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»

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

Виконав: студент III курсу

ФПМ групи КВ-84

Азиранкулов Є.А.

Перевірив:

Київ – 2020

Загальне завдання роботи полягає у наступному:

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).

Деталізоване завдання:

1. Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв'язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати **вилучення** рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.
2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не мовою програмування, а відповідним SQL-запитом**!

Приклад генерації 100 псевдовипадкових чисел:

```
select trunc(random()*1000)::int
from generate_series(1,100)
```

Data Output | Explain | Messages | Notific

	trunc integer
1	368
2	773
3	29
4	66
5	497
6	956

Приклад генерації 5 псевдовипадкових рядків:

```
select chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)
from generate_series(1,5)
```

Data Output | Explain | Messages | Notifications

	?column? text
1	NE
2	MQ
3	RN
4	DW
5	DA

Приклад генерації псевдовипадкової мітки часу з діапазону доступний за посиланням.

Кількість даних для генерування має вводити користувач з клавіатури. Для тесту взяти 100 000 записів для однієї-двох таблиць.

Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (foreign key).

3. Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

4. Програмний код організувати згідно шаблону Model-View-Controller(MVC). Приклад організації коду згідно шаблону доступний за даним посиланням. При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

Вимоги до пункту №1 деталізованого завдання:

- ілюстрації обробки виняткових ситуацій (помилок) при уведенні/вилучення даних;
- ілюстрації валідації даних при уведенні користувачем.

Вимоги до пункту №2 деталізованого завдання:

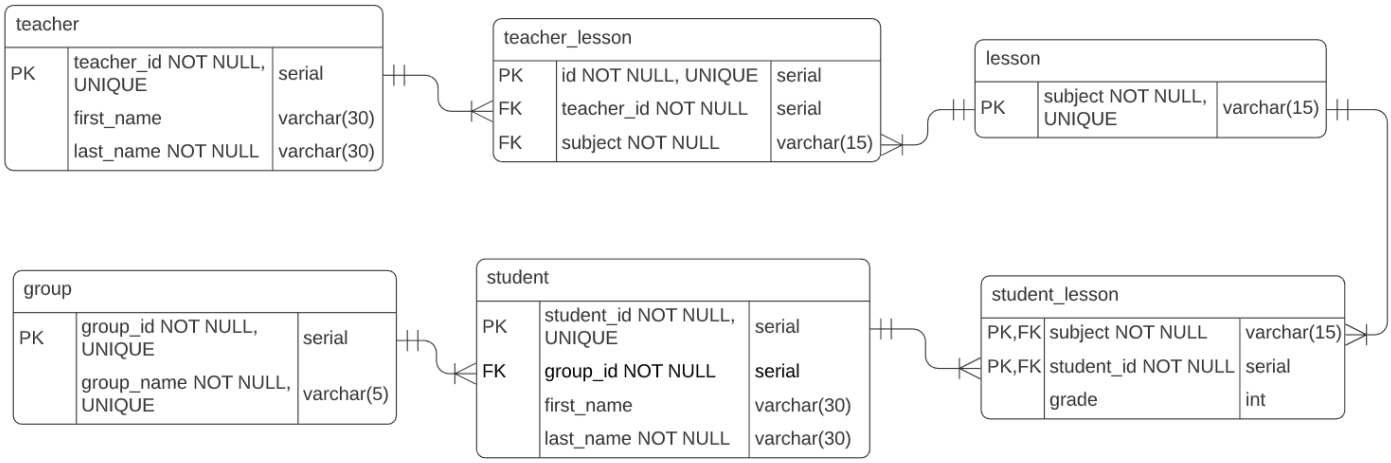
- копії екрану (ілюстрації) з фрагментами згенерованих даних таблиць;
- копії SQL-запитів, що ілюструють генерацію при визначених вхідних параметрах.

Вимоги до пункту №3 деталізованого завдання:

- ілюстрації уведення пошукового запиту та результатів виконання запитів;
- копії SQL-запитів, що ілюструють пошук з зазначеними початковими параметрами

Вимоги до пункту №4 деталізованого завдання:

- ілюстрації програмного коду з репозиторію Git.



GitHub репозиторій

Пункт №1

- ілюстрації обробки виняткових ситуацій (помилок) при уведенні/вилучення даних:

```
1. INSERT
2. DELETE
3. UPDATE
4. RANDOM
5. SELECT
6. EXIT
>1
1
Enter table name:wring
Enter comma-separated values:a,a,a,
Oops, table doesn't exists
```

- ілюстрації валідації даних при уведенні користувачем:

```
1. INSERT
2. DELETE
3. UPDATE
4. RANDOM
5. SELECT
6. EXIT
>2
2
Enter table name:teacher
Enter column:wront_col
Enter value:val
ОШИБКА:  столбец "wront_col" не существует
LINE 1: DELETE FROM "teacher" WHERE "wront_col"='val'
                                    ^
```

Пункт №2

- копії екрану (ілюстрації) з фрагментами згенерованих даних таблиць;

```
1. INSERT
2. DELETE
3. UPDATE
4. RANDOM
5. SELECT
6. EXIT
>4
4
Enter table name:teacher
Enter rows number:100000
INSERT INTO "teacher" ("first_name","last_name") SELECT chr(trunc(65 + random()
```

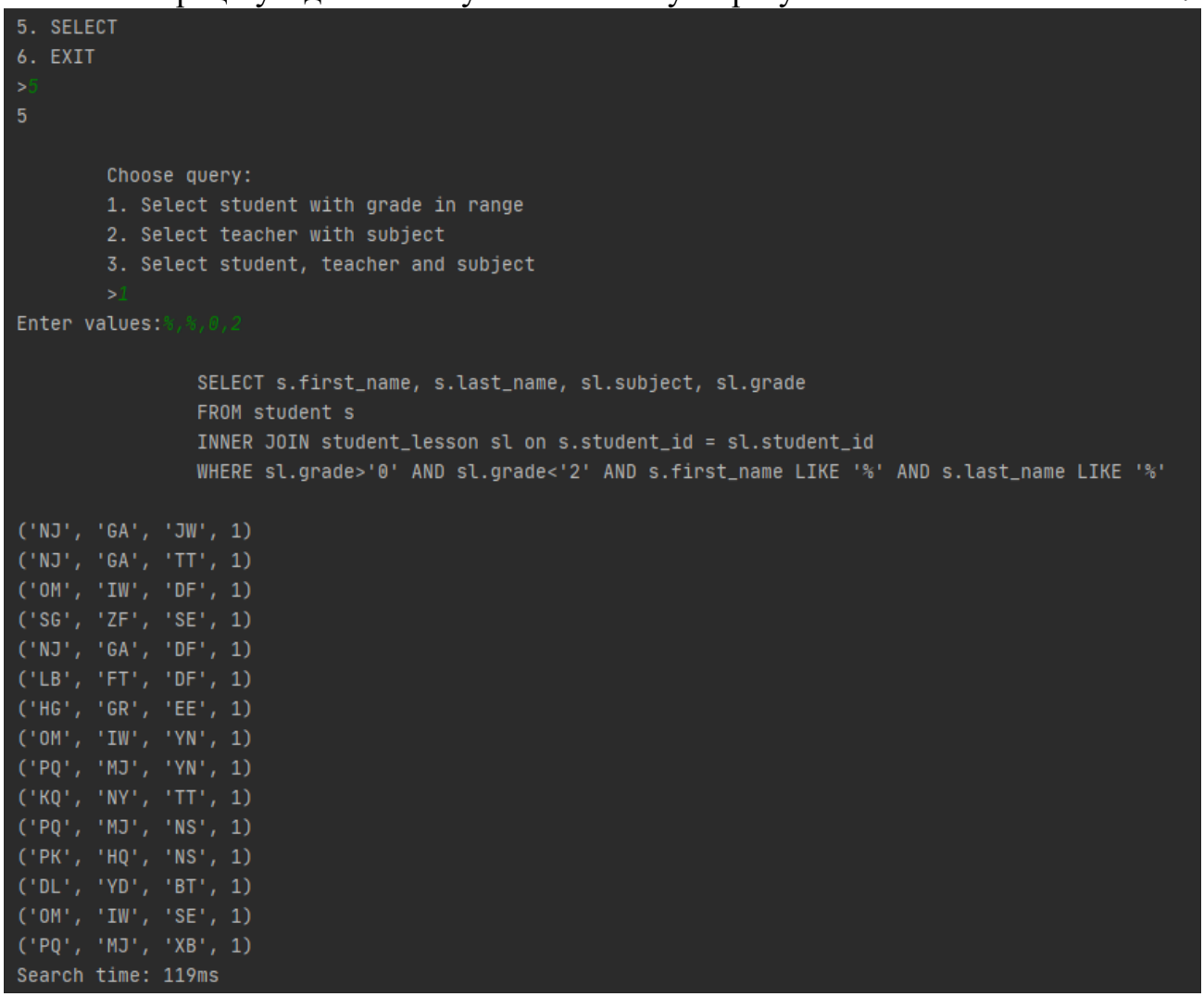
	teacher_id [PK] integer	first_name character varying (30)	last_name character varying (30)
22992	23636	VN	BM
22993	23637	YV	UO
22994	23638	KU	VD
22995	23639	YJ	QJ
22996	23640	BC	ZO
22997	23641	MN	LN
22998	23642	QE	YR
22999	23643	CX	QU
23000	23644	MF	CD
23001	23645	CL	WV
23002	23646	BQ	EN
23003	23647	PZ	UF
23004	23648	NI	UA
23005	23649	CT	XM
23006	23650	OR	BY

- копії SQL-запитів, що ілюструють генерацію при визначених вхідних параметрах:

```
INSERT INTO "teacher" ("first_name","last_name") SELECT chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + random()*26)::int),chr(trunc(65 +
random()*26)::int)||chr(trunc(65 + random()*26)::int) FROM generate_series(1, 100000) ON
CONFLICT DO NOTHING
```

Пункт №3

- ілюстрації уведення пошукового запиту та результатів виконання запитів:



```
5. SELECT
6. EXIT
>5
5

        Choose query:
        1. Select student with grade in range
        2. Select teacher with subject
        3. Select student, teacher and subject
        >1
Enter values:%,%,0,2

                SELECT s.first_name, s.last_name, sl.subject, sl.grade
                FROM student s
                INNER JOIN student_lesson sl on s.student_id = sl.student_id
                WHERE sl.grade>'0' AND sl.grade<'2' AND s.first_name LIKE '%' AND s.last_name LIKE '%'

('NJ', 'GA', 'JW', 1)
('NJ', 'GA', 'TT', 1)
('OM', 'IW', 'DF', 1)
('SG', 'ZF', 'SE', 1)
('NJ', 'GA', 'DF', 1)
('LB', 'FT', 'DF', 1)
('HG', 'GR', 'EE', 1)
('OM', 'IW', 'YN', 1)
('PQ', 'MJ', 'YN', 1)
('KQ', 'NY', 'TT', 1)
('PQ', 'MJ', 'NS', 1)
('PK', 'HQ', 'NS', 1)
('DL', 'YD', 'BT', 1)
('OM', 'IW', 'SE', 1)
('PQ', 'MJ', 'XB', 1)
Search time: 119ms
```

- копії SQL-запитів, що ілюструють пошук з зазначеними початковими параметрами:

```
SELECT s.first_name, s.last_name, sl.subject, sl.grade

FROM student s

INNER JOIN student_lesson sl on s.student_id = sl.student_id

WHERE sl.grade>'0' AND sl.grade<'2' AND s.first_name LIKE '%' AND s.last_name LIKE '%'
```

Пункт №4

master ▾ DB_Labs / lab2 / Go to file A

egorkavin complete second lab 95b3fd2 2 minutes ago

..

README.md	complete second lab	2 min
controller.py	complete second lab	2 min
model.py	complete second lab	2 min
structure.png	complete second lab	2 min
view.py	complete second lab	2 min

README.md

Лабораторна робота №1 з дисципліни «Бази даних і засоби управління»

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