НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра системного проектування

Звіт

про виконання лабораторної роботи №2
з дисципліни «Системи баз даних»
Варіант 5

Виконав:

студент 3 курсу, групи ИТ ЗП-01
Косенко Антон Павлович
Прийняв:
Кислий Р.В.

Київ – 2021

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

Мета:

Здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Завдання:

1. Реалізувати функції 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).

Хід виконання:

Підготовка:

1) Встановлено IDE PyCharm.
2) Встановлюємо бібліотеку psycopg2 для роботи з PostgreSQL.

Виконання завдання:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

Створимо додаток та виконаємо з'єднання з базою даних "booking", створеною у лабораторній роботі №1, а також створимо курсор, як зображено на рисунку 1.

```
import psycopg2

connection = psycopg2.connect(
   database="booking",
   user="postgres",
   password="postgres",
   host="127.0.0.1",
   port="5432"
)

cursor = connection.cursor()
print("Database opened successfully")
```

Рисунок 1. Підключення до бази даних.

Результат роботи програми зображено на рисунку 2.

```
C:\Users\kosen\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\kosen
Database opened successfully

Process finished with exit code 0
```

Рисунок 2. Результат роботи програми.

Реалізуємо функцію внесення наступних даних у таблицю events.

event_id	title	starts	ends	venue_id
4	Workers Day	2021-01-05	2021-01-05	
5	Easter	2021-02-05	2021-02-05	

Програмна реалізація зображена на рисунку 3.

```python
def execute_query(connection, query):
    connection.autocommit = True
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        print("Query executed successfully")
    except OperationalError as e:
        print(f"The error '{e}' occurred")

events = [
    (4, "Workers Day", "2021-01-05", "2021-01-05"),
    (5, "Easter", "2021-02-05", "2021-02-05")
]
events_records = ", ".join(["%s"] * len(events))

insert_query = (
    f"INSERT INTO events (event_id, title, starts, ends) VALUES {events_records}"
)

connection.autocommit = True
cursor = connection.cursor()
cursor.execute(insert_query, events)
```

Рисунок 3. Внесення даних у базу даних.

Результат роботи програми зображено на рисунку 4.

```
booking=# SELECT event_id, title, starts, ends FROM events;
 event_id |      title      |       starts        |        ends
----------+-----------------+---------------------+---------------------
        1 | LARP Club       | 2012-02-14 00:00:00 | 2012-02-15 00:00:00
        2 | April Fools Day | 2012-04-01 00:00:00 | 2012-04-09 00:00:00
        3 | Christmas Day   | 2012-12-20 00:00:00 | 2012-12-25 00:00:00
        4 | Workers Day     | 2021-01-05 00:00:00 | 2021-01-05 00:00:00
        5 | Easter          | 2021-02-05 00:00:00 | 2021-02-05 00:00:00
(5 rows)
```

Рисунок 4. Результат внесення даних у базу даних.

Виконаємо зміну даних у записі в таблиці events. Змінимо дані в графі “title” з “Workers Day” на “International Labors Day”.

Програмна реалізація зображена на рисунку 5.

```python
import psycopg2
from psycopg2 import Error


def update_table(event_id, title):
    try:
        # Подключиться к существующей базе данных
        connection = psycopg2.connect(user="postgres",
                                      # пароль, который указали при установке PostgreSQL
                                      password="postgres",
                                      host="127.0.0.1",
                                      port="5432",
                                      database="booking")

        cursor = connection.cursor()
        print("Таблица до обновления записи")
        sql_select_query = """SELECT * FROM events WHERE event_id = %s"""
        cursor.execute(sql_select_query, (event_id,))
        record = cursor.fetchone()
        print(record)
        # Обновление отдельной записи
        sql_update_query = """UPDATE events SET title = %s WHERE event_id = %s"""
        cursor.execute(sql_update_query, (title, event_id))
        connection.commit()
        count = cursor.rowcount
        print(count, "Запись успешно обновлена")

        print("Таблица после обновления записи")
        sql_select_query = """SELECT * FROM events WHERE event_id = %s"""
        cursor.execute(sql_select_query, (event_id,))
        record = cursor.fetchone()
        print(record)

    except (Exception, Error) as error:
        print("Ошибка при работе с PostgreSQL", error)
    finally:
        if connection:
            cursor.close()
            connection.close()
            print("Соединение с PostgreSQL закрыто")

update_table(4, "International Labor Day")
```

Рисунок 5. Зміна даних в таблиці.

Результат виконання програми наведено на рисунку 6.

```
booking=# SELECT event_id, title, starts, ends FROM events;
 event_id |      title      |       starts        |        ends
----------+-----------------+---------------------+---------------------
        1 | LARP Club       | 2012-02-14 00:00:00 | 2012-02-15 00:00:00
        2 | April Fools Day | 2012-04-01 00:00:00 | 2012-04-09 00:00:00
        3 | Christmas Day   | 2012-12-20 00:00:00 | 2012-12-25 00:00:00
        4 | Workers Day     | 2021-01-05 00:00:00 | 2021-01-05 00:00:00
        5 | Easter          | 2021-02-05 00:00:00 | 2021-02-05 00:00:00
(5 rows)


booking=# SELECT event_id, title, starts, ends FROM events;
 event_id |          title          |       starts        |        ends
----------+-------------------------+---------------------+---------------------
        1 | LARP Club               | 2012-02-14 00:00:00 | 2012-02-15 00:00:00
        2 | April Fools Day         | 2012-04-01 00:00:00 | 2012-04-09 00:00:00
        3 | Christmas Day           | 2012-12-20 00:00:00 | 2012-12-25 00:00:00
        5 | Easter                  | 2021-02-05 00:00:00 | 2021-02-05 00:00:00
        4 | International Labor Day | 2021-01-05 00:00:00 | 2021-01-05 00:00:00
(5 rows)
```

Рисунок 6. Таблиця events до та після змін.

Вилучимо додані записи 4 та 5 з таблиці events.
Програмна реалізація наведена на рисунку 7.

```python
import psycopg2
from psycopg2 import Error


def delete_data(event_id):
    try:
        # Подключиться к существующей базе данных
        connection = psycopg2.connect(user="postgres",
                                      password="postgres",
                                      host="127.0.0.1",
                                      port="5432",
                                      database="booking")

        cursor = connection.cursor()
        # Удаление записи
        sql_delete_query = """DELETE FROM events WHERE event_id = %s"""
        cursor.execute(sql_delete_query, (event_id,))
        connection.commit()
        count = cursor.rowcount
        print(count, "Запись успешно удалена")

    except (Exception, Error) as error:
        print("Ошибка при работе с PostgreSQL", error)
    finally:
        if connection:
            cursor.close()
            connection.close()
            print("Соединение с PostgreSQL закрыто")

delete_data(4)
delete_data(5)
```

Рисунок 7. Видалення записів з таблиці.

Результат роботи програми наведено на рисунку 8.

```
booking=# SELECT event_id, title, starts, ends FROM events;
 event_id |          title          |       starts        |        ends
----------+-------------------------+---------------------+---------------------
        1 | LARP Club               | 2012-02-14 00:00:00 | 2012-02-15 00:00:00
        2 | April Fools Day         | 2012-04-01 00:00:00 | 2012-04-09 00:00:00
        3 | Christmas Day           | 2012-12-20 00:00:00 | 2012-12-25 00:00:00
        5 | Easter                  | 2021-02-05 00:00:00 | 2021-02-05 00:00:00
        4 | International Labor Day | 2021-01-05 00:00:00 | 2021-01-05 00:00:00
(5 rows)


booking=# SELECT event_id, title, starts, ends FROM events;
 event_id |      title      |       starts        |        ends
----------+-----------------+---------------------+---------------------
        1 | LARP Club       | 2012-02-14 00:00:00 | 2012-02-15 00:00:00
        2 | April Fools Day | 2012-04-01 00:00:00 | 2012-04-09 00:00:00
        3 | Christmas Day   | 2012-12-20 00:00:00 | 2012-12-25 00:00:00
(3 rows)
```

Рисунок 8. Таблиця events до та після змін.

2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.

Встановити розширення для автоматичного генерування унікального ідентифікатора.

```
booking=# CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION
booking=# SELECT FROM pg_available_extensions;
--
(86 rows)


booking=# \df
                                  List of functions
 Schema |        Name        | Result data type |    Argument data types     | Type
--------+--------------------+------------------+----------------------------+------
 public | uuid_generate_v1   | uuid             |                            | func
 public | uuid_generate_v1mc | uuid             |                            | func
 public | uuid_generate_v3   | uuid             | namespace uuid, name text  | func
 public | uuid_generate_v4   | uuid             |                            | func
 public | uuid_generate_v5   | uuid             | namespace uuid, name text  | func
 public | uuid_nil           | uuid             |                            | func
 public | uuid_ns_dns        | uuid             |                            | func
 public | uuid_ns_oid        | uuid             |                            | func
 public | uuid_ns_url        | uuid             |                            | func
 public | uuid_ns_x500       | uuid             |                            | func
(10 rows)
```

Рисунок 9. Встановлення розширення “uuid-ospp”.

Створимо таблицю та заповнимо один з параметрів автоматично рандомно-згенерованими значеннями, як зображено на рисунку 10.

booking/postgres@PostgreSQL 13

Query Editor | Query History

```
1  CREATE TABLE tourists (
2      id    uuid DEFAULT uuid_generate_v4(),
3      name  text NOT NULL
4  );
5
6  INSERT INTO tourists (name)
7       VALUES ('Brandon'),
8              ('Andy'),
9              ('James')
10   RETURNING id, name;
```

Data Output | Explain | Messages | Notifications

	id uuid	name text
1	bde838...	Brandon
2	982fa7...	Andy
3	565c23...	James

Рисунок 10. Створення таблиці та заповнення одного з параметрів автоматично рандомними значеннями.

Програмна реалізація зображена на рисунку 11.

```python
import psycopg2
from psycopg2 import Error

try:
    # Подключиться к существующей базе данных
    connection = psycopg2.connect(user="postgres",
                                  # пароль, который указали при установке PostgreSQL
                                  password="postgres",
                                  host="127.0.0.1",
                                  port="5432",
                                  database="booking")

    # Создайте курсор для выполнения операций с базой данных
    cursor = connection.cursor()
    # SQL-запрос для создания новой таблицы
    create_table_query = '''CREATE TABLE tourists
                          (ID UUID DEFAULT uuid_generate_v4(),
                          NAME TEXT NOT NULL); '''
    # Выполнение команды: это создает новую таблицу
    cursor.execute(create_table_query)
    connection.commit()
    print("Таблица успешно создана в PostgreSQL")

except (Exception, Error) as error:
    print("Ошибка при работе с PostgreSQL", error)
finally:
    if connection:
        cursor.close()
        connection.close()
        print("Соединение с PostgreSQL закрыто")
```

Рисунок 11. Створення таблиці та заповнення одного з параметрів автоматично рандомними значеннями.

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.

Реалізуємо наступний пошук, відповідно до поставленого завдання:

1) Здійснимо пошук назви міста в якому як відомо з нашої бази даних має відбутися одна з двох перших подій;
2) Здійснимо пошук назви країни в якій має відбутися подія в назві якої зустрічається “LARP”;
3) Здійснимо пошук назви країни для якої значення логічного типу в графі “private” таблиці місць відповідає значенню “True”;
4) Здійснимо пошук назви місця в якому має відбутися подія у рамках дат з “2012-01-01” по “2012-03-31”.

Програмна реалізація наведена на рисунку 12.

```
import psycopg2
from psycopg2 import Error

try:
    # Подключиться к существующей базе данных
    connection = psycopg2.connect(user="postgres",
                                  # пароль, который указали при установке PostgreSQL
                                  password="postgres",
                                  host="127.0.0.1",
                                  port="5432",
                                  database="booking")

    cursor = connection.cursor()
    cursor.execute("SELECT c.name FROM cities c LEFT JOIN venues v "
                   "ON c.country_code=v.country_code LEFT JOIN events e "
                   "ON v.venue_id=e.venue_id WHERE e.event_id BETWEEN 1 AND 2"
                   )
    record = cursor.fetchall()
    print("Результат", record)

    cursor = connection.cursor()
    cursor.execute("SELECT c.country_name FROM countries c LEFT JOIN cities ci "
                   "ON ci.country_code=c.country_code LEFT JOIN venues v "
                   "ON v.country_code=ci.country_code LEFT JOIN events e "
                   "ON e.venue_id=v.venue_id WHERE e.title LIKE 'LARP%'"
                   )
    record = cursor.fetchall()
    print("Результат", record)

    cursor = connection.cursor()
    cursor.execute("SELECT c.country_name FROM countries c LEFT JOIN cities ci "
                   "ON c.country_code=ci.country_code LEFT JOIN venues v "
                   "ON ci.country_code=v.country_code WHERE private is True"
                   )
    record = cursor.fetchall()
    print("Результат", record)

    cursor = connection.cursor()
    cursor.execute("SELECT v.name FROM venues v LEFT JOIN events e "
                   "ON v.venue_id=e.venue_id WHERE e.starts BETWEEN SYMMETRIC '2012-01-01' AND '2012-03-31'"
                   )
    record = cursor.fetchall()
    print("Результат", record)

except (Exception, Error) as error:
    print("Ошибка при работе с PostgreSQL", error)
finally:
    if connection:
        cursor.close()
        connection.close()
        print("Соединение с PostgreSQL закрыто")
```

Рисунок 12. Реалізація пошуку.

Результат виконання програми наведено на рисунку 13.

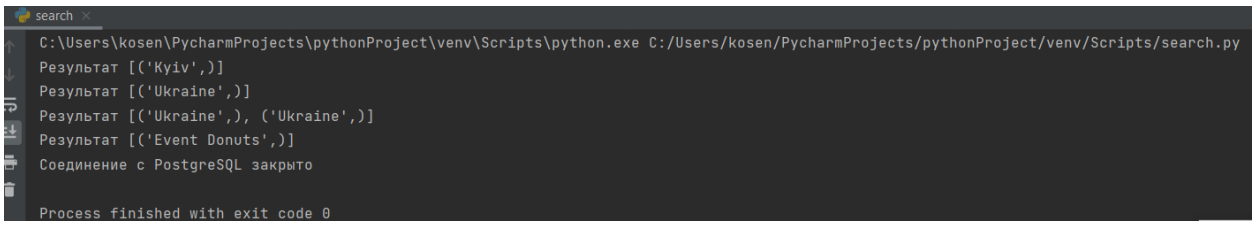

Рисунок 13. Результат пошуку.

ВИСНОВОК

В ході виконання лабораторної роботи було набуто навички створення додатків для роботи з базами даних PostgreSQL.

Написано скрипти для:

- додавання записів в таблиці;
- зміни записів в таблиці;
- видалення записів з таблиці;
- пошук записів в базі даних.

Для написання коду обрано мову python та середовище розробки PyCharm.

Для відлагодження запитів використовувалася програма pgAdmin4.