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: "Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL"

Виконав:
студент III курсу
групи КВ-91
Селетков В. Р.
Перевірив:
Павловський В. І.

Київ – 2021

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).

Інформація про програму

Посилання на репозиторій у GitHub з вихідним кодом програми та прикладеним звітом: https://github.com/vladsel/database

Використана мова програмування: Python 3.10.

Використані бібліотеки: psycopg2 (для зв’язку з СУБД), time (для виміру часу запиту пошуку, що у 3 завданні)

Використаний шаблон проектування: MVC.

Відомості про предметну галузь з лабораторної роботи №1

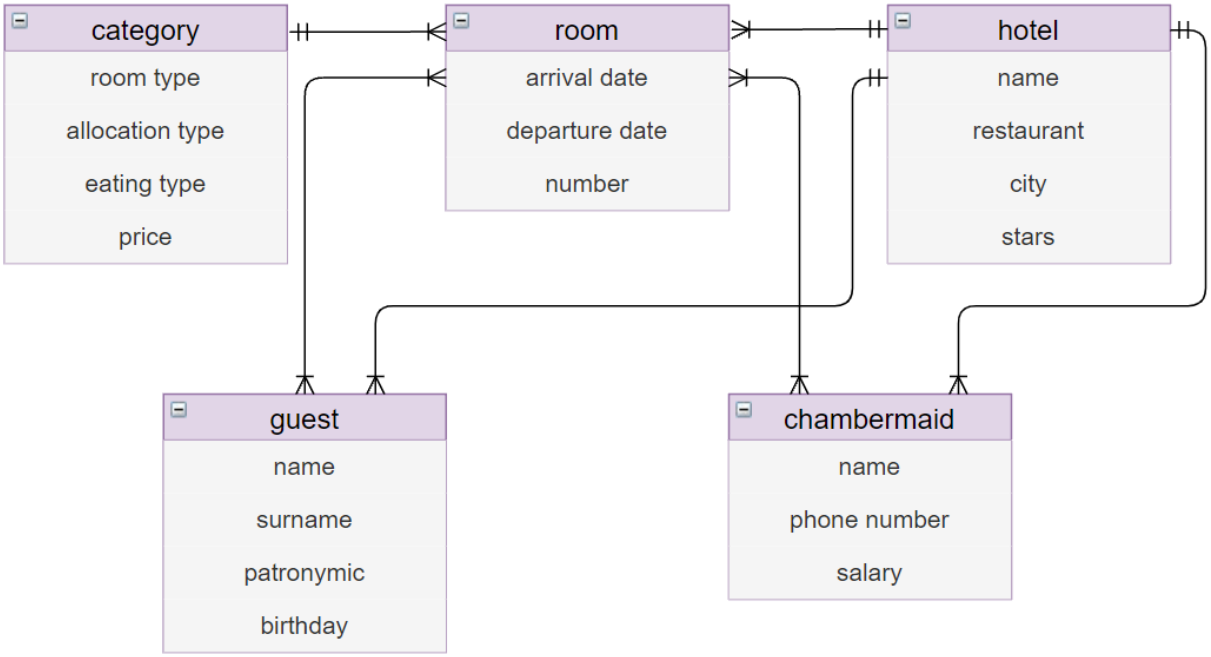

Рисунок 1 - ER-діаграма побудована за нотацією “Пташиної лапки (Crow’s foot)”, задана ER-діаграма була побудована у додатку [draw.io]

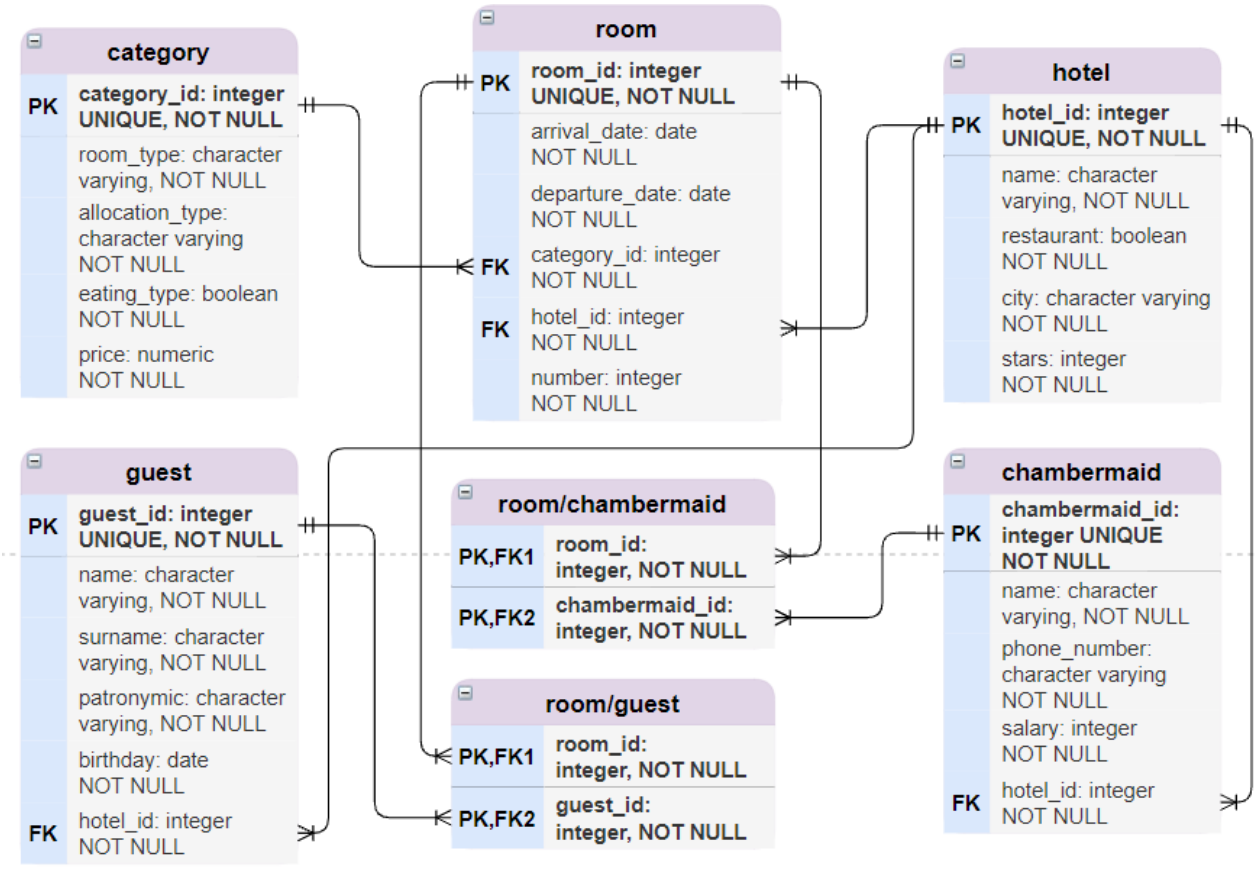

Рисунок 2 - Схема бази даних, побудовано у додатку [draw.io]

Таблиця 1 - Опис структури БД.

Відношення	Атрибут	Тип атрибуту
hotel – містить дані про готель	**hotel_id –** унікальний ідентифікатор **name –** назва готелю **restaurant –** наявність ресторану **city –** місто **stars –** кількість зірок	**integer** (числовий) **character varying** (рядок) **boolean** (булевий) **character varying** (рядок) **integer** (числовий)
category – містить дані про категорію номеру у готелі	**category_id** – унікальний ідентифікатор **room_type –** тип номеру **allocation_type –** тип розселення в номері **eating_type –** наявність харчування **price –** ціна	**integer** (числовий) **character varying** (рядок) **character varying** (рядок) **boolean** (булевий) **numeric** (фіксований)
guest – містить дані про постояльців готелю	**guest_id** – унікальний ідентифікатор **name** – ім’я **surname –** прізвище **patronymic –** по батькові **birthday –** день народження **hotel_id –** ідентифікатор готелю	**integer** (числовий) **character varying** (рядок) **character varying** (рядок) **character varying** (рядок) **date** (дата) **integer** (числовий)
room – містить дані щодо номеру	**room_id -** унікальний ідентифікатор **arrival_date –** дата заселення **departure_date –** дата виселення **category_id** – ідентифікатор категорії **hotel_id –** ідентифікатор готелю **number –** номер кімнати	**integer** (числовий) **date** (дата) **date** (дата) **integer** (числовий) **integer** (числовий) **integer** (числовий)
chambermaid – містить дані про покоївок готелю	**chambermaid_id –** унікальний ідентифікатор **name –** ім’я **phone –** номер телефону **salary –** заробітня плата **hotel_id –** ідентифікатор готелю	**integer** (числовий) **character varying** (рядок) **character varying** (рядок) **integer** (числовий) **integer** (числовий)
room/ chambermaid - відношення покоївок до кімнат	**room_id–** ідентифікатор номера **chambermaid_id –** ідентифікатор покоївки	**integer** (числовий) **integer** (числовий)

room/guest - відношення постояльців до кімнат	**room_id–** ідентифікатор номера **chambermaid_id –** ідентифікатор постояльця	**integer** (числовий) **integer** (числовий)

У Обраній базі даних «Готель» можна виділити наступні таблиці: загальні відомості про готель (hotel), тип заданого номера (room), категорія номера (category), загальні відомості про постояльця (guest), інформація про покоївку (chambermaid), відношення покоївок до кімнат (room/chambermaid), відношення постояльців до кімнат (room/guest).

Стовпці заданих таблиць:

1. hotel: hotel_id, name, restaurant, city, stars.
2. room: room_id, arrival date, departure date, category_id, hotel_id, number.
3. category: category_id, room type, allocation type, eating type, price.
4. guest: guest_id, name, surname, patronymic, birthday, hotel_id.
5. chambermaid: chambermaid_id, name, phone number, salary, hotel_id.

Структура програми

За шаблоном проектування MVC, додаток складається з таких модулей:
model.py — підключається до БД та виконує операції – SELECT, INSERT, DELETE, UPDATE та, більш складні операції з БД.
controller.py — головне та допоміжні меню, для зручного керування БД.
mainl.py — точка входу в програму, та підключення до БД.

Схема меню користувача

```
Successfully CONNECTED to database hotel

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:
```

На знімку екрану термінала продемонстровано початкове меню, де можна побачити функції які можна виконати з БД. Кожна команда запускає відповідну функцію з файлу controller.py, яка в подальшому передає аргументи у функцію файлу model.py, яка в свою чергу формує і здійснює запит до бази даних.

Методи реалізовані до пункту 1 завдання лабораторної роботи:

1. `INSERT data in table` — викликає функцію вставки даних у таблицю бази даних;
2. `EDIT data in table` — викликає функцію редагування даних у таблиці бази даних;
3. `DELETE data from table` — викликає функцію видалення даних у таблиці бази даних;
4. `PRINT rows` — викликає функцію виводу даних з таблиці бази даних.

Метод реалізований до пункту 2 завдання лабораторної роботи:

5. `GENERATE random data` — дозволяє користувачеві заповнити таблицю, або всю базу даних випадково згенерованими даними.

Кожна з описаних вище функцій викликає допоміжну функцію `select_table()`, даний скрипт дозволяє користувачеві обирати таблицю

Метод реалізований до пункту 3 завдання лабораторної роботи:

6. `SEARCH data from tables` — викликає функцію пошуку даних у таблицях за атрибутами та поєднання таблиць за ключем.

Завдання 1

Запит на видалення

```python
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
```

Для перевірки роботи розглянемо запити на видалення даних з дочірньої таблиці **room** та батьківської таблиці **hotel**.

Таблиця **hotel** до видалення даних:

hotel_id	name	restaurant	city	star
1	Турист	True	Київ	3
2	Плазма	True	Львів	3
3	Зірка	False	Одеса	2
4	Вікторія	True	Харків	4
5	Маріон	True	Буковель	4
6	Жовтневий	False	Київ	2
7	Хілтон	True	Київ	5
59	9f72fe892	False	7e08464	5
60	9b3033de6	False	3beabd5	4
61	dacac5ae6	True	367c86d	2
62	ba8103fea	False	85a8e67	3

Таблиця **room** до видалення даних:

room_id	arrival_date	departure_date	category_id	hotel_id	number	price
1	2021-10-30	2021-11-10	1	2	4	175.43
2	2020-08-25	2020-09-05	3	2	5	976.50
3	2021-05-15	2021-05-17	6	4	25	1025.25
4	2015-01-07	2015-03-07	1	6	19	200.55
5	2020-12-27	2021-01-03	5	4	4	1480.42
6	2020-12-28	2021-01-04	1	7	32	130.00
7	2020-05-06	2020-05-06	7	3	10	786.60
581	2021-12-01	2021-12-02	5	1	3	370.73
582	2021-11-24	2021-12-09	3	4	21	547.43
583	2021-12-01	2021-12-02	6	4	2	365.79
811	2021-12-02	2021-12-02	66	4	10	500.62
812	2021-12-01	2021-12-02	77	61	11	589.48
813	2021-11-30	2021-12-04	76	61	15	770.96
814	2021-12-01	2021-12-02	6	62	11	559.79

У даній програмній реалізації видалення запису з батьківської таблиці, який зв’язаний з дочірньою таблицею, буде видалено каскадно, тобто всі дані з цим зовнішнім ключем будуть видалені.

Видалимо з таблиці **hotel** рядок з hotel_id = 6:

```
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: 3
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 4
Enter id of row that you want to DELETE
'p' => print rows
'r' => return to menu
6
The row DELETED successfully
```

Таблиця **hotel** після видалення даних:

hotel_id	name	restaurant	city	star
1	Турист	True	Київ	3
2	Плазма	True	Львів	3
3	Зірка	False	Одеса	2
4	Вікторія	True	Харків	4
5	Маріон	True	Буковель	4
7	Хілтон	True	Київ	5
59	9f72fe892	False	7e08464	5
60	9b3033de6	False	3beabd5	4
61	dacac5ae6	True	367c86d	2
62	ba8103fea	False	85a8e67	3

Таблиця **room** після видалення даних:

room_id	arrival_date	departure_date	category_id	hotel_id	number	price
1	2021-10-30	2021-11-10	1	2	4	175.43
2	2020-08-25	2020-09-05	3	2	5	976.50
3	2021-05-15	2021-05-17	6	4	25	1025.25
5	2020-12-27	2021-01-03	5	4	4	1480.42
6	2020-12-28	2021-01-04	1	7	32	130.00
7	2020-05-06	2020-05-06	7	3	10	786.60
581	2021-12-01	2021-12-02	5	1	3	370.73
582	2021-11-24	2021-12-09	3	4	21	547.43
583	2021-12-01	2021-12-02	6	4	2	365.79
811	2021-12-02	2021-12-02	66	4	10	500.62
812	2021-12-01	2021-12-02	77	61	11	589.48
813	2021-11-30	2021-12-04	76	61	15	770.96
814	2021-12-01	2021-12-02	6	62	11	559.79

З даних екранних відображень можна помітити, що при видаленні з таблиці **hotel** рядку з hotel_id = 6, також було видалено рядок з таблиці **room** зі значеннями, де room_id = 4 та hotel_id = 6.

Запит на вставку поля

```python
def insert(choice: int, data: list) -> bool:
    if connection is None or cursor is None:
        return False
    else:
        try:
            match choice:
                case 1:
                    cursor.execute(f"""INSERT INTO public.\"category\" (room_type, allocation_type, eating_type) \
                                    VALUES (\'{data[0]}\', \'{data[1]}\', {data[2]});""")
                case 2:
                    cursor.execute(f"""INSERT INTO public.\"chambermaid\" (name, phone_number, salary, hotel_id) \
                                    VALUES (\'{data[0]}\', \'{data[1]}\', {data[2]}, {data[3]});""")
                case 3:
                    cursor.execute(f"""INSERT INTO public.\"guest\" (name, surname, patronymic, birthday, hotel_id) \
                                    VALUES (\'{data[0]}\', \'{data[1]}\', \'{data[2]}\', \'{data[3]}\', {data[4]});""")
                case 4:
                    cursor.execute(f"""INSERT INTO public.\"hotel\" (name, restaurant, city, star) \
                                    VALUES (\'{data[0]}\', {data[1]}, \'{data[2]}\', {data[3]});""")
                case 5:
                    cursor.execute(f"""INSERT INTO public.\"room\"(arrival_date, departure_date, category_id, hotel_id,\
                     number,price) VALUES(\'{data[0]}\', \'{data[1]}\', {data[2]}, {data[3]}, {data[4]}, {data[5]});""")
                case 6:
                    cursor.execute(f"""INSERT INTO public.\"room/chambermaid\" (room_id, chambermaid_id) \
                                    VALUES ({data[0]}, {data[1]});""")
                case 7:
                    cursor.execute(f"""INSERT INTO public.\"room/guest\" (room_id, guest_id) \
                                    VALUES ({data[0]}, {data[1]});""")
            connection.commit()
        except Exception as _ex:
            print("Impossible to INSERT data into table", _ex)
            return False
    return True
```

Для перевірки роботи розглянемо запити на вставку в таблиці **guest**. Спочатку спробуємо коректний запис, а потім з неіснуючим значенням зовнішнього ключа таблиці **hotel**.

Таблиця **guest** до вставки даних:

guest_id	name	surname	patronymic	birthday	hotel_id
1	Олена	Кузнєцова	Максимівна	1988-06-16	3
2	Аріна	Жукова	Львівна	2002-10-02	5
3	Анна	Березина	Степанівна	1956-01-01	4
4	Артем	Моргунов	Дмитрович	2000-05-29	3
5	Ксенія	Анікіна	Еміровна	1998-06-25	4
6	Анна	Борисова	Макарівна	1987-07-10	1
340	4e32c98a2	46af767912d	768814506b200	1963-02-03	4
341	4d330f2dc	a82df54008c	2ee09cd4de4dd	1982-06-22	7
342	616d3225b	898799b7e06	777720cafacf2	1970-07-22	7

Таблиця **hotel** до вставки даних:

hotel_id	name	restaurant	city	star
1	Турист	True	Київ	3
2	Плазма	True	Львів	3
3	Зірка	False	Одеса	2
4	Вікторія	True	Харків	4
5	Маріон	True	Буковель	4
7	Хілтон	True	Київ	5
59	9f72fe892	False	7e08464	5
60	9b3033de6	False	3beabd5	4
61	dacac5ae6	True	367c86d	2
62	ba8103fea	False	85a8e67	3

Вставимо в таблицю **guest** нові дані (guest_id = 350):

```
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: 1
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 3
Input data separated by comma
Table: guest. Input: name->text, surname->text, patronymic->text, birthday->date, hotel_id->int
Vladyslav, Seletkov, Ruslanovich, 2002-10-30, 7
Data INSERTED successfully

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:
```

Таблиця **guest** після вставки даних (guest_id = 350):

guest_id	name	surname	patronymic	birthday	hotel_id
1	Олена	Кузнєцова	Максимівна	1988-06-16	3
2	Аріна	Жукова	Львівна	2002-10-02	5
3	Анна	Березина	Степанівна	1956-01-01	4
4	Артем	Моргунов	Дмитрович	2000-05-29	3
5	Ксенія	Анікіна	Еміровна	1998-06-25	4
6	Анна	Борисова	Макарівна	1987-07-10	1
340	4e32c98a2	46af767912d	768814506b200	1963-02-03	4
341	4d330f2dc	a82df54008c	2ee09cd4de4dd	1982-06-22	7
342	616d3225b	898799b7e06	777720cafacf2	1970-07-22	7
350	Vladyslav	Seletkov	Ruslanovich	2002-10-30	7

Можна помітити, що вставка виконалася правильно, оскільки зовнішній ключ hotel_id посилається на існуючий запис у табличці **hotel**.

Виконаємо вставку в таблицю **guest** з неіснуючим значенням hotel_id:

```
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: 1
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 3
Input data separated by comma
Table: guest. Input: name->text, surname->text, patronymic->text, birthday->date, hotel_id->int
Daniel, Dzyamyla, Mukolayevuch, 2001-05-23, 75
Impossible to INSERT data into table ОШИБКА:  INSERT или UPDATE в таблице "guest" нарушает ограничение внешнего ключа "fk_guest_hotel"
DETAIL:  Ключ (hotel_id)=(75) отсутствует в таблице "hotel".

Impossible to insert data
```

На екранному відображенні видно, що виникає помилка, оскільки зовнішній ключ hotel_id посилається на неіснуючий запис у табличці **hotel**.

Запит на редагування рядків

```
def update(choice: int, data: list, id1: int, id2: int = 0) -> bool:
    if connection is None or cursor is None:
        return False
    else:
        try:
            match choice:
                case 1:
                    cursor.execute(f"""UPDATE public.\"category\" SET room_type = \'{data[0]}\', \
                    allocation_type = \'{data[1]}\', eating_type = {data[2]} WHERE category_id = {id1};""")
                case 2:
                    cursor.execute(f"""UPDATE public.\"chambermaid\" SET name = \'{data[0]}\', phone_number = \
                    \'{data[1]}\', salary = {data[2]}, hotel_id = {data[3]} WHERE chambermaid_id = {id1};""")
                case 3:
                    cursor.execute(f"""UPDATE public.\"guest\" SET name = \'{data[0]}\', surname = \'{data[1]}\', \
                    patronymic = \'{data[2]}\',birthday = \'{data[3]}\',hotel_id = {data[4]} WHERE guest_id = {id1};""")
                case 4:
                    cursor.execute(f"""UPDATE public.\"hotel\" SET name = \'{data[0]}\', restaurant = {data[1]}, \
                    city = \'{data[2]}\', star = {data[3]} WHERE hotel_id = {id1};""")
                case 5:
                    cursor.execute(f"""UPDATE public.\"room\" SET arrival_date = \'{data[0]}\', departure_date = \
                    \'{data[1]}\', category_id = {data[2]}, hotel_id = {data[3]}, number = {data[4]}, \
                    price = {data[5]} WHERE room_id = {id1};""")
                case 6:
                    cursor.execute(f"""UPDATE public.\"room/chambermaid\" SET room_id = {data[0]}, \
                    chambermaid_id = {data[1]} WHERE room_id = {id1} AND chambermaid_id = {id2};""")
                case 7:
                    cursor.execute(f"""UPDATE public.\"room/guest\" SET room_id = {data[0]}, \
                    guest_id = {data[1]} WHERE room_id = {id1} AND guest_id = {id2};""")
            connection.commit()
        except Exception as _ex:
            print("Impossible to UPDATE data into table", _ex)
            return False
    return True
```

Для перевірки роботи розглянемо запити на редагування в таблиці **room/chambermaid**. Спочатку спробуємо коректний запис, а потім з неіснуючим значенням зовнішнього ключа таблиці **chambermaid**.

Таблиця **room/chambermaid** до редагування даних:

room_id	chambermaid_id
2	7
3	107
5	110
7	4
581	1
582	6

Таблиця **chambermaid** до редагування даних:

```
 chambermaid_id name         phone_number salary       hotel_id
============== ========== ============ ============ ============
1              Іван         380664281802 5000         5
2              Аліна        380688686982 5500         4
3              Єва          380938149603 7750         1
4              Антоніна     380669985293 12000        3
5              Олександра   380662271206 14895        7
6              Аліса        380936566397 8785         2
7              Вероніка     380681217252 6435         4
104            2d15378a543 38a9f91ae86   5398         4
105            4338a818c85 df51c366b33   5884         7
107            2185291cdd0 3250a846206   15030        3
108            7fb87a7183b a10e26aa241   6035         3
109            4314f776e1a 75e60120b3f   6347         1
110            29261748e08 e62ef9cdde7   9398         4
```

Відредагуємо в таблиці **room/chambermaid** рядок з room_id = 7 та chambermaid_id = 4:

```
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: 2
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 6
Enter id of row that you want to UPDATE
'p' => print rows
'r' => return to menu
7
Enter id2 of row that you want to UPDATE
4
If you don't want to UPDATE column -> write as it was
Input data separated by comma
Table: room/chambermaid. Input: room_id->int, chambermaid_id->int
7, 2
UPDATED successfully
```

Таблиця **room/chambermaid** після редагування даних:

room_id	chambermaid_id
2	7
3	107
5	110
7	2
581	1
582	6

Можна помітити, що редагування виконалося правильно (room_id = 7 та chambermaid_id = 2), зовнішній ключ chambermaid_id посилається на вже існуючий запис у табличці **chambermaid**.

Виконаємо редагування даних в таблиці **room/chambermaid**, змінимо значення chambermaid_id, яке не існує у табличці **chambermaid**. Спробуємо відредагувати в таблиці **room/chambermaid** рядок з room_id = 582 та chambermaid_id = 6 на наступні дані room_id = 582 та chambermaid_id = 45.

```
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: 2
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 6
Enter id of row that you want to UPDATE
'p' => print rows
'r' => return to menu
582
Enter id2 of row that you want to UPDATE
6
If you don't want to UPDATE column -> write as it was
Input data separated by comma
Table: room/chambermaid. Input: room_id->int, chambermaid_id->int
582, 45
Impossible to UPDATE data into table ОШИБКА:  INSERT или UPDATE в таблице "room/chambermaid" нарушает ограничение внешнего ключа "fk_chambermaid_room"
DETAIL:  Ключ (chambermaid_id)=(45) отсутствует в таблице "chambermaid".

Impossible to UPDATE table
```

На екранному відображенні чітко видно, що виникає помилка при редагуванні даних, оскільки зовнішній ключ chambermaid_id = 45, посилається на неіснуючий запис у табличці **chambermaid**.

Завдання 2

```python
def generate(choice: int, count: int) -> bool:
    if connection is None or cursor is None:
        return False
    try:
        for i in range(count):
            match choice:
                case 1:
                    cursor.execute(f"""INSERT INTO public.\"category\" (room_type, allocation_type, eating_type) \
                                    VALUES (substr(md5(random()::text), 0, 10), substr(md5(random()::text), 0, 10), \
                                            (round(random())::int)::boolean);""")
                case 2:
                    cursor.execute(f"""INSERT INTO public.\"chambermaid\" (name, phone_number, salary, hotel_id) \
                                    SELECT (substr(md5(random()::text), 0, 12)), \
                                        (substr(md5(random()::character varying(12)), 0, 12)), \
                                        (floor(random() * (25000 - 5000 + 1)) + 5000), \
                                        hotel_id FROM public."hotel" order by random() limit 1;""")
                case 3:
                    cursor.execute(f"""INSERT INTO public.\"guest\" (name, surname, patronymic, birthday, hotel_id) \
                                    SELECT substr(md5(random()::text), 0, 10), \
                                        substr(md5(random()::character varying(12)), 0, 12), \
                                        substr(md5((random() * 2)::text), 0, 14), \
                                        to_timestamp(-286_782_355 + random() * 3_270_071_999), \
                                        hotel_id FROM public."hotel" order by random() limit 1;""")
                case 4:
                    cursor.execute(f"""INSERT INTO public.\"hotel\" (name, restaurant, city, star) \
                                    VALUES (substr(md5(random()::text), 0, 10), (round(random())::int)::boolean, \
                                        substr(md5(random()::text), 0, 8), (floor(random() * (5 - 1 + 1)) + 1));""")
                case 5:
                    cursor.execute(f"""INSERT INTO public.\"room\"(arrival_date, departure_date, number, price, \
                                    category_id, hotel_id) \
                                    SELECT NOW() + (random() * (NOW() - NOW() - '360 days')), \
                                        NOW() + (random() * (NOW() - NOW() + '360 days')), \
                                        floor(random() * (1000 - 10 + 1) + 10), random() * (50000 - 500 ) + 500, \
                                        category_id, hotel_id FROM public."category", public."hotel" \
                                        order by random() limit 1;""")
                case 6:
                    cursor.execute(f"""INSERT INTO public.\"room/chambermaid\" (room_id, chambermaid_id) \
                                    SELECT room_id, chambermaid_id FROM public."room", public."chambermaid" \
                                        order by random() limit 1;""")
                case 7:
                    cursor.execute(f"""INSERT INTO public.\"room/guest\" (room_id, guest_id) \
                                    SELECT room_id, guest_id FROM public."room", public."guest" \
                                        order by random() limit 1;""")
        connection.commit()
    except Exception as _ex:
        print("Impossible to GENERATE data to database hotel", _ex)
        return False
    return True
```

Вставимо по 5 псевдорандомізованих записів у кожну таблицю.

Записи таблиць перед вставкою даних:

Таблиця **category**:

category_id	room_type	allocation_type	eating_type
1	стандарт	одномісний	True
2	люкс	одномісний	True
3	люкс	двомісний	False
4	стандарт	одномісний	False
5	апартамент	трьохмісний	False
6	люкс	двомісний	True
7	апартмент	двомісний	True

Таблиця **chambermaid**:

chambermaid_id	name	phone_number	salary	hotel_id
1	Іван	380664281802	5000	5
2	Аліна	380688686982	5500	4
3	Єва	380938149603	7750	1
4	Антоніна	380669985293	12000	3
5	Олександра	380662271206	14895	7
6	Аліса	380936566397	8785	2
7	Вероніка	380681217252	6435	4

Таблиця **guest**:

guest_id	name	surname	patronymic	birthday	hotel_id
1	Олена	Кузнєцова	Максимівна	1988-06-16	3
2	Аріна	Жукова	Львівна	2002-10-02	5
3	Анна	Березина	Степанівна	1956-01-01	4
4	Артем	Моргунов	Дмитрович	2000-05-29	3
5	Ксенія	Анікіна	Еміровна	1998-06-25	4
6	Анна	Борисова	Макарівна	1987-07-10	1

Таблиця **hotel**:

hotel_id	name	restaurant	city	star
1	Турист	True	Київ	3
2	Плазма	True	Львів	3
3	Зірка	False	Одеса	2
4	Вікторія	True	Харків	4
5	Маріон	True	Буковель	4
7	Хілтон	True	Київ	5

Таблиця **room**:

room_id	arrival_date	departure_date	category_id	hotel_id	number	price
1	2021-10-30	2021-11-10	1	2	4	175.43
2	2020-08-25	2020-09-05	3	2	5	976.50
3	2021-05-15	2021-05-17	6	4	25	1025.25
5	2020-12-27	2021-01-03	5	4	4	1480.42
6	2020-12-28	2021-01-04	1	7	32	130.00
7	2020-05-06	2020-05-06	7	3	10	786.60

Таблиця **room/chambermaid**:

room_id	chambermaid_id
2	7
3	1
5	2

Таблиця **room/guest**:

room_id	guest_id
3	2
7	1

Таблиці після вставки згенерованих записів:

```
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: 5

1. Generate data for all tables
2. Generate data for one table

Choose the table: 1
Input the data quantity to GENERATE: 5
Data GENERATED and INSERTED into table category successfully
Data GENERATED and INSERTED into table chambermaid successfully
Data GENERATED and INSERTED into table guest successfully
Data GENERATED and INSERTED into table hotel successfully
Data GENERATED and INSERTED into table room successfully
Data GENERATED and INSERTED into table room/chambermaid successfully
Data GENERATED and INSERTED into table room/guest successfully
```

Таблиця **category**:

category_id	room_type	allocation_type	eating_type
1	стандарт	одномісний	True
2	люкс	одномісний	True
3	люкс	двомісний	False
4	стандарт	одномісний	False
5	апартамент	трьохмісний	False
6	люкс	двомісний	True
7	апартмент	двомісний	True
82	5ba0adc4f	66e51bbd9	True
83	cdfd7cb09	c870903a1	True
84	39d19f3a5	4a2165c3c	True
85	c88317440	74131e70d	True
86	fe7965e8a	e954dd60d	True

Таблиця **chambermaid**:

chambermaid_id	name	phone_number	salary	hotel_id
1	Іван	380664281802	5000	5
2	Аліна	380688686982	5500	4
3	Єва	380938149603	7750	1
4	Антоніна	380669985293	12000	3
5	Олександра	380662271206	14895	7
6	Аліса	380936566397	8785	2
7	Вероніка	380681217252	6435	4
114	f8c91610f17	d0b0686efdb	7364	2
115	23137333c6b	dd9623df2c4	6787	5
116	59acc5387fb	59234a9ba60	6495	7
117	8db158ab1e3	5b486b89136	6887	1
118	9cc9d0fa378	75b232033de	6417	3

Таблиця **guest**:

guest_id	name	surname	patronymic	birthday	hotel_id
1	Олена	Кузнєцова	Максимівна	1988-06-16	3
2	Аріна	Жукова	Львівна	2002-10-02	5
3	Анна	Березина	Степанівна	1956-01-01	4
4	Артем	Моргунов	Дмитрович	2000-05-29	3
5	Ксенія	Анікіна	Еміровна	1998-06-25	4
6	Анна	Борисова	Макарівна	1987-07-10	1
352	826c84376	2fc6d377f24	54dfe98ca9ec3	1964-01-24	1
353	48be48a47	9fd1c505b94	1a975d2a935c7	1982-11-14	7
354	34db14fc1	675691b030f	e42dafc2549db	1963-03-17	3
355	039569fd8	e907d3fa57e	182ad89934136	1979-04-20	5
356	714b3e463	0959f10eec6	f78895bcd4f30	1980-03-01	7

Таблиця **hotel**:

hotel_id	name	restaurant	city	star
1	Турист	True	Київ	3
2	Плазма	True	Львів	3
3	Зірка	False	Одеса	2
4	Вікторія	True	Харків	4
5	Маріон	True	Буковель	4
7	Хілтон	True	Київ	5
69	326b9603d	True	9d4d73f	3
70	3c3cd08c0	True	b85e294	1
71	5263ae602	True	4b8ec29	2
72	3e28909db	True	02e433e	5
73	697773ba8	False	e35507d	1

Таблиця **room**:

room_id	arrival_date	departure_date	category_id	hotel_id	number	price
1	2021-10-30	2021-11-10	1	2	4	175.43
2	2020-08-25	2020-09-05	3	2	5	976.50
3	2021-05-15	2021-05-17	6	4	25	1025.25
5	2020-12-27	2021-01-03	5	4	4	1480.42
6	2020-12-28	2021-01-04	1	7	32	130.00
7	2020-05-06	2020-05-06	7	3	10	786.60
815	2021-11-30	2021-12-04	3	4	14	725.19
816	2021-11-24	2021-12-10	3	73	32	1611.75
817	2021-11-30	2021-12-03	82	69	14	709.50
818	2021-11-23	2021-12-11	86	7	34	1728.97
819	2021-12-01	2021-12-03	85	71	13	677.72

Таблиця **room/chambermaid**:

room_id	chambermaid_id
2	1
2	7
3	1
3	2
5	2
6	3
815	115
816	6

Таблиця **room/guest**:

room_id	guest_id
2	1
3	2
5	5
6	355
6	2
7	1
817	353

Завдання 3

```python
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 hotel", _ex)
        return ()
    return rows, run_time
```

Виконаємо об’єднання двох таблиць та пошук у них за трьома атрибутами:

Пошук перший:

```
Choose the first table
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 1
Choose the second table
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 5
Input the connecting key: category_id
Input the expression. Use "first" and "second" to address to the table attributes, if with string use like
first.room_type like 'люкс' and second.arrival_date = '2020-08-25' or second.departure_date = '2021-12-04'
SQL request: SELECT * FROM public."category" as first INNER JOIN public."room" as second on first."category_id" = second."category_id"
 WHERE first.room_type like 'люкс' and second.arrival_date = '2020-08-25' or second.departure_date = '2021-12-04'

 category_id room_type allocation_type eating_type room_id   arrival_date departure_date category_id hotel_id  number    price
=========== ========= ============== =========== ========= ============ ============== =========== ========= ========= =========
3           люкс      двомісний      False       815       2021-11-30   2021-12-04     3           4         14        725.19
3           люкс      двомісний      False       2         2020-08-25   2020-09-05     3           2         5         976.50

Time of the executing program: 6980.5  ms
```

Пошук другий:

```
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: 6
Choose the first table
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 4
Choose the second table
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu
Choose the table: 4
Choose the second table
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 2
Input the connecting key: hotel_id
Input the expression. Use "first" and "second" to address to the table attributes, if with string use like
first.restaurant = 'True' and second.salary >= 7000
SQL request: SELECT * FROM public."hotel" as first INNER JOIN public."chambermaid" as second on first."hotel_id" = second."hotel_id"
 WHERE first.restaurant = 'True' and second.salary >= 7000

 hotel_id name   restaurant city   star   chambermaid_id name        phone_number salary       hotel_id
======== ====== ========== ====== ====== ============== =========== ============ ============ ============
1        Турист True       Київ   3      3              Єва         380938149603 7750         1
7        Хілтон True       Київ   5      5              Олександра  380662271206 14895        7
2        Плазма True       Львів  3      6              Аліса       380936566397 8785         2
2        Плазма True       Львів  3      114            f8c91610f17 d0b0686efdb  7364         2

Time of the executing program: 1002.8  ms
```

Третій пошук:

```
Choose the first table
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 4
Choose the second table
1. category
2. chambermaid
3. guest
4. hotel
5. room
6. room/chambermaid
7. room/guest
0. menu

Choose the table: 3
Input the connecting key: hotel_id
Input the expression. Use "first" and "second" to address to the table attributes, if with string use like
first.star >= 3 and second.name like 'Анна' and second.birthday >= '1955-01-01'
SQL request: SELECT * FROM public."hotel" as first INNER JOIN public."guest" as second on first."hotel_id" = second."hotel_id"
 WHERE first.star >= 3 and second.name like 'Анна' and second.birthday >= '1955-01-01'

 hotel_id name     restaurant city     star     guest_id name     surname  patronymic birthday   hotel_id
======== ======== ========== ======== ======== ======== ======== ========= ========== ==========
4        Вікторія True       Харків   4        3        Анна     Березина Степанівна 1956-01-01 4
1        Турист   True       Київ     3        6        Анна     Борисова Макарівна  1987-07-10 1

Time of the executing program: 997.5  ms
```

Завдання 4

Код програмного 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="postgres",
            database="hotel_db",
            port="5432"
        )

        cursor = connection.cursor()

        print("Successfully CONNECTED to database hotel")

        # cursor.execute("SELECT version();")
        # print(f"Server version {cursor.fetchone()}")

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


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


def insert(choice: int, data: list) -> bool:
    if connection is None or cursor is None:
        return False
    else:
        try:
            match choice:
                case 1:
                    cursor.execute(f"""INSERT INTO public.\"category\" (room_type,
allocation_type, eating_type) \
                                   VALUES (\'{data[0]}\', \'{data[1]}\',
{data[2]});""")
                case 2:
                    cursor.execute(f"""INSERT INTO public.\"chambermaid\" (name,
phone_number, salary, hotel_id) \
                                   VALUES (\'{data[0]}\', \'{data[1]}\', {data[2]},
{data[3]});""")
                case 3:
```

```
                    cursor.execute(f"""INSERT INTO public.\"guest\" (name, surname,
patronymic, birthday, hotel_id) \
                                       VALUES (\'{data[0]}\', \'{data[1]}\',
\'{data[2]}\', \'{data[3]}\', {data[4]});""")
                case 4:
                    cursor.execute(f"""INSERT INTO public.\"hotel\" (name,
restaurant, city, star) \
                                       VALUES (\'{data[0]}\', {data[1]}, \'{data[2]}\',
{data[3]});""")
                case 5:
                    cursor.execute(f"""INSERT INTO public.\"room\"(arrival_date,
departure_date, category_id, hotel_id,\
                     number,price) VALUES(\'{data[0]}\', \'{data[1]}\', {data[2]},
{data[3]}, {data[4]}, {data[5]});""")
                case 6:
                    cursor.execute(f"""INSERT INTO public.\"room/chambermaid\"
(room_id, chambermaid_id) \
                                       VALUES ({data[0]}, {data[1]});""")
                case 7:
                    cursor.execute(f"""INSERT INTO public.\"room/guest\" (room_id,
guest_id) \
                                       VALUES ({data[0]}, {data[1]});""")
            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:
            match choice:
                case 1:
                    cursor.execute(f"""UPDATE public.\"category\" SET room_type =
\'{data[0]}\', \
                    allocation_type = \'{data[1]}\', eating_type = {data[2]} WHERE
category_id = {id1};""")
                case 2:
                    cursor.execute(f"""UPDATE public.\"chambermaid\" SET name =
\'{data[0]}\', phone_number = \
                    \'{data[1]}\', salary = {data[2]}, hotel_id = {data[3]} WHERE
chambermaid_id = {id1};""")
                case 3:
                    cursor.execute(f"""UPDATE public.\"guest\" SET name =
\'{data[0]}\', surname = \'{data[1]}\', \
                    patronymic = \'{data[2]}\',birthday = \'{data[3]}\',hotel_id =
{data[4]} WHERE guest_id = {id1};""")
                case 4:
                    cursor.execute(f"""UPDATE public.\"hotel\" SET name =
\'{data[0]}\', restaurant = {data[1]}, \
                    city = \'{data[2]}\', star = {data[3]} WHERE hotel_id =
{id1};""")
                case 5:
                    cursor.execute(f"""UPDATE public.\"room\" SET arrival_date =
\'{data[0]}\', departure_date = \
                    \'{data[1]}\', category_id = {data[2]}, hotel_id = {data[3]},
number = {data[4]}, \
                    price = {data[5]} WHERE room_id = {id1};""")
                case 6:
                    cursor.execute(f"""UPDATE public.\"room/chambermaid\" SET room_id
= {data[0]}, \
                    chambermaid_id = {data[1]} WHERE room_id = {id1} AND
chambermaid_id = {id2};""")
                case 7:
                    cursor.execute(f"""UPDATE public.\"room/guest\" SET room_id =
{data[0]}, \
                    guest_id = {data[1]} WHERE room_id = {id1} AND guest_id =
{id2};""")
            connection.commit()
        except Exception as _ex:
```

```
            print("Impossible to UPDATE data into table", _ex)
            return False
    return True


def generate(choice: int, count: int) -> bool:
    if connection is None or cursor is None:
        return False
    try:
        for i in range(count):
            match choice:
                case 1:
                    cursor.execute(f"""INSERT INTO public.\"category\" (room_type,
allocation_type, eating_type) \
                                    VALUES (substr(md5(random()::text), 0, 10),
substr(md5(random()::text), 0, 10), \
                                            (round(random())::int)::boolean);""")
                case 2:
                    cursor.execute(f"""INSERT INTO public.\"chambermaid\" (name,
phone_number, salary, hotel_id) \
                                    SELECT (substr(md5(random()::text), 0, 12)), \
                                        (substr(md5(random()::character varying(12)),
0, 12)), \
                                        (floor(random() * (25000 - 5000 + 1)) +
5000), \
                                        hotel_id FROM public."hotel" order by
random() limit 1;""")
                case 3:
                    cursor.execute(f"""INSERT INTO public.\"guest\" (name, surname,
patronymic, birthday, hotel_id) \
                                    SELECT substr(md5(random()::text), 0, 10), \
                                        substr(md5(random()::character varying(12)),
0, 12), \
                                        substr(md5((random() * 2)::text), 0, 14), \
                                        to_timestamp(-286782355 + random() *
3270071999), \
                                        hotel_id FROM public."hotel" order by
random() limit 1;""")
                case 4:
                    cursor.execute(f"""INSERT INTO public.\"hotel\" (name,
restaurant, city, star) \
                                    VALUES (substr(md5(random()::text), 0, 10),
(round(random())::int)::boolean, \
                                        substr(md5(random()::text), 0, 8),
(floor(random() * (5 - 1 + 1)) + 1));""")
                case 5:
                    cursor.execute(f"""INSERT INTO public.\"room\"(arrival_date,
departure_date, number, price, \
                                    category_id, hotel_id) \
                                    SELECT NOW() + (random() * (NOW() - NOW() - '360
days')), \
                                        NOW() + (random() * (NOW() - NOW() + '360
days')), \
                                        floor(random() * (1000 - 10 + 1) + 10),
random() * (50000 - 500 ) + 500, \
                                        category_id, hotel_id FROM public."category",
public."hotel" \
                                        order by random() limit 1;""")
                case 6:
```

```
                    cursor.execute(f"""INSERT INTO public.\"room/chambermaid\"
(room_id, chambermaid_id) \
                                       SELECT room_id, chambermaid_id FROM
public."room", public."chambermaid" \
                                           order by random() limit 1;""")
                case 7:
                    cursor.execute(f"""INSERT INTO public.\"room/guest\" (room_id,
guest_id) \
                                       SELECT room_id, guest_id FROM public."room",
public."guest" \
                                           order by random() limit 1;""")
        connection.commit()
    except Exception as _ex:
        print("Impossible to GENERATE data to database hotel", _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 hotel", _ex)
        return ()
    return rows, run_time
```

Даний модуль є точкою доступу до бази даних з програми. Саме в ньому реалізуються всі запити. Для цього в ньому використовується бібліотека – psycopg2.

Функція connect() – намагається підключитись до БД.

Функція disconnect() – намагається відключитись від БД.

Функція insert(choice, data) – намагається вставити в таблицю по номеру choice дані зі списку data.

Функція delete(table, key_name, key_val) – намагається видали з таблиці table рядок з ключем key_name, який == key_val.

Функція select_by_key(table, key_name, key_val) – намагається взяти дані з таблиці table рядок з ключем key_name, який == key_val.

Функція select_by_table(table, quantity, offset) – намагається взяти дані з таблиці table та відсортувати їх у порядку зростання по primary key.

Функція update(choice, data, id1, id2) – намагається змінити в таблиці choice, даними зі списку data, рядок з первинним ключем, який == id, якщо первинний ключ складений, тоді має дорівнювати id1 та id2.

Функція generate(choice, count) – генерує дані і намагається вставити в таблицю choice, count разів ці дані.

Функція search(tables, key, value) – пробує об’єднати і взяти дані з таблиць.