

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Лабораторна робота № 2
з дисципліни «Бази даних»
«Створення додатку бази даних, орієнтованого на взаємодію з СУБД
PostgreSQL»

Виконала:

студентка 3 курсу ФПМ групи КП-82

Морщак Каріна Юріївна

Прийняв:

Радченко К. О.

Київ 2020

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних

PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення 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).

Код програмного додатку

Файл main.py

```
from controller import Controller
Controller().show_init_menu()
```

Файл model.py

```
import psycopg2

class Model:
    def __init__(self):
        try:
            self.connection = psycopg2.connect(host="localhost",
port="5432",

database='university', user='postgres', password='72307247')
            self.cursor = self.connection.cursor()
        except (Exception, psycopg2.Error) as error:
            print("Помилка при з'єднанні з PostgreSQL", error)

    def get_col_names(self):
        return [d[0] for d in self.cursor.description]

    def create_db(self):
        f = open("create_db.txt", "r")

        self.cursor.execute(f.read())
        self.connection.commit()

    def get(self, tname, parameter):
        try:
            query = f'SELECT * FROM {tname}'

            if parameter:
                query += ' WHERE ' + parameter

            self.cursor.execute(query)
            return self.get_col_names(), self.cursor.fetchall()
```

```

        finally:
            self.connection.commit()

    def insert(self, tname, columns, values):
        try:
            query = f'INSERT INTO {tname} ({columns}) VALUES
({values});'

            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def delete(self, tname, parameter):
        try:
            query = f'DELETE FROM {tname} WHERE {parameter};'

            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def update(self, tname, parameter, statement):
        try:
            query = f'UPDATE {tname} SET {statement} WHERE
{parameter}'

            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def search_task_by_student_group(self, groups):
        try:
            query = f'''
SELECT * from task
WHERE id in(
        SELECT task_id FROM student_task
        JOIN student on student_task.student_id=student.id
        WHERE LOWER(student_group) in ({groups.lower()}))

```

```

        );'''
        self.cursor.execute(query)
        return self.get_col_names(), self.cursor.fetchall()
    finally:
        self.connection.commit()

def search_student_by_task_is_passed(self, is_passed):
    try:
        query = f'''
        SELECT * from student
        WHERE id in(
            SELECT student_id FROM student_task
            JOIN task on task.id=student_task.task_id
            WHERE is_passed={is_passed});'''
        self.cursor.execute(query)
        return self.get_col_names(), self.cursor.fetchall()
    finally:
        self.connection.commit()

def fillFacultyByRandomData(self, number):
    sql = f"""
    CREATE OR REPLACE FUNCTION randomFaculties()
        RETURNS void AS $$
    DECLARE
        step integer := 0;
    BEGIN
        LOOP EXIT WHEN step >= {number};
            INSERT INTO faculty (name, number_of_students,
foundation_date)
                VALUES (
                    substring(md5(random()::text), 1, 30),
                    (random() * (2000 - 1) + 100)::integer,
                    timestamp '1900-01-01 20:00:00' +
                    random() * (timestamp '2015-12-31 20:00:00' -
                    timestamp '1900-01-01 10:00:00')
                );
            step := step + 1;
    """

```

```

        END LOOP ;
END;
$$ LANGUAGE PLPGSQL;
SELECT randomFaculties();
"""
try:
    self.cursor.execute(sql)
finally:
    self.connection.commit()

```

Файл view.py

```

from consolemenu import *
from consolemenu.items import *

class View:
    def print(self, data):
        columns, rows = data
        lineLen = 30 * len(columns)

        self.printSeparator(lineLen)
        self.printRow(columns)
        self.printSeparator(lineLen)

        for row in rows:
            self.printRow(row)
            self.printSeparator(lineLen)

    def printRow(self, row):
        for col in row:
            print(str(col).rjust(26, ' ') + ' |', end='')
        print('')

    def printSeparator(self, length):
        print('-' * length)

```

Файл controller.py

```
from consolemenu import SelectionMenu

from model import Model
from view import View

TABLES_NAMES = ['faculty', 'student', 'student_task', 'subject',
                'task']

TABLES = {
    'faculty': ['id', 'name', 'number_of_students',
                'foundation_date'],
    'student': ['id', 'fullname', 'email', 'course', 'faculty_id',
                'student_group'],
    'student_task': ['id', 'student_id', 'task_id'],
    'subject': ['id', 'name', 'description'],
    'task': ['id', 'name', 'subject_id', 'description', 'deadline',
             'is_passed']
}

def getInput(msg, tableName=''):
    print(msg)
    if tableName:
        print(' | '.join(TABLES[tableName]), end='\n\n')
    return input()

def getInsertInput(msg, tableName):
    print(msg)
    print(' | '.join(TABLES[tableName]), end='\n\n')
    return input(), input()

def pressEnter():
    input()
```

```

class Controller:
    def __init__(self):
        self.model = Model()
        self.view = View()

    def show_init_menu(self, msg=''):
        selectionMenu = SelectionMenu(
            TABLES_NAMES + ['Fill table "faculty" by random data (10
items)'],
            title='Select the table to work with | command:',
subtitle=msg)
        selectionMenu.show()

        index = selectionMenu.selected_option
        if index < len(TABLES_NAMES):
            tableName = TABLES_NAMES[index]
            self.show_entity_menu(tableName)
        elif index == 5:
            self.fillByRandom()
        else:
            print('Bye!')

    def show_entity_menu(self, tableName, msg=''):
        options = ['Get', 'Delete', 'Update', 'Insert']
        functions = [self.get, self.delete, self.update, self.insert]

        if tableName == 'task':
            options.append('Search task by student group')
            functions.append(self.search_task_by_student_group)
        elif tableName == 'student':
            options.append('Search student by his task is passed')
            functions.append(self.search_student_by_task_is_passed)

        selectionMenu = SelectionMenu(options, f'Name of table:
{tableName}',
                                     exit_option_text='Back',
subtitle=msg)

```



```

        selectionMenu.show()
    try:
        function = functions[selectionMenu.selected_option]
        function(tableName)
    except IndexError:
        self.show_init_menu()

def get(self, tableName):
    try:
        parameter = getInput(
            f'GET {tableName}\nEnter parameter or leave empty:',
tableName)
        data = self.model.get(tableName, parameter)
        self.view.print(data)
        pressEnter()
        self.show_entity_menu(tableName)
    except Exception as err:
        self.show_entity_menu(tableName, str(err))

def insert(self, tableName):
    try:
        columns, values = getInsertInput(
            f"INSERT {tableName}\nEnter columns divided with
commas and press Enter. Enter values like: 'value1', 'value2', ...",
            tableName)
        self.model.insert(tableName, columns, values)
        self.show_entity_menu(tableName, 'Success!')
    except Exception as err:
        self.show_entity_menu(tableName, str(err))

def delete(self, tableName):
    try:
        parameter = getInput(
            f'DELETE {tableName}\nEnter parameter:', tableName)
        self.model.delete(tableName, parameter)
        self.show_entity_menu(tableName, 'Success!')
    except Exception as err:

```

```

        self.show_entity_menu(tableName, str(err))

def update(self, tableName):
    try:
        parameter = getInput(
            f'UPDATE {tableName}\nEnter parameter:', tableName)
        statement = getInput(
            "Enter SQL statement in format [<key>='<value>']",
tableName)

        self.model.update(tableName, parameter, statement)
        self.show_entity_menu(tableName, 'Success!')
    except Exception as err:
        self.show_entity_menu(tableName, str(err))

def search_task_by_student_group(self, tableName):
    try:
        groups = getInput(
            'Search task where student\'s groups are: \nEnter
groups divided with commas:')
        data = self.model.search_task_by_student_group(groups)
        self.view.print(data)
        pressEnter()
        self.show_entity_menu(tableName)
    except Exception as err:
        self.show_entity_menu(tableName, str(err))

def search_student_by_task_is_passed(self, tableName):
    try:
        is_passed = getInput('Search students that have passed
them tasks.\nIs task done?:').lower() in [
            'true', 't', 'yes', 'y', '+']
        data
        =
self.model.search_student_by_task_is_passed(is_passed)
        self.view.print(data)
        pressEnter()
        self.show_entity_menu(tableName)


```

```
except Exception as err:
    self.show_entity_menu(tableName, str(err))

def fillByRandom(self):
    try:
        number = getInput('Enter the number of random entries in
the table:')
        self.model.fillFacultyByRandomData(number)
        self.show_init_menu('Success!')
    except Exception as err:
        self.show_init_menu(str(err))
```

Приклади результатів

1. Головне меню:

A screenshot of a terminal window with a dark background and light gray text. The text displays a menu for selecting a table to work with. The prompt is "Select the table to work with | command:". Below it, there is a list of options: "1 - faculty", "2 - student", "3 - student_task", "4 - subject", "5 - task", "6 - Fill table 'faculty' by random data", and "7 - Exit". At the bottom left of the terminal, there is a prompt ">>".

```
Select the table to work with | command:

1 - faculty
2 - student
3 - student_task
4 - subject
5 - task
6 - Fill table "faculty" by random data
7 - Exit

>>
```

2. Меню таблиць (на скріншоті таблиця student)

```
Name of table: student

1 - Get
2 - Delete
3 - Update
4 - Insert
5 - Search student by his task is passed
6 - Back
```

3. Операція Get (таблиця faculty). Виведення з параметром (записи таблиці, в яких $id > 3$)

```
GET faculty
Enter parameter in SQL-form or leave empty:
id | name | number_of_students | foundation_date
id>3
-----
      id |          name | number_of_students | foundation_date |
-----
      4 | Institute for Applied Systems Analysis |      890 | 1999-04-28 |
      5 | Heat and power faculty |      743 | 1978-10-10 |
```

4. Операція Delete (таблиця subject) з параметром $id = 7$. Оскільки таблиця task має поле subject_id (id предмета), то після виконання операції Видалення відповідний запис у таблиці task буде мати в атрибуті subject_id значення NULL (при створенні таблиці task було вказано що FOREIGN KEY (subject_id) REFERENCES Subject (Id) ON DELETE SET DEFAULT, а оскільки дефолтне значення не було вказане, записано NULL)

```
DELETE subject
Enter parameter:
id | name | description

id=7

```

Name of table: subject

Success!

- 1 - Get
- 2 - Delete
- 3 - Update
- 4 - Insert
- 5 - Back

Перевіримо відповідний запис в таблиці task:

```
GET task
Enter parameter or leave empty:
id | name | subject_id | description | deadline | is_passed

```

id	name	subject_id	description	deadline
9	Lab1	2	The simplest one	2020-10-01
10	Lab2	2	Normal	2020-10-14
11	Lab3	1	More less	2020-09-09
12	Lab1	1	Simple	2020-11-01
13	Lab2	5	More less	2020-12-05
14	Lab4	4	Terrible	2020-12-25
15	Lab5	4	Nightmare	2020-10-10
16	Lab10	None	blabla	2020-10-11

Тепер атрибут `subject_id` відповідного запису (з `id=16`) має значення `NULL`.

5. Операція Update (таблиця task, параметр id=6)

```
>> 3
UPDATE task
Enter parameter:
id | name | subject_id | description | deadline | is_passed

id=16
Enter SQL statement in format [<key>='<value>']
id | name | subject_id | description | deadline | is_passed

subject_id='3',description='some information'

Name of table: task

Success!

1 - Get
2 - Delete
3 - Update
4 - Insert
5 - Search task by student group
```

Перевірка:

```
>> 1
GET task
Enter parameter or leave empty:
id | name | subject_id | description | deadline | is_passed

-----
id | name | subject_id | description | deadline
-----
9 | Lab1 | 2 | The simplest one | 2020-10-01
10 | Lab2 | 2 | Normal | 2020-10-14
11 | Lab3 | 1 | More less | 2020-09-09
12 | Lab1 | 1 | Simple | 2020-11-01
13 | Lab2 | 5 | More less | 2020-12-05
14 | Lab4 | 4 | Terrible | 2020-12-25
15 | Lab5 | 4 | Nightmare | 2020-10-10
16 | Lab10 | 3 | some information | 2020-10-11
-----
```

6. Операція Insert (таблиця student)

```
>> 4
INSERT student
Enter columns divided with commas and press Enter. Enter values like: 'value1', 'value2', ...
id | fullname | email | course | faculty_id | student_group

fullname, course, student_group
'Ivan Ivanov', '1', 'RT-01'

Name of table: student

Success!

1 - Get
2 - Delete
3 - Update
4 - Insert
5 - Search student by his task is passed
6 - Back
```

Перевірка:

fullname	email	course	faculty_id	student_group
Oleg Vinnik	oleg.vinnik@gmail.com	1	1	KP-91
Valery Meladze	valery.meladze@gmail.com	3	2	HY-63
Artem Piven	artem.piven@gmail.com	2	5	GT-71
Anna Kravchenko	anna.kravchenko@gmail.com	4	3	FR-92
Yuriy Kaplan	yuriy.kaplan@gmail.com	2	4	BT-84
Andrey Ivanov	andrey.ivanov@gmail.com	1	1	KP-91
Sasha Pshenichnaya	sasha.pshenichnaya@gmail.com	3	2	HY-63
Ivan Ivanov	None	1	None	RT-01

7. Пошук студента за критерієм чи здане його завдання

```
>> 5
Search students that have passed them tasks.
Is task done?:
y
```

id	fullname	email	course	faculty_id
3	Artem Piven	artem.piven@gmail.com	2	5
5	Oleg Vinnik	oleg.vinnik@gmail.com	1	1
4	Anna Kravchenko	anna.kravchenko@gmail.com	4	3
2	Sasha Pshenichnaya	sasha.pshenichnaya@gmail.com	3	2
1	Andrey Ivanov	andrey.ivanov@gmail.com	1	1

8. Пошук завдання за групою (групами) студента

```
>> 6
Search task where student's groups are:
Enter groups divided with commas:
KP-91, HY-63
```

id	name	subject_id	description	deadline
14	Lab4	4	Terrible	2020-12-25
13	Lab2	5	More less	2020-12-05
9	Lab1	2	The simplest one	2020-10-01
16	Lab10	3	some information	2020-10-11
11	Lab3	1	More less	2020-09-09

9. Генерування випадкових записів у таблицю faculty (100 000 записів)

```
>> 6
Enter the number of random entries in the table:
100000
Enter the command:

Select the table to work with | command:

Success!

1 - faculty
2 - student
3 - student_task
4 - subject
5 - task
6 - Fill table "faculty" by random data
7 - Exit

>>
```

Перевірка:

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure, including tables like 'faculty', 'student', 'student_task', 'subject', and 'task'. The 'faculty' table is selected, and its columns are listed: 'id' (integer, PK), 'name' (character varying (40)), 'number_of_students' (integer), and 'foundation_date' (date). The main window shows a query editor with the following SQL query:

```
1 SELECT * FROM public.faculty
2 ORDER BY id ASC
```

Below the query editor, a table displays the query results. The table has columns: 'id', 'name', 'number_of_students', and 'foundation_date'. The results show 12 rows of data, including faculty names like 'Faculty of Economics', 'Faculty of Applied Mathematics', 'Faculty of Information Technology', 'Institute for Applied Systems', 'Heat and power faculty', and 'Faculty of Engineering'.

id	name	number_of_students	foundation_date
1	Faculty of Economics	1010	2000-10-03
2	Faculty of Applied Mathematics	734	1998-04-20
3	Faculty of Information Technology	1750	1956-03-19
4	Institute for Applied Systems	890	1999-04-28
5	Heat and power faculty	743	1978-10-10
6	Faculty of Engineering	1581	1905-05-20
7	Faculty of Applied Mathematics	1427	2008-04-17
8	Faculty of Information Technology	1954	1939-11-27
9	Institute for Applied Systems	226	1939-02-07
10	Heat and power faculty	1472	1959-10-23
11	Faculty of Engineering	430	2010-08-13
12	Faculty of Applied Mathematics	430	2010-08-13

A green status bar at the bottom of the query results indicates: 'Successfully run. Total query runtime: 999 msec. 100005 rows affected.'

На скріншоті бачимо запис про те що у таблиці наразі 100 005 записів (з них 5 звичайних, інші – згенеровано випадковим чином)