



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики  
Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 2**  
з дисципліни “Бази даних ”

Виконав

студент II курсу

групи КП-03

Заїка Максим Олександрович

Перевірив

“ \_\_\_\_ ” “ \_\_\_\_\_ ” 20\_\_ р.

викладач

Радченко Костянтин

Олександрович

варіант № 4

Київ 2021

## Тема

Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL

## Мета

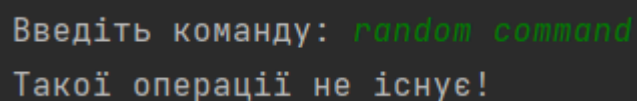
Здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

## Завдання

1. Реалізувати функції внесення, редагування та видалення даних у таблицях бази даних, створених у лабораторній роботі No1, 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 роботи

Обробка виключних ситуацій:



```
Введіть команду: random command
Такої операції не існує!
```

Рис. 1.1 – Невірна команда

```

Введіть команду: get
Введіть назву таблиці: random table name
Введіть умову відбору (all для всіх): all
ПОМИЛКА: відношення "random table name" не існує
LINE 1: SELECT * FROM "random table name" LIMIT 10 OFFSET 0
                        ^

```

Рис. 1.2 – Невірна назва таблиці

```

Введіть команду: insert
Введіть назву таблиці: authors
Введіть назви стовпців: name birthday
Введіть значення атрибутів: some name, asdfsfa
ПОМИЛКА: неприпустимий синтаксис для типу date: "asdfsfa"
LINE 1: ...authors" ("name","birthday") VALUES ('some name', 'asdfsfa')
                                                ^

```

Рис. 1.3 – Невірні параметри


| 1           | <i>select count(*) from authors</i>   |         |          |               |
|-------------|---|---------|----------|---------------|
| Data Output |   | Explain | Messages | Notifications |
|             | count<br>bigint  |         |          |               |
| 1           | 100015  |         |          |               |

Рис. 1.4 – Підтвердження кількості згенерованих даних

|    |  id<br>[PK] bigint |  birthday<br>date |  name<br>text |
|----|---|--|--|
| 5  | 6   | 1916-04-27   | MURDOWIHFX   |
| 6  | 7   | 1958-05-08   | WXGHEVSFEQ   |
| 7  | 8   | 1947-11-10   | FICTCXVMYA   |
| 8  | 9   | 1934-05-18   | XYOJSECUNK   |
| 9  | 10  | 1913-01-10   | BKRVCSSEXAD  |
| 10 | 11  | 1973-10-28   | DWIJEJSMEV   |
| 11 | 12  | 1935-08-31   | BRFORKYLRP   |
| 12 | 13  | 1927-06-17   | WDBGCERTYT   |
| 13 | 14  | 1962-08-15   | GGGTODJUJQ   |
| 14 | 15  | 1928-09-27   | VSBDVQRSU  |
| 15 | 16  | 1906-10-10   | JBSWRKMRAC   |
| 16 | 17  | 1993-12-12   | BYIHLUSOCB   |

Рис. 1.5 – Фрагменти згенерованих даних

```

Введіть команду: get
Введіть назву таблиці: authors, books
Введіть умову відбору (all для всіх): authors.id = books.author_id, authors.name = 'Test'
    id    birthday    name  year  pages  author_id
0  6824  1900-01-01  Test name  1990    300    100018
Час виконання запиту: 0.011997699737548828 секунд.
Успішно!

```

Рис. 1.6 – Приклад пошуку за декількома атрибутами з декількох таблиць

| Код контролера  |
|---|
| <pre> import repository as r import dataview as d import time  class Controller:     def __init__(self):         self._page: int = 0         self._rep = r.Repository()         self._view = d.get_formatted_data      def start(self) -&gt; None:         methods: dict = {             'insert': self._on_insert,             'update': self._on_update, </pre> |

```

        'get': self._on_get,
        'delete': self._on_delete
    }
    while True:
        command: str = input('Введіть команду: ').strip()
        if command == 'exit':
            return
        elif command == 'help':
            self._on_help()
            continue
        try:
            method = methods[command]
            table_name: str = input('Введіть назву таблиці: ').strip()
            method(table_name)
            print('Успішно!')
        except KeyError:
            print('Такої операції не існує!')
        except Exception as err:
            print(err)

    def _on_insert(self, table_name: str) -> None:
        columns: tuple = tuple(input('Введіть назви стовпців: ').strip().split())
        values: tuple = tuple(input('Введіть значення атрибутів: ').strip().split())
        try:
            self._rep.insert(table=table_name, columns=columns, value=values)
        except Exception as err:
            print(err)

    def _on_update(self, table_name: str) -> None:
        condition = self._read_condition()
        updated = None
        while True:
            updated = input('Введіть оновлені значення: ').strip()
            if updated:
                break
        self._rep.update(table=table_name, updated=updated, condition=condition)

    def _on_delete(self, table_name: str) -> None:
        condition = self._read_condition()
        self._rep.delete(table=table_name, condition=condition)

    def _on_get(self, table_name: str) -> None:
        condition = self._read_condition()
        start = time.time()
        data: dict = self._rep.get(table=table_name, condition=condition)
        end = time.time()
        print(self._view(data))
        print(f'Час виконання запиту: {end - start} секунд.')

    @classmethod
    def _read_condition(cls) -> str:
        while True:
            condition = input('Введіть умову відбору (all для всіх): ').strip()
            if not condition:
                continue
            if condition == 'all':
                condition = None
            else:
                condition = cls._prepare_condition(condition)
            return condition

    @staticmethod
    def _prepare_condition(condition: str) -> str:
        return condition.replace(',', ' and ')

    @staticmethod
    def _on_help():
        print('date format: "year-month-year"\n')

```

```

'table name : table1, table2, ...'
'insert : 1) column1 column2 ... 2) value1 value2 ... \n'
'get : 1) column1 = value1, column2 < value2, column3 like value3, ... \n'
'delete : 1) column1 = value1, ... \n'
'update : 1) column1 = value1, ... 2) column1 = value1, ... \n'

```

## Код вигляду

```
import pandas as pd
```

```
def get_formatted_data(data: dict) -> str:
    return str(pd.DataFrame(data)) if data else 'Пусто'
```

## Код моделі

```

import psycopg2 as ps
from psycopg2.extras import RealDictCursor
from psycopg2 import sql

class Repository:
    def __init__(self):
        self._connection = ps.connect(dbname="books", user="postgres",
                                       password="123123", host="localhost")
        self._connection.autocommit = True

    def insert(self, table: str, columns: tuple, value: tuple) -> None:
        with self._connection.cursor() as cursor:
            query = sql.SQL('INSERT INTO {} ({} VALUES {}').format(
                sql.Identifier(table),
                sql.SQL(',').join(map(sql.Identifier, columns)),
                sql.Literal(value)
            )
            cursor.execute(query)

    def get(self, table: str, condition: str = None, count: int = 10, offset: int = 0) -> dict:
        result: dict
        with self._connection.cursor(cursor_factory=RealDictCursor) as cursor:
            query = sql.SQL('SELECT * FROM {}').format(
                sql.SQL(',').join(map(sql.Identifier, table.split(', ')))
            )

            if condition is not None:
                query += sql.SQL(' WHERE {}').format(sql.SQL(condition))
            query += sql.SQL(' LIMIT {} OFFSET {}').format(
                sql.Literal(count),
                sql.Literal(offset)
            )
            cursor.execute(query)
            result = cursor.fetchall()
        return result

    def delete(self, table: str, condition: str) -> None:
        with self._connection.cursor() as cursor:
            query = sql.SQL('DELETE FROM {} WHERE {}').format(
                sql.Identifier(table),
                sql.Literal(condition)
            )
            cursor.execute(query)

    def update(self, table: str, updated: str, condition: str) -> None:
        with self._connection.cursor() as cursor:
            query = sql.SQL('UPDATE {} SET {} WHERE {}').format(
                sql.Identifier(table),
                sql.SQL(updated),
                sql.SQL(condition)
            )
            cursor.execute(query)

```

```
def __del__(self):  
    self._connection.close()
```

## **Висновок**

У даній роботі було генерацію рандомних даних в СУБД PostgreSQL за допомогою інструментів мови SQL. Досліджено роботу з фреймворком psycopg2 для мови Python.