_7   country: UA

-------------------------------------
id: 8   type_id: 2      name: eeznxfiedbmmlcnqpkcglgdc  country: ympqtmhrlqqxzcktnzbdeem

-------------------------------------
id: 9   type_id: 8      name: zeebkmjslheieohfow        country: nocntskdmcxwoqklfi

-------------------------------------
id: 10  type_id: 7      name: aiilwpgfyprrww    country: tltluojmejz

-------------------------------------
id: 11  type_id: 7      name: qwbwgjfkwvipudaxmxegu     country: drdefmnswjnlhwumgnwmvfr

-------------------------------------
id: 12  type_id: 3      name: veznmsuyupltokgqvp        country: lmvqtjpplhttzaq

-------------------------------------
```

Завдання 3

Пошук за двома атрибутами з двох таблиць (user, basket)

Формування запиту:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py search_records user basket id id
specify the number of attributes you`d like to search by: 2
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.id
specify the left end of search interval: 0
specify the right end of search interval: 5
select * from public."user" as one inner join public."basket" as two on one."id"=two."id" where 0<one.id and one.id<5 and 0<two.id and two.id<5
--- 1.6392338275909424 seconds ---
```

Запит:

```
select * from public."user" as one inner join public."basket" as two on one."id"=two."id" where 0<one.id and one.id<5 and 0<two.user_id and two.user_id<5;
```

Результат:

```
search result:
1
Pavlo
pavlo@gmail.com
1
2
1
-------------------------------------
2
Feofan
Feofan@gmail.com
2
3
4
-------------------------------------
4
Oleg
oleg@email.com
4
3
2
-------------------------------------
```

Пошук за трьома атрибутами з трьох таблиць (basket, device, type)

Форматування запиту:

```
PS C:\Users\Bohdan\Desktop\3курс\5семестр\DB\Lab2> python main.py search_records basket device type id id id id
specify the number of attributes you`d like to search by: 3
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.device_id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.id
specify the left end of search interval: 0
specify the right end of search interval: 5
specify the type of data you want to search for (numeric, string or date): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: three.id
specify the left end of search interval: 0
specify the right end of search interval: 5
select * from public."basket" as one inner join public."device" as two on one."id"=two."id" inner join public."type" as three on three."id"=one."id"where 0<one.device_id and one.device
_id<5 and 0<two.id and two.id<5 and 0<three.id and three.id<5
--- 0.0079782009124755866 seconds ---
```

Запит:

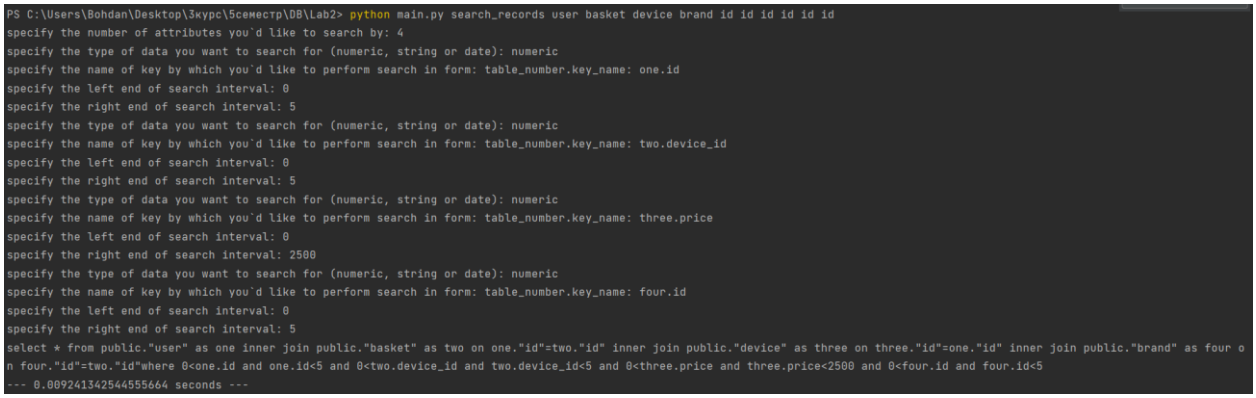
```
select * from public."basket" as one inner join public."device" as two on one."id"=two."id" inner join public."type" as three on three."id"=one."id"where 0<one.device_id and one.device_id<5 and 0<two.id and two.id<5 and 0<three.id and three.id<5;
```

Результат:

```
search result:
1
2
1
1
500
1
1
device_1
1
type_1
-------------------------------------
2
3
4
2
7000
2
3
device_2
2
type_2
-------------------------------------
3
1
5
3
2099
3
4
device_6
3
play
```

Пошук за чотирьма атрибутами з чотирьох таблиць (user, basket, device, brand)

Форування запиту:



Запит:

```
select * from public."user" as one inner join public."basket" as two on one."id"=two."id" inner join public."device" as three on three."id"=one."id" inner join public."brand" as four on four."id"=two."id"where 0<one.id and one.id<5 and 0<two.device_id and two.device_id<5 and 0<three.price and three.price<2500 and 0<four.id and four.id<5;
```

Результат:

Завдання 4

Програмний код модулю “model.py”:

```
import datetime
from typing import Union, List, Tuple, Any

import psycopg2 as ps

class Model:
    def __init__(self):
        self.conn = None
        try:
            self.conn = ps.connect(
                database="Lab1",
                user='postgres',
                password="qwerty123",
                host='127.0.0.1',
                port="5432",
            )
        except(Exception, ps.DatabaseError) as error:
            print("[INFO] Error while working with Postgresql", error)

    def request(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return True
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as
error:
```

```
            print(error)
            self.conn.rollback()
            return False

    def get(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return cursor.fetchall()
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as
error:
            print(error)
            self.conn.rollback()
            return False

    def get_el(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return cursor.fetchone()
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as
error:
            print(error)
            self.conn.rollback()
            return False

    def count(self, table_name: str):
```

```
        return self.get_el(f"select count(*) from
public.\"{table_name}\"")


    def find(self, table_name: str, key_name: str, key_value: int):
        return self.get_el(f"select count(*) from
public.\"{table_name}\" where {key_name}={key_value}")


    def max(self, table_name: str, key_name: str):
        return self.get_el(f"select max({key_name}) from
public.\"{table_name}\"")


    def min(self, table_name: str, key_name: str):
        return self.get_el(f"select min({key_name}) from
public.\"{table_name}\"")


    def print_users(self) -> None:
        return self.get(f"SELECT * FROM public.\"user\"")


    def print_baskets(self) -> None:
        return self.get(f"SELECT * FROM public.\"basket\"")


    def print_devices(self) -> None:
        return self.get(f"SELECT * FROM public.\"device\"")


    def print_types(self) -> None:
        return self.get(f"SELECT * FROM public.\"type\"")


    def print_brands(self) -> None:
        return self.get(f"SELECT * FROM public.\"brand\"")


    def delete_data(self, table_name: str, key_name: str, key_value) -
> None:
```

```
        self.request(f"DELETE FROM public.\"{table_name}\" WHERE
{key_name}={key_value};")


    def update_data_basket(self, key_value: int, user_id: int,
device_id: int) -> None:

        self.request(f"UPDATE public.\"basket\" SET
user_id=\'{user_id}\', device_id={device_id} "

                     f"WHERE id={key_value};")


    def update_data_device(self, key_value: int, price: int, type_id:
int, brand_id: int, name: str) -> None:

        self.request(f"UPDATE public.\"device\" SET price=\'{price}\',
type_id={type_id}, "

                     f"brand_id=\'{brand_id}\',name =\'{name}\' WHERE
id={key_value};")


    def update_data_type(self, key_value: int, name: str) -> None:

        self.request(f"UPDATE public.\"type\" SET name=\'{name}\'
WHERE id={key_value};")


    def update_data_brand(self, key_value: int, type_id: int, name:
str, country: str) -> None:

        self.request(

            f"UPDATE public.\"brand\" SET type_id = \'{type_id}\',
name=\'{name}\',country=\'{country}\' WHERE id={key_value};")


    def update_data_user(self, key_value: int, name: str, email: str)
-> None:

        self.request(f"UPDATE public.\"user\" SET name=\'{name}\', "

                     f"email=\'{email}\' WHERE id={key_value};")


    def insert_data_basket(self, id: int, device_id: int, user_id:
int) -> None:

        self.request(f"insert into public.\"basket\"
(id,device_id,user_id) "
```

```
                    f"VALUES ({id}, \'{device_id}\',
\'{user_id}\');")


    def insert_data_device(self, id: int, price: int, type_id: int,
brand_id: int, name: str) -> None:
        self.request(f"insert into public.\"device\" (id, price,
type_id,brand_id,name ) "
                     f"VALUES ({id}, \'{price}\', {type_id},
\'{brand_id}\',\'{name}\');")


    def insert_data_type(self, id: int, name: str) -> None:
        self.request(f"insert into public.\"type\" (id, name) "
                     f"VALUES ({id}, \'{name}\');")


    def insert_data_brand(self, id: int, type_id: int, name: str,
country: str) -> None:
        self.request(f"insert into public.\"brand\" (id,type_id,
name,country) "
                     f"VALUES ({id}, \'{type_id}\', \'{name}\',
\'{country}\');")


    def insert_data_user(self, id: int, name: str, email: str) ->
None:
        self.request(f"insert into public.\"user\" (id,name,email) "
                     f"VALUES ({id}, \'{name}\', \'{email}\');")


    def user_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"user\""
                         "select (SELECT MAX(id)+1 FROM
public.\"user\"), "
                         "array_to_string(ARRAY(SELECT chr((97 +
round(random() * 25)) :: integer) "
                         "FROM generate_series(1, FLOOR(RANDOM()*(25-
10)+10):: integer)), ''), "
```

```
                          "array_to_string(ARRAY(SELECT chr((97 +
round(random() * 25)) :: integer) "
                          "FROM generate_series(1, FLOOR(RANDOM()*(10-
4)+4):: integer)), ''); ")


    def basket_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"basket\" select (SELECT
(MAX(id)+1) FROM public.\"basket\"), "
                          "(SELECT id FROM public.\"device\" LIMIT 1
OFFSET (round(random() * "
                          "((SELECT COUNT(id) FROM public.\"device\")-
1)))),"
                          "(SELECT id FROM public.\"user\" LIMIT 1
OFFSET "
                          "(round(random() * ((SELECT COUNT(id) FROM
public.\"user\")-1))));")


    def device_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"device\" select (SELECT
MAX(id)+1 FROM public.\"device\"), "
                          "FLOOR(RANDOM()*(100000-1)+1),"
                          "(SELECT id FROM public.\"type\" LIMIT 1
OFFSET "
                          "(round(random() *((SELECT COUNT(id) FROM
public.\"type\")-1))), "
                          "(SELECT id FROM public.\"brand\" LIMIT 1
OFFSET "
                          "(round(random() * ((SELECT COUNT(id) FROM
public.\"brand\")-1)))),"
                          "array_to_string(ARRAY(SELECT chr((97 +
round(random() * 25)) :: integer) "
                          "FROM generate_series(1, FLOOR(RANDOM()*(25-
10)+10):: integer)), '')  ;")
```

```
    def type_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"type\" select (SELECT
MAX(id)+1 FROM public.\"type\"), "
                         "array_to_string(ARRAY(SELECT chr((97 +
round(random() * 25)) :: integer) "
                         "FROM generate_series(1, FLOOR(RANDOM()*(25-
10)+10):: integer)), ''); ")


    def brand_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"brand\" select (SELECT
MAX(id)+1 FROM public.\"brand\"), "
                         "(SELECT id FROM public.\"type\" LIMIT 1
OFFSET "
                         "(round(random() *((SELECT COUNT(id) FROM
public.\"type\")-1)))), "
                         "array_to_string(ARRAY(SELECT chr((97 +
round(random() * 25)) :: integer)\
                          FROM generate_series(1, FLOOR(RANDOM()*(25-
10)+10):: integer)), ''),"
                         "array_to_string(ARRAY(SELECT chr((97 +
round(random() * 25)) :: integer) "
                         "FROM generate_series(1, FLOOR(RANDOM()*(25-
10)+10):: integer)), ''); ")


    def search_data_two_tables(self, table1_name: str, table2_name:
str, table1_key, table2_key,
                               search: str):
        return self.get(f"select * from public.\"{table1_name}\" as
one inner join public.\"{table2_name}\" as two "
                        f"on one.\"{table1_key}\"=two.\"{table2_key}\"
"
                        f"where {search}")
```

```
    def search_data_three_tables(self, table1_name: str, table2_name:
str, table3_name: str,
                                 table1_key, table2_key, table3_key,
table13_key,
                                 search: str):
        return self.get(f"select * from public.\"{table1_name}\" as
one inner join public.\"{table2_name}\" as two "
                        f"on one.\"{table1_key}\"=two.\"{table2_key}\"
inner join public.\"{table3_name}\" as three "
                        f"on
three.\"{table3_key}\"=one.\"{table13_key}\""
                        f"where {search}")


    def search_data_all_tables(self, table1_name: str, table2_name:
str, table3_name: str, table4_name: str,
                               table1_key, table2_key, table3_key,
table13_key,
                               table4_key, table24_key,
                               search: str):
        return self.get(f"select * from public.\"{table1_name}\" as
one inner join public.\"{table2_name}\" as two "
                        f"on one.\"{table1_key}\"=two.\"{table2_key}\"
inner join public.\"{table3_name}\" as three "
                        f"on
three.\"{table3_key}\"=one.\"{table13_key}\" inner join
public.\"{table4_name}\" as four "
                        f"on
four.\"{table4_key}\"=two.\"{table24_key}\""
                        f"where {search}")
```

Опис функцій модуля:

Модуль «model.py» слугує точкою доступу до бази даних. Для реалізації запитів користувача до бази даних використовується бібліотека psycopg2.

У модулі використані такі функції:

1. request, get, get_el – здійстнюють запити до бази даних. При правильному запиті request повертає True, get повертає усі дані що було взято з запитів SELECT (масив кортежів з записами таблиць), get_el повертає тільки перший запис. У разі помилки вони повертають False;
2. max, min – повертають максимальне і мінімальне значення зазначеного ключа у таблиці;
3. count – повертає кількість усіх записів у таблиці;
4. find – повертає кількість записів таблиці, що відповідають заданій користувачем умові;
5. print_users – отримання з бази даних та виведення у консоль користувача таблиці “user”;
6. print_baskets – отримання з бази даних та виведення у консоль користувача таблиці “basket”;
7. print_devices – отримання з бази даних та виведення у консоль користувача таблиці “device”;
8. print_types – отримання з бази даних та виведення у консоль користувача таблиці “type”;
9. print_brands – отримання з бази даних та виведення у консоль користувача таблиці “brands”;
10. delete_data – реалізує видалення запису з обраної користувачем таблиці;
11. update_data_(назва таблиці) – реалізує запит на зміну даних у обраній користувачем таблиці;
12. insert_data_(назва таблиці) – реалізує запит на вставку запису до обраної користувачем таблиці;
13. (назва таблиці)_data_generator – реалізує запит на вставку рандомізованих записів до обраної користувачем таблиці;
14. search_data_(кількість таблиць)_tables – реалізує пошук даних у вибраній користувачем кількості таблиць;