

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

Розрахунково-графічна робота
з дисципліни: «Бази даних і засоби управління»

Тема: «Створення додатку бази даних, орієнтованого на взаємодію з
СУБД PostgreSQL»

Виконала: студентка III курсу
ФПМ групи КВ-13
Щербина Н. І.
[Телеграм](#)
Перевірів: Петрашенко А.В.

Постановка задачі

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).

Додаткова інформація

URL репозиторію з вихідним кодом та звітом - [github](#).

Для написання програми використано мову програмування: Python 3.9.13, для реалізації взаємодії з базою даних – PostgreSQL, а для вимірювання часу при запиті пошуку – time.

Шаблон проектування: MVC.

Інформація про предметну галузь з лр№1

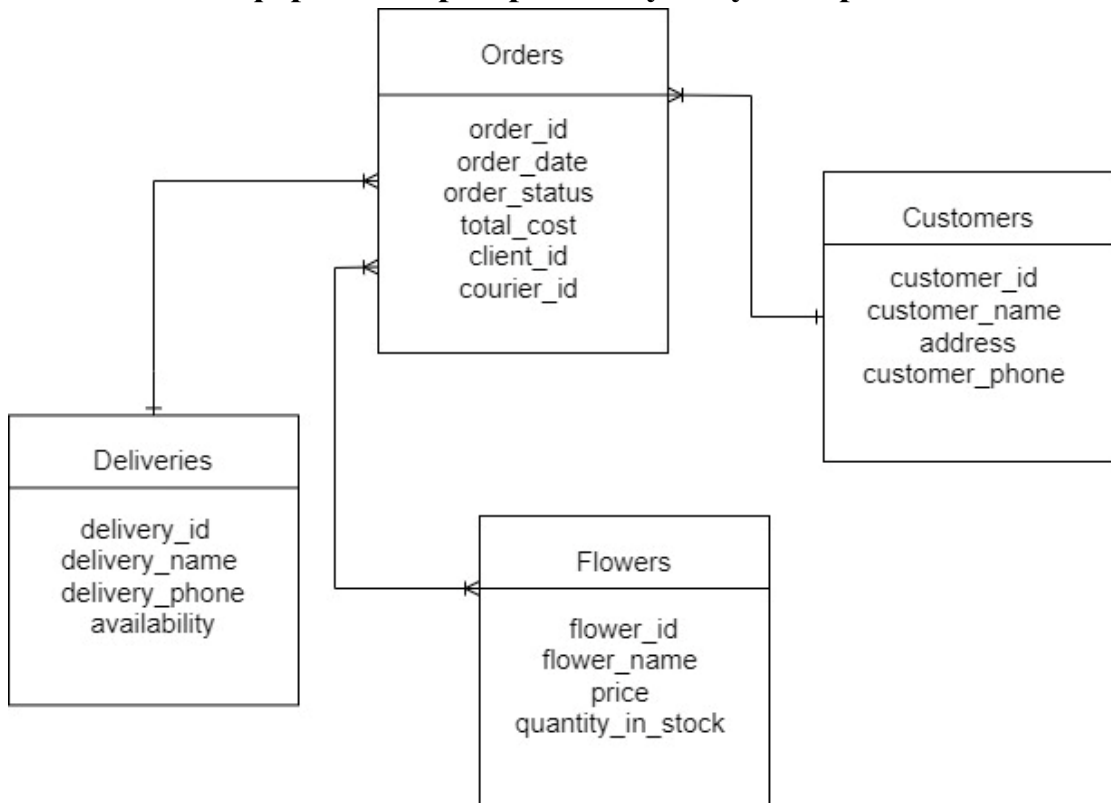


Рисунок 1 – Концептуальна модель предметної області «Система управління замовленнями та доставкою квітів».

Нотація: «Пташина лапка». Модель побудована засобами програми draw.io

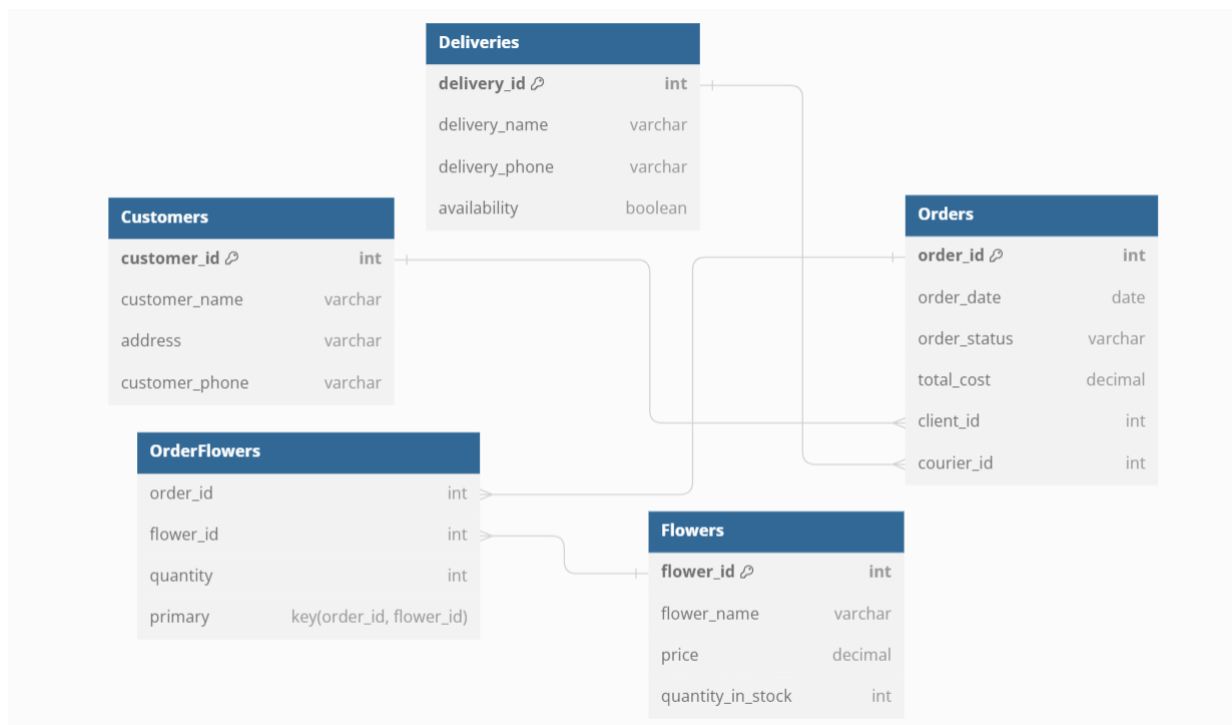


Рисунок 2 – Логічна модель БД « Система управління замовленнями та доставкою квітів »

Нотація: Модель побудована засобами dbdiagram.io.

Опис предметної галузі

Під час оформуванні обраної предметної галузі «Система управління замовленнями та доставкою квітів» було сформовано наступні сутності:

- Клієнт (Customers):

Призначення: Представляє інформацію про клієнтів, які роблять замовлення квітів.

Атрибути: customer_id (ідентифікатор клієнта), customer_name (ім'я клієнта), address (адреса клієнта), customer_phone (номер телефону клієнта).

- Замовлення (Orders):

Призначення: Зберігає дані про замовлення квітів, включаючи інформацію про клієнта, статус замовлення та загальну вартість.

Атрибути: order_id (ідентифікатор замовлення), order_date (дата замовлення), order_status (статус замовлення), total_cost (загальна вартість замовлення), client_id (ідентифікатор клієнта), courier_id (ідентифікатор кур'єра).

- Кур'єр (Deliveries):

Призначення: Представляє інформацію про кур'єрів, які доставляють замовлення квітів.

Атрибути: delivery_id (ідентифікатор кур'єра), delivery_name (ім'я кур'єра), delivery_phone (номер телефону кур'єра), availability

- Квіти (Flowers):

Призначення: Містить інформацію про різні види квітів, які доступні для замовлення.

Атрибути: flower_id (ідентифікатор квіту), flower_name (назва квіту), price (ціна квіту), quantity_in_stock (кількість квітів на складі).

Зв'язки:

Клієнт може розміщувати багато замовлень, але кожне замовлення належить лише одному клієнту. (1:N - Клієнт і Замовлення)

Замовлення може містити багато продуктів, і це працює в іншу сторону. (N:M - Замовлення і Квіти)

Кожне замовлення обслуговується одним кур'єром, але кур'єр може доставляти багато замовлень. (1:N - Замовлення і Кур'єр)

Структура програми та її опис

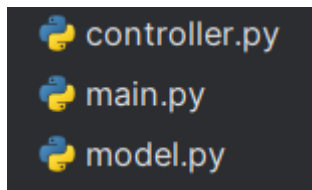


Рисунок 3 – структура програми

Структура проекту, що використовує шаблон проектування MVC (Model-View-Controller), включає наступні модулі:

1. **model.py**: Він взаємодіє з базою даних, виконуючи операції як SELECT, INSERT, DELETE, UPDATE, а також складніші операції. Цей модуль відповідає за роботу безпосередньо з даними.
2. **controller.py**: Тут розташовані основне та допоміжне меню для зручного управління базою даних. Він відповідає за логіку програми та координує взаємодію між моделлю та представленням.
3. **main.py**: Це головний файл, точка входу в програму, де відбувається підключення до бази даних та ініціалізація необхідних компонентів для початку роботи програми.

Такий підхід дозволяє розділити логіку програми на окремі компоненти, що сприяє більшій читабельності та розділення на частини коду.

Спеціальні методи виконання завдань у лабораторній роботі

1. **Вставка даних у таблицю (INSERT data in table)**: ця функція викликає операцію вставки інформації у базу даних.
2. **Редагування даних у таблиці (EDIT data in table)**: цей метод дозволяє змінювати вже існуючі дані у відповідній таблиці.
3. **Видалення даних з таблиці (DELETE data from table)**: вона викликає функцію для видалення конкретних даних із таблиці бази даних.
4. **Виведення рядків (PRINT rows)**: ця операція активує функцію виведення інформації з таблиці бази даних.

Додатковий метод, який реалізовано:

5. **Генерація випадкових даних (GENERATE random data)**: ця функція дає можливість користувачеві заповнювати таблицю або всю базу даних випадково згенерованими даними.

Кожна з цих функцій викликає допоміжну процедуру **select_table()**, що дозволяє користувачеві вибрати таблицю.

Також, існує метод для завдання 3:

6. **Пошук даних у таблицях (SEARCH data from tables):** він викликає функцію для пошуку інформації в таблицях за атрибутами та здійснення з'єднання таблиць за ключем.

Демонстрація роботи

- Початок роботи

```
Successfully CONNECTED to database Flowers

1. INSERT data in table
2. EDIT data in table
3. DELETE data from table
4. PRINT rows
5. GENERATE random data
6. SEARCH data from tables
0. Exit

Choose an option 1-6 or 0: 4
```

Рисунок 4 – старт програми

Завдання №1

- *Запит на вставку*

Лістинги

```
def insert(choice: int, data: list) -> bool:
    if connection is None or cursor is None:
        return False
    else:
        try:
            if choice == 1:
                cursor.execute(f"""INSERT INTO public.\"deliveries\"
(delivery_name, delivery_phone, availability) \
VALUES (\'{data[0]}\' \', \'{data[1]}\' \',
{data[2]});""")
            elif choice == 2:
                cursor.execute(f"""INSERT INTO public.\"orders\" (order_date,
order_status, total_cost, client_id, courier_id) \
VALUES (\'{data[0]}\' \', \'{data[1]}\' \',
{data[2]}, {data[3]}, {data[4]};""")
            elif choice == 3:
                cursor.execute(f"""INSERT INTO public.\"flowers\"
(flower_name, price, quantity_in_stock) \
VALUES (\'{data[0]}\' \', \'{data[1]}\' \',
\'>{data[2]}\');""")
            elif choice == 4:
                cursor.execute(f"""INSERT INTO public.\"orderflowers\"
(flowers_id, quantity) \
```

```

VALUES (\'{data[0]}\', {data[1]});"""
elif choice == 5:
    cursor.execute(f"""INSERT INTO
public.\"customers\"(customer_name, address, customer_phone) \
VALUES (\'{data[0]}\', \'{data[1]}\',
\>{data[2]}\'');""")
    connection.commit()
except Exception as _ex:
    print("Impossible to INSERT data into table", _ex)
    return False
return True

```

Демонстрація знімків екрану

deliveries_id	delivery_name	delivery_phone	availability
=====	=====	=====	=====
1	Василь	111-222-3333	True
2	Степан	444-555-6666	False
5	Nadyryshka	090	False
7	aaa	aaa	True
8	a	a	True
9	a	a	True
10	934dc03a7	64b3858f0	False
11	82fad4739	9c53abbfc	False
12	43e89fddb	fe48ec751	False
13	58f40a48d	e9372b6ab	True
14	9b7ee1d1c	8cc40195f	True
15	Stas	0671234567	True

Рисунок 5 – початкова таблиця deliveries

```

Input data separated by comma
Table: deliveries. Input: delivery_name->text, delivery_phone->text, availability->boolean
Max, 0123456789, false
Data INSERTED successfully

```

Рисунок 6 – процес занесення даних

deliveries_id	delivery_name	delivery_phone	availability
=====	=====	=====	=====
1	Василь	111-222-3333	True
2	Степан	444-555-6666	False
5	Nadyryshka	090	False
7	aaa	aaa	True
8	a	a	True
9	a	a	True
10	934dc03a7	64b3858f0	False
11	82fad4739	9c53abbfc	False
12	43e89fddb	fe48ec751	False
13	58f40a48d	e9372b6ab	True
14	9b7ee1d1c	8cc40195f	True
15	Stas	0671234567	True
16	Max	0123456789	False

Рисунок 7 – таблиця deliveries після вставки

customers_id	customer_name	address	customer_phone
=====	=====	=====	=====
1	Олександр	вул. Квіткова 65	123-456-7890
2	Валерія	вул. Травнева 42/5	987-654-3210

Input data separated by comma
Table: customers. Input: customer_name->text, address->text, customer_phone->text
Олександр, вул. Сумська 25, 0123456789
Data INSERTED successfully

customers_id	customer_name	address	customer_phone
=====	=====	=====	=====
1	Олександр	вул. Квіткова 65	123-456-7890
2	Валерія	вул. Травнева 42/5	987-654-3210
3	Олександр	вул. Сумська 25	0123456789

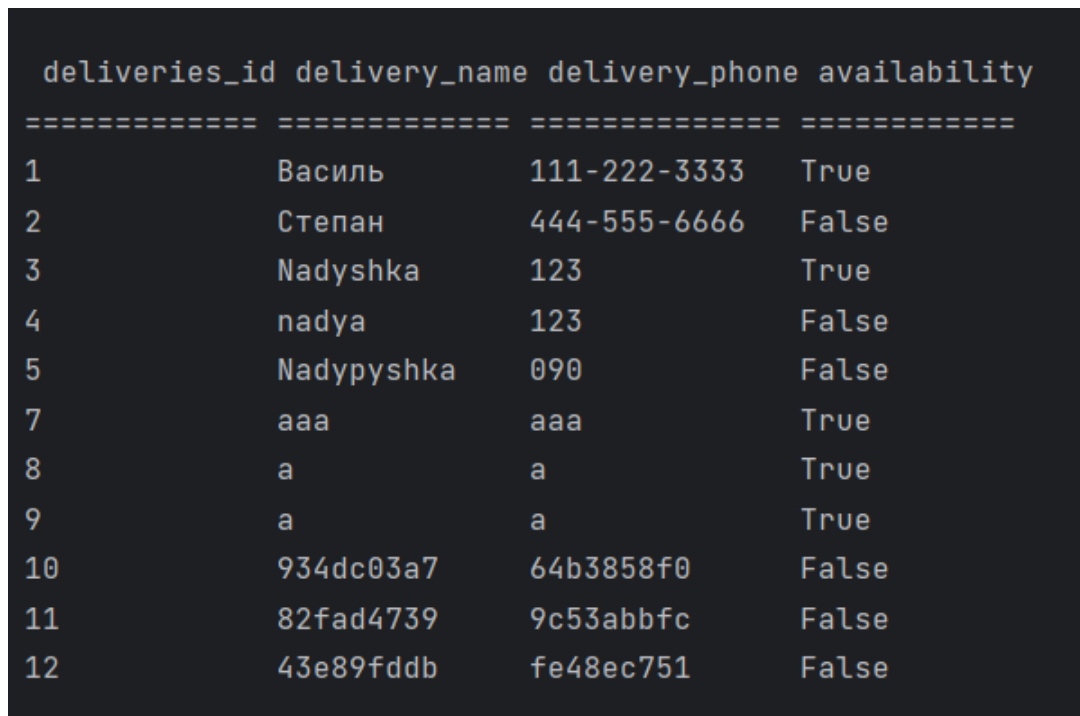
Рисунок 8 – таблиця customers до та після вставки

- *Запит на видалення*

Лістинги

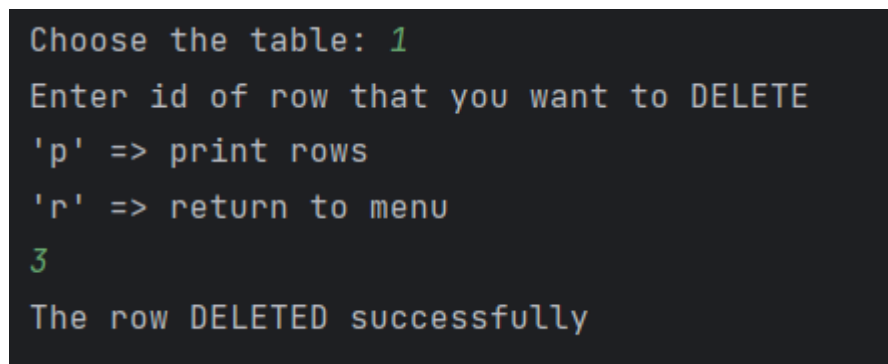
```
def delete(table: str, key_name: str, key_val: str) -> bool:
    if connection is None or cursor is None:
        return False
    else:
        try:
            cursor.execute(f"""DELETE FROM public.\"{table}\" WHERE
{key_name} = \"{key_val}\";""")
            connection.commit()
        except Exception as _ex:
            print(f"Impossible to DELETE data from table {table}", _ex)
            return False
    return True
```

Демонстрація знімків екрану



deliveries_id	delivery_name	delivery_phone	availability
1	Василь	111-222-3333	True
2	Степан	444-555-6666	False
3	Nadyshka	123	True
4	nadya	123	False
5	Nadypyshka	090	False
7	aaa	aaa	True
8	a	a	True
9	a	a	True
10	934dc03a7	64b3858f0	False
11	82fad4739	9c53abbfc	False
12	43e89fddb	fe48ec751	False

Рисунок 9 – початкова таблиця deliveries



```
Choose the table: 1
Enter id of row that you want to DELETE
'p' => print rows
'r' => return to menu
3
The row DELETED successfully
```

Рисунок 10 – процес запиту видалення

deliveries_id	delivery_name	delivery_phone	availability
=====	=====	=====	=====
1	Василь	111-222-3333	True
2	Степан	444-555-6666	False
5	Nadyryshka	090	False
7	aaa	aaa	True
8	a	a	True
9	a	a	True
10	934dc03a7	64b3858f0	False
11	82fad4739	9c53abbfc	False
12	43e89fddb	fe48ec751	False

Рисунок 11 – результат видалення рядку у deliveries (з індексом 3 не існує)

orders_id	order_date	order_status	order_cost	client_id	courier_id
=====	=====	=====	=====	=====	=====
1	2023-09-12	В обробці	50.50	1	1
2	2023-09-13	Доставлено	75.25	2	2
3	2023-11-30	c3d772ea054	5058	1	5
4	2023-11-17	7353634b555	5795	1	9
5	2023-11-29	7771f0bf823	5137	2	11
6	2023-10-19	2807946bf48	7487	2	12

```

Choose the table: 2
Enter id of row that you want to DELETE
'p' => print rows
'r' => return to menu
6
The row DELETED successfully

```

orders_id	order_date	order_status	order_cost	client_id	courier_id
=====	=====	=====	=====	=====	=====
1	2023-09-12	В обробці	50.50	1	1
2	2023-09-13	Доставлено	75.25	2	2
3	2023-11-30	c3d772ea054	5058	1	5
4	2023-11-17	7353634b555	5795	1	9
5	2023-11-29	7771f0bf823	5137	2	11

Рисунок 12 – видалення рядку у orders (з індексом 6 не існує після маніпуляцій)

- *Запит на редагування*

Лістинги

```
def update(choice: int, data: list, id1: int, id2: int = 0) -> bool:
    if connection is None or cursor is None:
        return False
    else:
        try:
            if choice == 1:
                cursor.execute(f"""UPDATE public.\"deliveries\" SET
delivery_name = \"{data[0]}\", \
                delivery_phone = \"{data[1]}\", availability = {data[2]}
WHERE deliveries_id = {id1};""")
            elif choice == 2:
                cursor.execute(f"""UPDATE public.\"orders\" SET order_date =
 \"{data[0]}\", order_status = \
                \"{data[1]}\", total_cost = {data[2]}, client_id =
{data[3]}, courier_id = {data[4]} WHERE orders_id = {id1};""")
            elif choice == 3:
                cursor.execute(f"""UPDATE public.\"flowers\" SET flower_name
= \"{data[0]}\", price = \"{data[1]}\", \
                quantity_in_stock = \"{data[2]}\" WHERE flowers_id =
{id1};""")
            elif choice == 4:
                cursor.execute(f"""UPDATE public.\"orderflowers\" SET
flowers_id = \"{data[0]}\", quantity = {data[1]}, \
                WHERE orderflowers_id = {id1};""")
            elif choice == 5:
                cursor.execute(f"""UPDATE public.\"customers\" SET
customer_name = \"{data[0]}\", address = \
                \"{data[1]}\", customer_phone = {data[2]} WHERE
customers_id = {id1};""")
            return True
        except Exception as _ex:
            print("Impossible to UPDATE data into table", _ex)
            return False
```

Демонстрація знімків екрану

```

Enter id of row that you want to UPDATE
'p' => print rows
'r' => return to menu
p
flowers_id flower_name price    quantity_in_stock
=====
1          Тюльпани    10.99    100
2          Рози       12.99    75

Enter id of row that you want to UPDATE
'n' => next 15 rows
'b' => previous 15 rows
'r' => return to menu
2
If you don't want to UPDATE column -> write as it was
Input data separated by comma
Table: flowers. Input: flower_name->text, price->numeric, quantity_in_stock
Хризантеми, 13.99, 40
UPDATED successfully

```

Рисунок 13 – процес редагування рядку у flowers (з індексом 2)

flowers_id	flower_name	price	quantity_in_stock
1	Тюльпани	10.99	100
2	Хризантеми	13.99	40

Рисунок 15 – результат зміни рядка у flowers з індексом 2

```
Enter id of row that you want to UPDATE
'p' => print rows
'r' => return to menu
p
customers_id customer_name address customer_phone
=====
1 Олександр вул. Квіткова 65 123-456-7890
2 Валерія вул. Травнева 42/5 987-654-3210
3 Олександр вул. Сумська 25 0123456789
4 Катерина вул. Ковалева 6 0987654321

Enter id of row that you want to UPDATE
'n' => next 15 rows
'b' => previous 15 rows
'r' => return to menu
3
If you don't want to UPDATE column -> write as it was
Input data separated by comma
Table: customers. Input: customer_name->text, address->text, customer_phone->text
Микола, вул. Кременчуцька 25, 0999999999
UPDATED successfully
```

customers_id	customer_name	address	customer_phone
1	Олександр	вул. Квіткова 65	123-456-7890
2	Валерія	вул. Травнева 42/5	987-654-3210
3	Микола	вул. Кременчуцька 25	999999999
4	Катерина	вул. Ковалева 6	0987654321

Рисунок 16 – редагування рядка у customers з індексом 3

Завдання №2

- Запит на генерацію даних

Лістинги

```
def generate(choice: int, count: int) -> bool:
    if connection is None or cursor is None:
        return False
    try:
        for i in range(count):
            if choice == 1:
                cursor.execute(f"""INSERT INTO public.\"deliveries\"
(delivery_name, delivery_phone, availability) \
VALUES (substr(md5(random()::text), 0, 10),
substr(md5(random()::text), 0, 10), \
(round(random()::int)::boolean));""")
            elif choice == 2:
                cursor.execute(f"""INSERT INTO public.\"orders\" (order_date,
order_status, total_cost, client_id, courier_id) \
SELECT NOW() + (random() * (NOW() - NOW() -
```

```

'360 days')), \
                                (substr(md5(random())::character
varying(12)), 0, 12)), \
                                (floor(random() * (25000 - 5000 + 1)) +
5000), \
                                customers_id, deliveries_id FROM
public."customers", public."deliveries" \
                                order by random() limit 1;""")
        elif choice == 3:
            cursor.execute(f"""INSERT INTO public.\"flowers\"
(fl原因ower_name, price, quantity_in_stock) \
                                SELECT substr(md5(random())::text), 0, 10), \
                                (floor(random() * (25000 - 5000 + 1)) +
5000), \
                                floor(random() * (1000 - 10 + 1) +
10);""")
        elif choice == 4:
            cursor.execute(f"""
                                INSERT INTO public.\"orderflowers\" (flowers_id,
quantity) \
                                SELECT (flowers_id FROM public."flowers" order by
random() limit 1, \
                                ( floor(random() * (25000 - 5000 + 1)) + 5000);
                                """)
        elif choice == 5:
            cursor.execute(f"""INSERT INTO public.\"customers\"
(customer_name, address, customer_phone) \
                                SELECT substr(md5(random())::text), 0, 10), \
                                substr(md5(random())::text), 0, 10), \
                                substr(md5(random())::text), 0, 10);""")

        connection.commit()
    except Exception as _ex:
        print("Impossible to GENERATE data to database FLOWERS", _ex)
        return False
    return True

```

Демонстрація знімків екрану

В запиті виконується вставка даних у таблицю на основі випадкових значень. Для таблиць внесено по одному рядку згенерованих даних.

```

Input the data quantity to GENERATE: 1
Data GENERATED and INSERTED into table number 1 successfully

```

Рисунок 17 – процес генерації

deliveries_id	delivery_name	delivery_phone	availability
=====	=====	=====	=====
1	Василь	111-222-3333	True
2	Степан	444-555-6666	False
5	Nadyryshka	090	False
7	aaa	aaa	True
8	a	a	True
9	a	a	True
10	934dc03a7	64b3858f0	False
11	82fad4739	9c53abbfc	False
12	43e89fddb	fe48ec751	False
13	58f40a48d	e9372b6ab	True
14	9b7ee1d1c	8cc40195f	True
15	Stas	0671234567	True
16	Max	0123456789	False
17	5c35cf9cc	6832fb1ab	True

deliveries_id	delivery_name	delivery_phone	availability
=====	=====	=====	=====
1	Василь	111-222-3333	True
2	Степан	444-555-6666	False
5	Nadyryshka	090	False
7	aaa	aaa	True
8	a	a	True
9	a	a	True
10	934dc03a7	64b3858f0	False
11	82fad4739	9c53abbfc	False
12	43e89fddb	fe48ec751	False
13	58f40a48d	e9372b6ab	True
14	9b7ee1d1c	8cc40195f	True
15	Stas	0671234567	True
16	Max	0123456789	False
17	5c35cf9cc	6832fb1ab	True
18	6ee3206cc	8d7c4d16e	True

Рисунок 18 – таблица Deliveries

orders_id	order_date	order_status	order_cost	client_id	courier_id
=====	=====	=====	=====	=====	=====
1	2023-09-12	В обробці	50.50	1	1
2	2023-09-13	Доставлено	75.25	2	2
3	2023-11-30	c3d772ea054	5058	1	5
4	2023-11-17	7353634b555	5795	1	9
5	2023-11-29	7771f0bf823	5137	2	11
7	2023-12-03	ab6864d7b40	5046	4	13

orders_id	order_date	order_status	order_cost	client_id	courier_id
=====	=====	=====	=====	=====	=====
1	2023-09-12	В обробці	50.50	1	1
2	2023-09-13	Доставлено	75.25	2	2
3	2023-11-30	c3d772ea054	5058	1	5
4	2023-11-17	7353634b555	5795	1	9
5	2023-11-29	7771f0bf823	5137	2	11
7	2023-12-03	ab6864d7b40	5046	4	13
8	2023-11-19	41908fe62ef	5836	4	7

Рисунок 19 – таблиця Orders

flowers_id	flower_name	price	quantity_in_stock
=====	=====	=====	=====
1	Тюльпани	10.99	100
2	Рожі	12.99	75
8	79260aa1f	15855	69

flowers_id	flower_name	price	quantity_in_stock
=====	=====	=====	=====
1	Тюльпани	10.99	100
2	Рожі	12.99	75
8	79260aa1f	15855	69
9	04f709d82	7081	958

Рисунок 20 – таблиця Flowers

customers_id	customer_name	address	customer_phone
=====	=====	=====	=====
1	Олександр	вул. Квіткова 65	123-456-7890
2	Валерія	вул. Травнева 42/5	987-654-3210
3	Олександр	вул. Сумська 25	0123456789
4	Катерина	вул. Ковалева 6	0987654321

customers_id	customer_name	address	customer_phone
=====	=====	=====	=====
1	Олександр	вул. Квіткова 65	123-456-7890
2	Валерія	вул. Травнева 42/5	987-654-3210
3	Олександр	вул. Сумська 25	0123456789
4	Катерина	вул. Ковалева 6	0987654321
5	7ас5847с7	Ь4439550f	194be128b

Рисунок 21 – таблиця Customers

Завдання №3

- *Запит на пошук даних*

Лістинги

```
def search(tables: list[str], key: str, value: str) -> tuple:
    if connection is None or cursor is None:
        return ()
    try:
        request = f"""SELECT * FROM public.\"{tables[0]}\" as first INNER
JOIN public.\"{tables[1]}\" as second on first.\"{key}\" = second.\"{key}\"
WHERE {value}"""
        print(f"SQL request: {request}")
        start_time = time.time_ns()
        cursor.execute(request)
        rows = cursor.fetchall()
        run_time = time.time_ns() - start_time
    except Exception as _ex:
        print("Impossible to SEARCH data in database FLOWERS", _ex)
        return ()
    return rows, run_time
```

Демонстрація знімків екрану

```
Choose an option 1-6 or 0: 6

Choose the first table
1. Deliveries
2. Orders
3. Flowers
4. OrderFlowers
5. Customers
0. menu

Choose the table: 3
Choose the second table
1. Deliveries
2. Orders
3. Flowers
4. OrderFlowers
5. Customers
0. menu
```

Рисунок 22 – виконується об'єднання двох таблиць та пошук у них за трьома атрибутами

```
Input the connecting key: flowers_id
Input the expression. Use "first" and "second" to address to the table attributes, if with string use like
first.quantity_in_stock > 35 and second.quantity > 5
SQL request: SELECT * FROM public."flowers" as first INNER JOIN public."orderflowers" as second on first."f

flowers_id flower_name price    quantity_in_stock orderflowers_id flowers_id quantity
=====
8          79260aa1f  15855      69                1                8         10
12         92f9eb3cb  22240     462                5                12         25

Time of the executing program: 4013.6 ms
```

Рисунок 23 – результат пошуку

Завдання №4

Код модулю “model.py”

```
import psycopg2
import time

cursor = None
connection = None

def connect():
    try:
        global cursor, connection
        connection = psycopg2.connect(
            host="localhost",
            user="postgres",
            password="19401944",
            database="postgres",
            port="5432"
        )

        cursor = connection.cursor()

        print("Successfully CONNECTED to database FLOWERS")

    except Exception as _ex:
        print("Failed CONNECTION to database FLOWERS", _ex)

def disconnect():
    try:
        cursor.close()
        connection.close()
        print("Successfully DISCONNECTED from database FLOWERS")
    except Exception as _ex:
        print("Impossible to DISCONNECT from database FLOWERS", _ex)

def insert(choice: int, data: list) -> bool:
    if connection is None or cursor is None:
        return False
    else:
        try:
            if choice == 1:
                cursor.execute(f"""INSERT INTO public.\"deliveries\"
(delivery_name, delivery_phone, availability) \
VALUES (\'{data[0]}\' , \'{data[1]}\' ,
{data[2]});""")
            elif choice == 2:
                cursor.execute(f"""INSERT INTO public.\"orders\" (order_date,
order_status, total_cost, client_id, courier_id) \
VALUES (\'{data[0]}\' , \'{data[1]}\' ,
{data[2]}, {data[3]}, {data[4]};""")
            elif choice == 3:
                cursor.execute(f"""INSERT INTO public.\"flowers\"
(flower_name, price, quantity_in_stock) \
VALUES (\'{data[0]}\' , \'{data[1]}\' ,
\{data[2]\};""")
            elif choice == 4:
                cursor.execute(f"""INSERT INTO public.\"orderflowers\"
(flowers_id, quantity) \
```

```

VALUES (\'{data[0]}\', {data[1]});"""
        elif choice == 5:
            cursor.execute(f"""INSERT INTO
public.\"customers\"(customer_name, address, customer_phone) \
VALUES (\'{data[0]}\', \'{data[1]}\',
\ '{data[2]}\');""")
            connection.commit()
        except Exception as _ex:
            print("Impossible to INSERT data into table", _ex)
            return False
        return True

def delete(table: str, key_name: str, key_val: str) -> bool:
    if connection is None or cursor is None:
        return False
    else:
        try:
            cursor.execute(f"""DELETE FROM public.\"{table}\" WHERE
{key_name} = \'{key_val}\';""")
            connection.commit()
        except Exception as _ex:
            print(f"Impossible to DELETE data from table {table}", _ex)
            return False
        return True

def select_by_key(table: str, key_name: str, key_val: str) -> list:
    if connection is None or cursor is None:
        return []
    else:
        try:
            cursor.execute(f"""SELECT * FROM public.\"{table}\" WHERE
{key_name} = \'{key_val}\';""")
        except Exception as _ex:
            print(f"Impossible to SELECT data from table {table} by key
{key_name}", _ex)
            return []
        return cursor.fetchall()

def select_by_table(table: str, quantity: str = '100', offset: str = '0') ->
list:
    if connection is None or cursor is None:
        return []
    else:
        try:
            if table == 'room/chambermaid' or table == 'room/guest':
                cursor.execute(f"""SELECT * FROM public.\"{table}\" ORDER BY
{"room_id"} \
                                ASC limit {quantity} offset {offset};""")
            else:
                cursor.execute(f"""SELECT * FROM public.\"{table}\" ORDER BY
{table + "_id"} \
                                ASC limit {quantity} offset {offset};""")
        except Exception as _ex:
            print(f"Impossible to SELECT data from table {table}", _ex)
            return []
        return cursor.fetchall()

def update(choice: int, data: list, id1: int, id2: int = 0) -> bool:
    if connection is None or cursor is None:
        return False

```

```

else:
    try:
        if choice == 1:
            cursor.execute(f"""UPDATE public.\"deliveries\" SET
delivery_name = \"'{data[0]}'\", \
delivery_phone = \"'{data[1]}'\", availability = {data[2]}
WHERE deliveries_id = {id1};""")
        elif choice == 2:
            cursor.execute(f"""UPDATE public.\"orders\" SET order_date =
\"'{data[0]}'\", order_status = \
\"'{data[1]}'\", total_cost = {data[2]}, client_id =
{data[3]}, courier_id = {data[4]} WHERE orders_id = {id1};""")
        elif choice == 3:
            cursor.execute(f"""UPDATE public.\"flowers\" SET flower_name
= \"'{data[0]}'\", price = \"'{data[1]}'\", \
quantity_in_stock = \"'{data[2]}'\", WHERE flowers_id =
{id1};""")
        elif choice == 4:
            cursor.execute(f"""UPDATE public.\"orderflowers\" SET
flowers_id = \"'{data[0]}'\", quantity = {data[1]}, \
WHERE orderflowers_id = {id1};""")
        elif choice == 5:
            cursor.execute(f"""UPDATE public.\"customers\" SET
customer_name = \"'{data[0]}'\", address = \
\"'{data[1]}'\", customer_phone = {data[2]} WHERE
customers_id = {id1};""")
        return True
    except Exception as _ex:
        print("Impossible to UPDATE data into table", _ex)
        return False

def generate(choice: int, count: int) -> bool:
    if connection is None or cursor is None:
        return False
    try:
        for i in range(count):
            if choice == 1:
                cursor.execute(f"""INSERT INTO public.\"deliveries\"
(delivery_name, delivery_phone, availability) \
VALUES (substr(md5(random()::text), 0, 10), \
substr(md5(random()::text), 0, 10), \
(round(random()::int)::boolean));""")
            elif choice == 2:
                cursor.execute(f"""INSERT INTO public.\"orders\" (order_date,
order_status, total_cost, client_id, courier_id) \
SELECT NOW() + (random() * (NOW() - NOW() -
'360 days')), \
(substr(md5(random()::character
varying(12)), 0, 12)), \
(floor(random() * (25000 - 5000 + 1)) +
5000), \
customers_id, deliveries_id FROM
public.\"customers\", public.\"deliveries\" \
order by random() limit 1;""")
            elif choice == 3:
                cursor.execute(f"""INSERT INTO public.\"flowers\"
(flower_name, price, quantity_in_stock) \
SELECT substr(md5(random()::text), 0, 10), \
(floor(random() * (25000 - 5000 + 1)) +
5000), \
floor(random() * (1000 - 10 + 1) +
10);""")
            elif choice == 4:

```

```

        cursor.execute(f"""
            INSERT INTO public.\"orderflowers\" (flowers_id,
quantity)
            SELECT (flowers_id FROM public.\"flowers\" order by
random() limit 1,
            ( floor(random() * (25000 - 5000 + 1)) + 5000);
        """)
    elif choice == 5:
        cursor.execute(f"""INSERT INTO public.\"customers\"
(customer_name, address, customer_phone) \
            SELECT substr(md5(random()::text), 0, 10), \
            substr(md5(random()::text), 0, 10), \
            substr(md5(random()::text), 0, 10);""")

    connection.commit()
except Exception as _ex:
    print("Impossible to GENERATE data to database FLOWers", _ex)
    return False
return True

def search(tables: list[str], key: str, value: str) -> tuple:
    if connection is None or cursor is None:
        return ()
    try:
        request = f"""SELECT * FROM public.\"{tables[0]}\" as first INNER
JOIN public.\"{tables[1]}\" as second on first.\"{key}\" = second.\"{key}\"
WHERE {value}"""
        print(f"SQL request: {request}")
        start_time = time.time_ns()
        cursor.execute(request)
        rows = cursor.fetchall()
        run_time = time.time_ns() - start_time
    except Exception as _ex:
        print("Impossible to SEARCH data in database FLOWers", _ex)
        return ()
    return rows, run_time

```

Це модуль, що дозволяє взаємодіяти з базою даних у програмі. Він використовує бібліотеку `psycopg2` для здійснення запитів. Містить функції для з'єднання та від'єднання від бази даних, вставки, видалення, вибірки, оновлення даних у таблицях, а також для генерації та пошуку інформації.

1. **connect()**: Ця функція намагається здійснити з'єднання з базою даних.
2. **disconnect()**: Відповідно, ця функція намагається відключитись від бази даних після завершення роботи.
3. **insert(choice, data)**: Вставляє дані у таблицю за номером **choice** зі списку **data**.
4. **delete(table, key_name, key_val)**: Намагається видалити рядок з таблиці **table**, де значення ключа **key_name** дорівнює **key_val**.
5. **select_by_key(table, key_name, key_val)**: Вибирає дані з таблиці **table**, де значення ключа **key_name** дорівнює **key_val**.

6. **select_by_table(table, quantity, offset)**: Витягує дані з таблиці **table** та сортує їх у порядку зростання за первинним ключем, можливо, з параметрами кількості **quantity** та зсувом **offset**.
7. **update(choice, data, id1, id2)**: Змінює дані в таблиці **choice** на дані зі списку **data** для рядка з первинним ключем, що дорівнює **id**, або, якщо первинний ключ складений, то з ключами **id1** та **id2**.
8. **generate(choice, count)**: Генерує дані і намагається вставити їх в таблицю **choice** **count** разів.
9. **search(tables, key, value)**: Пошук та об'єднання даних з таблиць, де значення **key** дорівнює **value**.

Кожна з цих функцій відповідає за певну операцію в базі даних та надає можливість взаємодіяти з даними відповідного типу.