



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”

тема “Створення додатку бази даних, орієнтованого на взаємодію з СУБД
PostgreSQL”

Виконав(ла)
студент(ка) І курсу
групи КП-03

Тимощук Роман Олександрович
(прізвище, ім'я, по батькові)

Перевірів
“ ____ ” “ ____ ” 20__ р.
викладач

Радченко Константин
Олександрович
(прізвище, ім'я, по батькові)

Київ 2021

Мета роботи

Здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Постановка 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 помилок

```
Enter command
smash
Entered command does not exist
Enter command
█

deleteVisitor 5 11
Incorrect number of arguments entered
Enter command
█

insertVisitor Roman 333 Moshkun
ПОМИЛКА: неприпустимий синтаксис для типу integer: "Moshkun"
LINE 2:          VALUES ('Roman', '333', 'Moshkun')
                                   ^

Enter command
█
```

Фрагменти згенерованих таблиць в результаті пошукових запитів

YJ58	13.0	58.0
X061	13.0	58.0
YQ76	8.0	63.0
JQ12	8.0	63.0
EE75	8.0	63.0
KY90	17.0	68.0

visitor2	JU	19
visitor3	IH	20
visitorAB	VW	33
visitorAD	BQ	29
visitorBW	MH	31
visitorER	IM	23

visitorRK	BT	65
visitorER	IM	66
visitorPB	VF	67
visitorAI	VF	67
visitor2	JU	69
visitorOQ	WK	69
visitorBW	MH	70















Введення пошукового запиту та його результат

```
1 select firstname, lastname, age
2   from visitors v full outer join abonnements q on q.visitor_id = v.visitor_id
3   where age > 20 and age < 40 order by age
```

Data Output Explain Messages Notifications

	firstname character varying (50)	lastname character varying (50)	age integer	
1	visitorGJ	LO	21	
2	Roman	333	22	
3	visitor2	OK	24	
4	visitorIN	RC	24	
5	visitorCO	WN	24	
6	visitorCG	test	29	

Файли програми з репозиторію Git

 .vscode	 __pycache__
 __pycache__	
 model	 deleteData.py
 config.py	 executeQuery.py
 controller.py	 generateData.py
 executeCommands.py	
 main.py	 insertData.py
 view.py	 updateData.py

Программный код MVC

Model

```
class DeleteData:

    def __init__(self, connection):
        self.connection = connection

    def delete_visitor(self, id):
        cursor = self.connection.cursor()
        delete_query = """DELETE FROM visitors WHERE visitor_id = %s"""
        cursor.execute(delete_query, id)
        self.connection.commit()

        cursor.close()

    def delete_gym(self, id):
        cursor = self.connection.cursor()
        delete_query = """DELETE FROM gyms WHERE gym_id = %s"""
        cursor.execute(delete_query, id)
        self.connection.commit()

        cursor.close()

    def delete_simulator(self, id):
        cursor = self.connection.cursor()
        delete_query = """DELETE FROM simulators WHERE simulator_id = %s"""
        cursor.execute(delete_query, id)
        self.connection.commit()

        cursor.close()

    def delete_abonnement(self, id):
        cursor = self.connection.cursor()
        delete_query = """DELETE FROM abonnements WHERE visitor_id = %s"""
        cursor.execute(delete_query, id)
        self.connection.commit()

        cursor.close()
```

```

class ExecuteQuery:

    def __init__(self, connection):
        self.connection = connection

    def select_simulators_by_fee(self, input):
        cursor = self.connection.cursor()
        select_query = """select model, area, fee from simulators s
                           full outer join gyms q on q.gym_id = s.gym_id where fee > %s and fee <
%s order by fee"""
        cursor.execute(select_query, input)
        data = cursor.fetchall()
        cursor.close()

        return data

    def select_visitors_with_abonnements(self, data):
        cursor = self.connection.cursor()
        select_query = """SELECT visitors.firstname, visitors.lastname, visitors.visitor_id from
                           visitors inner join abonnements ON
                           visitors.visitor_id = abonnements.visitor_id where
visitors.visitor_id > %s order by firstname"""
        cursor.execute(select_query, data)
        data = cursor.fetchall()
        cursor.close()

        return data

    def select_abonnements_by_age(self, input):
        cursor = self.connection.cursor()
        select_query = """select firstname, lastname, age
                           from visitors v full outer join abonnements q on q.visitor_id =
v.visitor_id
                           where age > %s and age < %s order by age"""
        cursor.execute(select_query, input)
        data = cursor.fetchall()
        cursor.close()

        return data

```

```

class GenerateData:

    def __init__(self, connection):
        self.connection = connection

    def generate_visitors(self, number):
        cursor = self.connection.cursor()
        generate_query = """INSERT INTO visitors (firstname, lastname, age)
                           SELECT 'visitor' || chr(trunc(65+random()*25)::int) ||
                           chr(trunc(65+random()*25)::int) as firstname ,
                           chr(trunc(65+random()*25)::int) ||
                           chr(trunc(65+random()*25)::int) as lastname,
                           trunc(random()*(99-5)+5)::int as age
                           FROM generate_series(1,%s) """
        cursor.execute(generate_query, number)
        self.connection.commit()
        cursor.close()

    def generate_gyms(self, number):
        cursor = self.connection.cursor()
        generate_query = """INSERT INTO gyms (address, area, fee)
                           SELECT trunc(random()*(200-1)+1)::int || ', ' ||

```

```

        substr(md5(random()::text), 1,20) || ' Street' as address,
        trunc(random()*(20-5)+5)::int as area,
        trunc(random()*(200-50)+5)::int as fee
    FROM generate_series(1,%s)"""
    cursor.execute(generate_query, number)
    self.connection.commit()
    cursor.close()

def generate_simulators(self, number):
    cursor = self.connection.cursor()
    cursor.execute("SELECT gyms.gym_id from gyms ORDER BY random() LIMIT 1")
    a = cursor.fetchone()
    generate_query = f"""INSERT INTO simulators (gym_id, model, weight)

        SELECT {a[0]} as gym_id ,
        chr(trunc(65+random()*25)::int) ||
        chr(trunc(65+random()*25)::int) ||
        trunc(random()*(100-1)+1)::int as model,
        trunc(random()*(50-10)+10)::int as weight
    FROM generate_series(1, %s) """
    cursor.execute(generate_query, number)
    self.connection.commit()
    cursor.close()

```

```

class InsertData:

```

```

    def __init__(self, connection):
        self.connection = connection

    def insert_visitor(self, input):
        cursor = self.connection.cursor()
        insert_query = """INSERT INTO visitors (firstname, lastname, age)
                                VALUES (%s, %s, %s)"""
        cursor.execute(insert_query, input)
        self.connection.commit()
        cursor.close()

    def insert_gym(self, input):
        cursor = self.connection.cursor()
        insert_query = """INSERT INTO gyms (address, area, fee)
                                VALUES (%s, %s, %s)"""
        cursor.execute(insert_query, input)
        self.connection.commit()
        cursor.close()

    def insert_simulator(self, input):
        cursor = self.connection.cursor()
        insert_query = """INSERT INTO simulators (gym_id, model, weight)
                                VALUES (%s, %s, %s)"""
        cursor.execute(insert_query, input)
        self.connection.commit()

        cursor.close()

    def insert_abonnement(self, input):
        cursor = self.connection.cursor()
        insert_query = """INSERT INTO abonnements (visitor_id, gym_id)
                                VALUES (%s, %s)"""
        cursor.execute(insert_query, input)
        self.connection.commit()
        cursor.close()

```

```

class UpdateData:

    def __init__(self, connection):
        self.connection = connection

    def update_lastname(self, input):
        cursor = self.connection.cursor()
        update_query = f"""UPDATE visitors SET lastname = %s WHERE visitor_id = %s"""
        cursor.execute(update_query, input)
        self.connection.commit()
        cursor.close()

    def update_fee(self, input):
        cursor = self.connection.cursor()
        update_query = f"""UPDATE gyms SET fee = %s WHERE gym_id = %s"""
        cursor.execute(update_query, input)
        self.connection.commit()
        cursor.close()

```

Controller

```

import executeCommands
import time
import view
class Controller:
    def __init__(self, deleteData, executeQuery, generateData, insertData, updateData, view) :
        self.deleteData = deleteData
        self.executeQuery = executeQuery
        self.generateData = generateData
        self.insertData = insertData
        self.updateData = updateData
        self.view = view
        self.command = {}
        self.command["deleteVisitor"] = self.deleteData.delete_visitor
        self.command["deleteGym"] = self.deleteData.delete_gym
        self.command["deleteSimulator"] = self.deleteData.delete_simulator
        self.command["deleteAbonnement"] = self.deleteData.delete_abonnement
        self.command["printSimulatorsByFee"] = executeQuery.select_simulators_by_fee
        self.command["printVisitorsWithAbs"] = executeQuery.select_visitors_with_abonnements
        self.command["printAbonnementsByAge"] = executeQuery.select_abonnements_by_age
        self.command["generateVisitors"] = generateData.generate_visitors
        self.command["generateGyms"] = generateData.generate_gyms
        self.command["generateSimulators"] = generateData.generate_simulators
        self.command["updateLastname"] = updateData.update_lastname
        self.command["updateFee"] = updateData.update_fee
        self.command["insertVisitor"] = insertData.insert_visitor
        self.command["insertGym"] = insertData.insert_gym
        self.command["insertSimulator"] = insertData.insert_simulator
        self.command["insertAbonnement"] = insertData.insert_abonnement

    def handleCommand(self, command):
        commands = ["deleteVisitor", "deleteGym", "deleteSimulator", "deleteAbonnement",
                    "printSimulatorsByFee", "printVisitorsWithAbs", "printAbonnementsByAge",
                    "generateVisitors", "generateGyms", "generateSimulators", "updateLastname",
                    "updateFee", "insertVisitor", "insertGym", "insertSimulator",
                    "insertAbonnement"]
        commandName = command.split(' ')[0]
        if commandName not in commands:
            raise Exception("Entered command does not exist")

        commandData = executeCommands.get_command_args(command)

```

```

        if commandName in ["printSimulatorsByFee", "printAbonnementsByAge",
"printVisitorsWithAbs"]:
            start = time.time()
            fetched_data = self.command[commandName](commandData)
            end = time.time()
            self.view.print_table_of_data(fetched_data)
            print(f"The request was performed in {int((end - start)*1000)} ms")
        else:
            self.command[commandName](commandData)

```

View

```

class View:

    def print_table_of_data(self, data) :
        for row in data:
            subrow = ""
            for element in row:
                subrow += "{:20}".format(str(element))
            print(subrow)

```

Висновки

Виконавши дану лабораторну роботу я **навчився** створювати додаток для взаємодії з СУБД.

В ході роботи **було реалізовано** шаблон MVC (модель-подання-контроллер), а також **використано** бібліотеку psycorp2.