



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

Тупало К.С

Перевірив:

Київ – 2020

Загальне завдання роботи полягає в такому:

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 введення/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 підлеглий 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)
```

	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)
```

	?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.

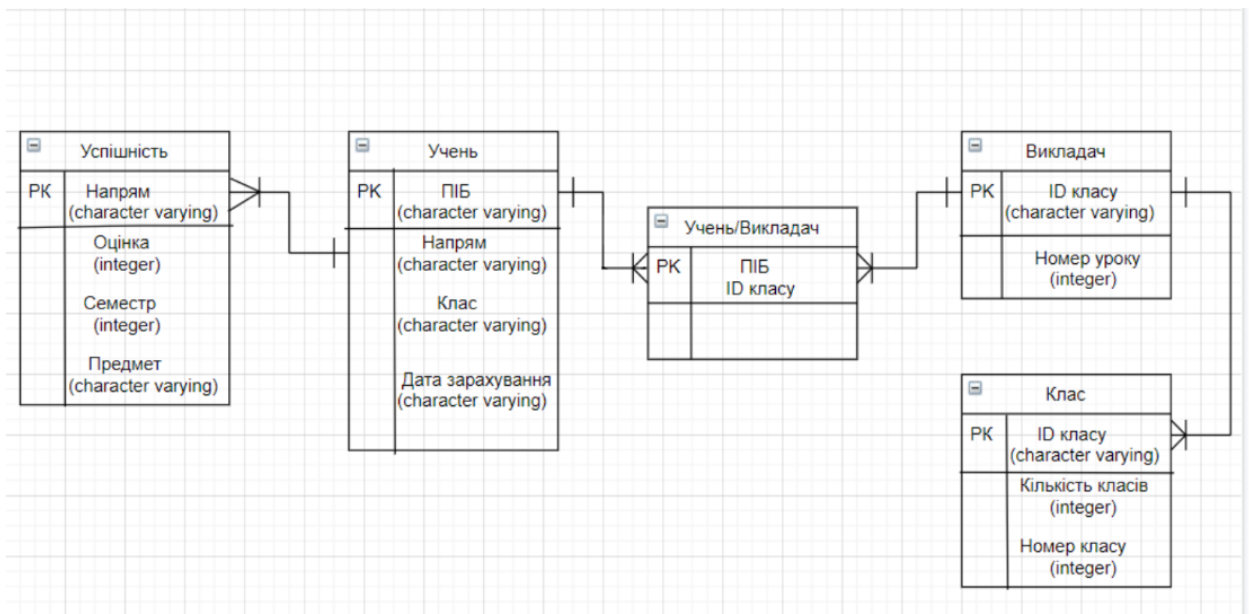
Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (foreign key).

- Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість введення 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).

Рекомендована бібліотека взаємодії з PostgreSQL Psycopg2: <http://initd.org/psycopg/docs/usage.html>



Відповідь на вимоги до пункту №1 деталізованого завдання:

Ілюстрації обробки виняткових ситуацій (помилки) при введенні/вилучення даних:

```
1.UPDATE
2.ADD
3.DELETE
4.RAND
5.SEARCH
Enter the request number >>> 2
Enter table name >>> treacger
WRONG TABLE NAME
```

Ілюстрації валідації даних при введенні користувачем:

```
1.UPDATE
2.ADD
3.DELETE
4.RAND
5.SEARCH
Enter the request number >>> 1
Enter table name >>> student
Student and Success connection 1:N , changes in the "direction" column , touch both tables >>>
Student column >>> ['student_data', 'direction', 'class', 'receipt_date']
Success column >>> ['direction', 'score', 'semester', 'subject']
Enter column name >>> direction
42703
Error ПОМИЛКА: стовпця "direction" не існує
LINE 1: SELECT direction FROM student
          ^
HINT: Можливо, передбачалось посилання на стовпець "student.direction".
```

Вимоги до пункту №2 деталізованого завдання:

Меню генерації:

```
1.UPDATE
2.ADD
3.DELETE
4.RAND
5.SEARCH
Enter the request number >>> 4
Enter table name >>> class
Teacher and Class connection 1:N , changes in the "id_class" column , touch both tables >>>
Teacher column >>> ['class_id', 'lesson_count']
Class column >>> ['id_class', 'class_count', 'class_number']
Input random size >>> 10000
```

Копії екрану з фрагментами згенерованих даних таблиць:

До

1	K-2	2	9
2	M-2	4	11
3	P-3	1	10

Після

9993	𐄓	17685	12488
9994	撈	18305	22739
9995	Ω	27393	25401
9996	𐄒	18873	9864
9997	曜	27294	735
9998	𐄚	1692	19933
9999	⋮	11705	4590
10000	へ	13711	20632

Копії SQLзапитів, що ілюструють генерацію при визначених вхідних параметрах:

```
1.UPDATE
2.ADD
3.DELETE
4.RAND
5.SEARCH
Enter the request number >>> 4
Enter table name >>> student
Student and Success connection 1:N , changes in the "direction" column , touch both tables >>>
Student column >>> ['student_data', 'direction', 'class', 'receipt_date']
Success column >>> ['direction', 'score', 'semester', 'subject']
Input random size >>> 3
WITH test AS(INSERT INTO student SELECT chr(trunc(65+random()*30000)::int), chr(trunc(65 + random()*30000)::int),
chr(trunc(65 + random()*30000)::int),chr(trunc(65 + random()*30000)::int) FROM generate_series(1,{count}) RETURNING direction)
INSERT INTO success SELECT direction FROM test
```

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації введення пошукового запиту та результатів виконання запитів:

```
1.UPDATE
2.ADD
3.DELETE
4.RAND
5.SEARCH
Enter the request number >>> 5
Input quantity of attributes to search by >>> 2
Input name of the attribute number 1 to search by >>> direction
Input name of the attribute number 2 to search by >>> subject
['direction', 'subject']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'direction' INTERSECT ALL SELECT table_name FROM information_schema.columns WHERE info
['character varying', 'character varying']
Input string for direction to search by >>> Mathematics
Input string for subject to search by >>> Geometry
[('Mathematics', 12, 1, 'Geometry')]
Time:0.0009982585906982422 seconds
```








Копії SQL-запитів, що ілюструють генерацію при визначених запитів,
що ілюструють пошук з зазначеними початковими параметрами

```
1.UPDATE
2.ADD
3.DELETE
4.RAND
5.SEARCH
Enter the request number >>> 5
Input quantity of attributes to search by >>> 1
Input name of the attribute number 1 to search by >>> direction
['direction']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'direction'
['character varying', 'character varying']
Input string for direction to search by >>> Mathematics
[('Mathematics',), ('Mathematics',)]
Time:0.001997709274291992 seconds
```

Вимоги до пункту №4 деталізованого завдання:

Ілюстрації програмного коду з репозиторію Git:

idktupalo Add files via upload			068dc15 now 25 commits
	DB_Lab1_Tupalo_84.docx	Add files via upload	last month
	DB_Lab1_Tupalo_84.pdf	Add files via upload	last month
	MyData_sxema.png	Add files via upload	now
	README.md	Update README.md	last month
	controler.py	Add files via upload	now
	model.py	Add files via upload	now
	view.py	Add files via upload	now

README.md



database

Лабораторна робота №2

Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL

База даних з лабораторної №1 : Школа(предмети,вчителі,оцінки,учні)

Лабораторна робота № 1.

Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL