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

Вазерцев Д.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 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.

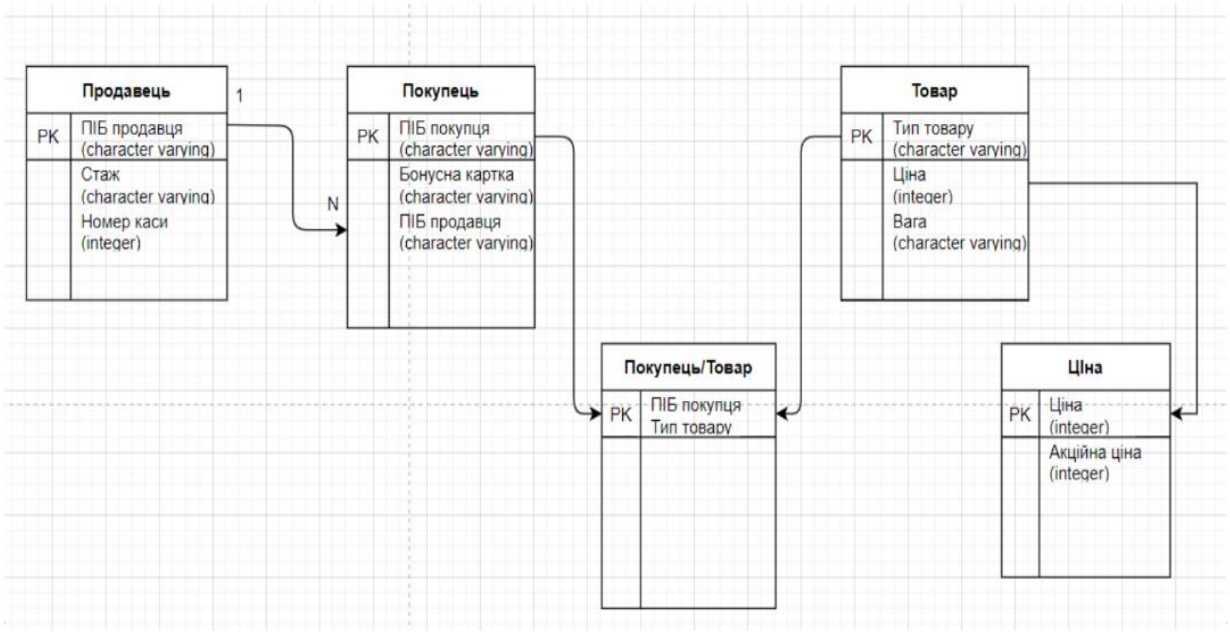
Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності foreign key).

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

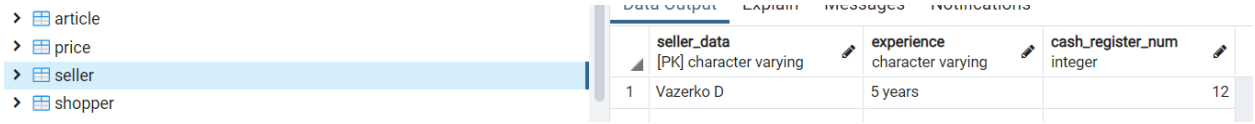
виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

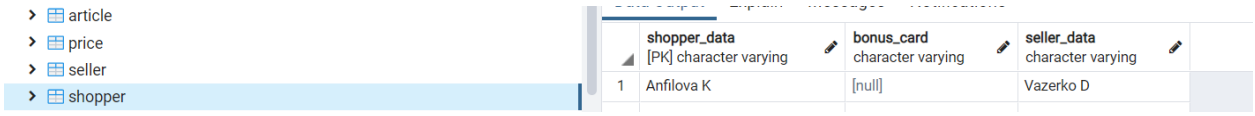
4. Програмний код організувати згідно шаблону Model-View-Controller(MVC). Приклад організації коду згідно шаблону доступний за даним посиланням. При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

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



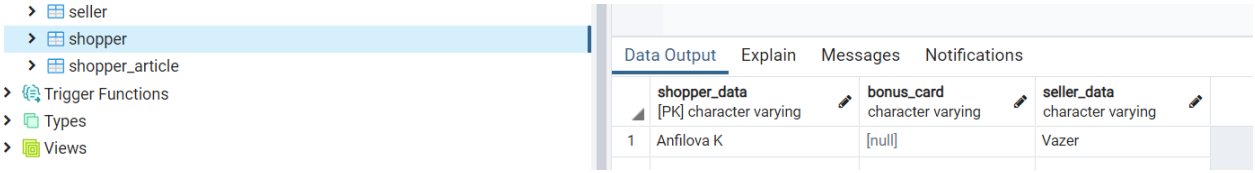
Робота програми з урахуванням foreign key

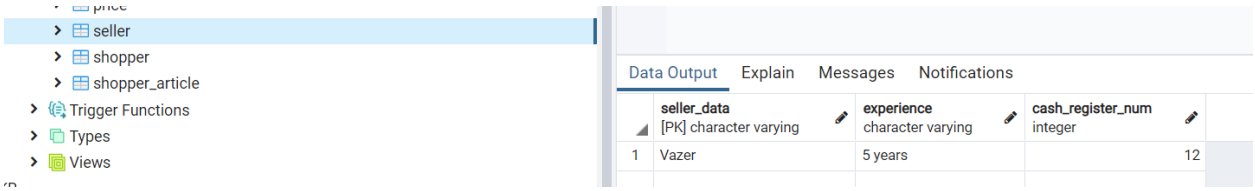




Після виконання

```
+ =+=+=+=+=+=+=+=___Program___+=+=___commands___+=+=+=+=+=+=+=+=+ +
+   random   +   insert   +   update   +   delete   +   search   +
+ =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+= +
Enter command > update
Input table name > seller
Column name > seller_data
[('Vazerko D',)]
Old name > Vazerko D
New name > Vazer
Press any key to continue . . .
```





Відповідь на вимоги до пункту №1 деталізованого завдання:

Ілюстрації обробки виняткових ситуацій (помилок) при уведенні/вилученні даних:

```
+ =+=+=+=+=+=+=+=___Program___+=+=___commands___+=+=+=+=+=+=+=+=+ +
+   random   +   insert   +   update   +   delete   +   search   +
+ =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+= +
Enter command > update
Input table name > shopper
Column name > seller_data
[('Vazerko ',)]
Old name > Boiko
New name > Popov
42703
WARNING: ПОМИЛКА:  стовпця "boiko" не існує
LINE 1: ...opper SET seller_data = Popov WHERE seller_data = Boiko )UPD...
                                                             ^
```

Вимоги до пункту №2 деталізованого завдання:

Копії екрану з фрагментами згенерованих даних таблиць:

До

	seller_data [PK] character varying	experience character varying	cash_register_num integer
1	Vazerko D	5 years	12

Після

```
+ =+=+=+=+=+=+=+=___Program___+=+=___commands___+=+=+=+=+=+=+=+=+ +
+   random   +   insert   +   update   +   delete   +   search   +
+ =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ +
Enter command > random
Input table name > seller
Enter random size > 10000
WITH rndm AS(INSERT INTO seller SELECT chr(trunc(65+random()*35000)::int),chr(trunc(65+random()*35000)::int),(trunc(65+random()*35000)::int) FROM
generate_series(1,{random_size}) RETURNING seller_data)INSERT INTO shopper SELECT seller_data FROM rndm
```

9994	澅	《	23077
9995	⧄	ᾀ	21672
9996	篤	j	12656
9997	°	ⓤ	5988
9998	愜	ঌ	30788
9999	糬	愤	33759
10000	缋	ঝ	29000

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації уведення пошукового запиту та результатів виконання запитів:

```
+   random   +   insert   +   update   +   delete   +   search   +
+ =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+= +
Enter command > search
Input quantity of attributes to search by >>> 2
Input name of the attribute number 1 to search by >>> seller_data
Input name of the attribute number 2 to search by >>> cash_register_num
['seller_data', 'cash_register_num']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'seller_data
 INTERSECT ALL SELECT table_name FROM information_schema.columns WHERE information_schema.columns.column_name LIKE 'cash_registe
r_num'
['character varying', 'integer']
Input string for seller_data to search by >>> Vazerko D
Enter left limit for cash_register_num8
Enter right limit for cash_register_num15
[('Vazerko D', '5 years', 12)]
Time:0.014961957931518555 seconds
Press any key to continue . . .
```

Вимоги до пункту №4 деталізованого завдання:

Ілюстрації програмного коду з репозиторію Git: