



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

Іванюк В.І

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:




```
Enter command > update
Input table name : order
Input column name : order_i
42601
ПОМИЛКА: синтаксична помилка в або поблизу "order"
LINE 1: SELECT order_i FROM order
                        ^
```

Результат роботи команд (insert ,update , delete)

Insert

	product_id [PK] integer	price integer
1	16532	165
2	63353	885
3	263354	333
4	966832	550

```
Enter command : insert
Input table name > product
Enter 2 value :
215
521
INSERT INTO product VALUES ( 215 , 521 )
Press any key to continue . . .
```




	 product_id [PK] integer 	price integer 
1	215	521
2	16532	165
3	63353	885
4	263354	333
5	966832	550

Update

```

Enter command : update
Input table name : product
Input column name : price
[(165,), (885,), (333,), (550,), (521,)]
Input old_name : 521
Input new_name : 2020
Press any key to continue . . .

```

	 product_id [PK] integer 	price integer 
1	215	2020
2	16532	165
3	63353	885
4	263354	333
5	966832	550

Delete

```

Enter command : delete
Input table name : product
Enter column name : product_id
[(16532,), (63353,), (263354,), (966832,), (215,)]
Enter value : 215
Press any key to continue . . .

```

	product_id [PK] integer	price integer
1	16532	165
2	63353	885
3	263354	333
4	966832	550

Вимоги до пункту №2 деталізованого завдання:
 Меню генерації:

```
Enter command : random
Input table name > order
Random size :10000
INSERT INTO order SELECT (trunc(65+random()*54000)::int),chr(trunc(65+random()*54000)::int) FROM generate_series(1,10000)
```

Копії екрану з фрагментами згенерованих даних таблиць:

9994	3288	匜
9995	47262	ㄥ
9996	905	щ
9997	8324	臍
9998	4659	颯
9999	14313	∅
10000	45109	帶

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації введення пошукового запиту та результатів виконання запитів:

```
Enter command : search
Input quantity of attributes to search by : 2
Input name of the attribute number 1 to search by : product_id
Input name of the attribute number 2 to search by : price
['product_id', 'price']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'product_id'
INTERSECT ALL SELECT table_name FROM information_schema.columns WHERE information_schema.columns.column_name LIKE 'price'
['integer', 'integer']
Enter left limit for product_id 18900
Enter right limit for product_id 75890
Enter left limit for price 512
Enter right limit for price 997
[(63353, 885)]
request time: 0.0009927749633789062 seconds
```

Вимоги до пункту №4 деталізованого завдання:

Ілюстрації програмного коду з репозиторію Git:



Farrongua Add files via upload

..



controler.py

Add files via upload



model.py

Add files via upload



view.py

Add files via upload