



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

Мулярчук М

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.

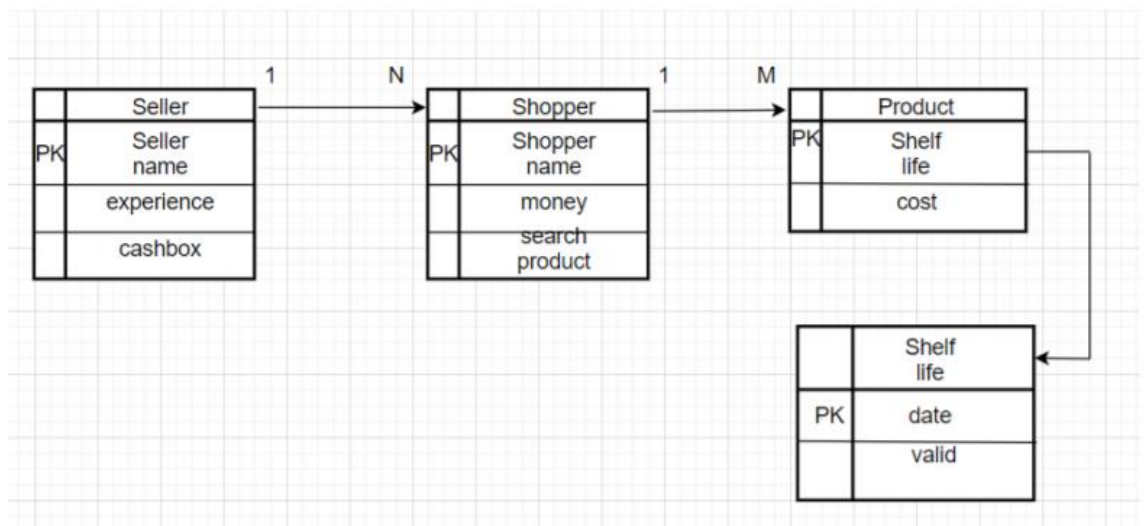
Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (foreign key).

- 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 - delete , 2 - update , 3 - insert , 4 - rand , 5 - search
Input number of request:1
Input table name: product
Input column name: shelf_life
[('2 роки',), ('4 місяці',), ('4 дня',)]
Enter value: 10 year
42601
ПОМИЛКА: синтаксична помилка в або поблизу "year"
LINE 1: DELETE FROM product WHERE shelf_life = 10 year
                                                ^
```

Результат роботи команд (insert ,update , delete)

Початкова таблиця

	seller_name [PK] character varying	experience character varying	cashbox integer
1	Гоголь О.В	10 років	3
2	Петров А.А	2 роки	1
3	Сидорчук	5 років	2

Insert

```
1 - delete , 2 - update , 3 - insert , 4 - rand , 5 - search
Input number of request:3
Input table name: seller
Enter int(value):3
Франко І.Я
7 років
7
```

	seller_name [PK] character varying	experience character varying	cashbox integer
1	Гоголь О.В	10 років	3
2	Петров А.А	2 роки	1
3	Сидорчук	5 років	2
4	Франко І.Я	7 років	7

Delete

```

1 - delete , 2 - update , 3 - insert , 4 - rand , 5 - search
Input number of request:1
Input table name: seller
Input column name: cashbox
[(1,), (2,), (3,), (12,)]
Enter value: 2

```

	seller_name [PK] character varying	experience character varying	cashbox integer
1	Гоголь О.В	10 років	3
2	Петров А.А	2 роки	1
3	Франко І.Я	7 років	12

Update

```

1 - delete , 2 - update , 3 - insert , 4 - rand , 5 - search
Input number of request:2
Input table name: seller
Input column name: cashbox
[(1,), (2,), (3,), (7,)]
Enter new value: 12
Enter old value: 7

```

	seller_name [PK] character varying	experience character varying	cashbox integer
1	Гоголь О.В	10 років	3
2	Петров А.А	2 роки	1
3	Сидорчук	5 років	2
4	Франко І.Я	7 років	12

Вимоги до пункту №2 деталізованого завдання:
Меню генерації:

```
1 - delete , 2 - update , 3 - insert , 4 - rand , 5 - search
Input number of request:4
Input table name: seller
Enter value: 10000
```

```
INSERT INTO seller SELECT chr(trunc(65+random() * 15000)::int),chr(trunc(65+random() * 15000)::int),(trunc(65+random() * 15000)::int) FROM generate_series(1,10000)
```

Копії екрану з фрагментами згенерованих даних таблиць:

9990	𐀀	𐀁	7173
9991	𐀂	𐀃	2479
9992	𐀄	𐀅	9287
9993	𐀆	𐀇	7070
9994	𐀈	𐀉	4987
9995	𐀊	𐀋	5912
9996	𐀌	𐀍	6001
9997	𐀎	𐀏	943
9998	𐀐	𐀑	8686
9999	𐀒	𐀓	6639
10000	𐀔	𐀕	7557

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації введення пошукового запиту та результатів виконання запитів:

```
1 - delete , 2 - update , 3 - insert , 4 - rand , 5 - search
Input number of request:5
Enter int(value):2
Input name of the attribute number 1 to search by : seller_name
Input name of the attribute number 2 to search by : cashbox
['seller_name', 'cashbox']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'sell
er_name' INTERSECT ALL SELECT table_name FROM information_schema.columns WHERE information_schema.columns.column_name LI
KE 'cashbox'
['character varying', 'integer']
Input string for seller_name to search by : Гоголь О.В
Enter left limit for cashbox 1
Enter right limit for cashbox 5
[('Гоголь О.В', '10 років', 3)]
TIME = 0.0009984970092773438 seconds
```

Вимоги до пункту №4 деталізованого завдання:

Ілюстрації програмного коду з репозиторію Git:

main database / controller.py / <> Jump to

Bellis-lab Add files via upload

1 contributor

27 lines (25 sloc) | 790 Bytes

```
1 from model import *
2
3 def controll(request):
4     if request == 1:
5         table_name = table_db()
6         column_name = column_db()
7         information(table_name, column_name)
8         delete_DB(table_name, column_name, value_db())
9     if request == 2:
10        table_name = table_db()
11        column_name = column_db()
12        information(table_name, column_name)
13        new=value_new()
14        old=value_old()
15        update_DB(table_name, column_name, new, old)
16    if request == 3:
17        table_name=table_db()
18        size = value_db_int()
19        insert_DB(table_name, size)
20    if request == 4:
21        table_name=table_db()
22        size = value_db()
23        random_DB(table_name, size)
24    if request == 5:
25        search_DB(value_db_int())
26
27 controll(view_menu())
```


 **main** ▾







 **1 branch**

 **0 tags**

Go to file

Add file ▾

 **Code** ▾

 Bellis-lab Add files via upload	e5828ee 2 minutes ago	 2 commits
 controler.py	Add files via upload	2 minutes ago
 lab2	Create lab2	6 minutes ago
 model.py	Add files via upload	2 minutes ago
 view.py	Add files via upload	2 minutes ago

Help people interested in this repository understand your project by adding a README.

Add a README