



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-91

Бубнов Ілля

Перевірів:

Київ – 2021

*Загальне завдання роботи полягає в такому:*

1. Реалізувати функції внесення, редагування та видалення 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/видалення 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 **видалення** 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.

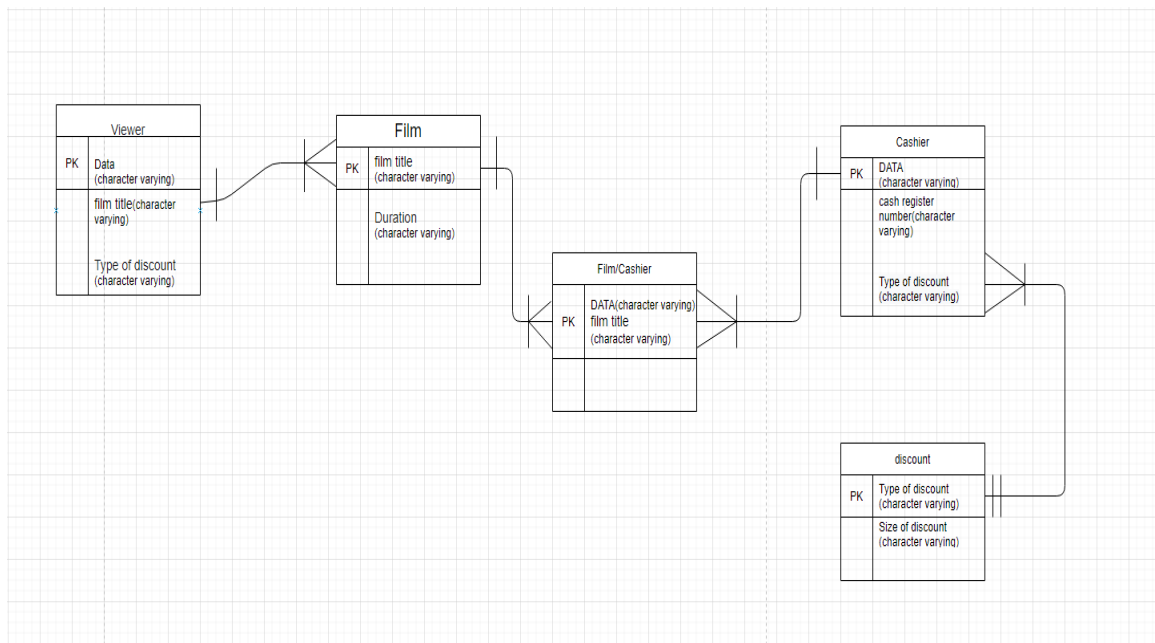
Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (#foreign key).

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 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:

```

Press 1 if you want to ADD
Press 2 if you want to DELETE
Press 3 if you want to UPDATE
Press 4 if you want to SEARCH
Press 5 if you want to RANDOM
Enter command --> 1
Enter table name --> WRONG_TABLE_NAME_FOR_TEST
Error: wrong table name

```

Ілюстрації валідації даних при введенні користувачем:

```

Press 1 if you want to ADD
Press 2 if you want to DELETE
Press 3 if you want to UPDATE
Press 4 if you want to SEARCH
Press 5 if you want to RANDOM
Enter command --> 3
Enter table name --> viewer
Film and Viewer connection 1:N , changes in the "film_title" column , touch both tables -->
Film column --> ['tilm_title', 'duration']
Viewer column --> ['data', 'film_title']
Enter column name --> WRONG_COLUMN_FOR_TEST
42703
WARNING:Error ОШИБКА: столбец "wrong_column" не существует
LINE 1: SELECT WRONG_COLUMN_FOR_TEST FROM viewer

```

**Вимоги до пункту №2 деталізованого завдання:**

Меню генерації:

```

Press 1 if you want to ADD
Press 2 if you want to DELETE
Press 3 if you want to UPDATE
Press 4 if you want to SEARCH
Press 5 if you want to RANDOM
Enter command --> 5
Enter table name --> viewer
Film and Viewer connection 1:N , changes in the "film_title" column , touch both tables -->
Film column --> ['tilm_title', 'duration']
Viewer column --> ['data', 'film_title']
Input how much random you need3

```

Копії екрану з фрагментами згенерованих даних таблиць:

Data Output	Explain	Messages	Notifications
data1	cat		
character varying	character varying		
9977	端	端	
9978	端	端	
9979	端	端	
9980	端	端	
9981	端	端	
9982	端	端	
9983	端	端	
9984	端	端	
9985	端	端	
9986	端	端	
9987	端	端	
9988	端	端	
9989	端	端	
9990	端	端	
9991	端	端	
9992	端	端	
9993	端	端	
9994	端	端	
9995	端	端	
9996	端	端	
9997	端	端	
9998	端	端	
9999	端	端	
10000	端	端	

Копії SQL запитів, що ілюструють генерацію при визначених вхідних параметрах:

```

Press 1 if you want to ADD
Press 2 if you want to DELETE
Press 3 if you want to UPDATE
Press 4 if you want to SEARCH
Press 5 if you want to RANDOM
Enter command --> 5
Enter table name --> viewer
Film and Viewer connection 1:N , changes in the "film_title" column , touch both tables -->
Film column --> ['film_title', 'duration']
Viewer column --> ['data', 'film_title']
Input how much random you need3
WITH table_m AS(INSERT INTO viewer SELECT chr(trunc(65+random()*500)::int), chr(trunc(65 + random()*500)::int) FROM generate_series(1,3) RETURNING film_title)INSERT INTO
film SELECT film_title FROM table_m

```

## Вимоги до пункту №3 деталізованого завдання:

Ілюстрації уведення пошукового запиту та результатів виконання запитів:

```

C:\Users\Boss\Desktop\БД\lab2\final>python control_func.py
Press 1 if you want to ADD
Press 2 if you want to DELETE
Press 3 if you want to UPDATE
Press 4 if you want to SEARCH
Press 5 if you want to RANDOM
Enter command --> 4
Input quantity of attributes to search by >>> 1
Input name of the attribute number 1 to search by >>> film_title
['film_title']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'film_title'
['viewer', 'film']
['character varying', 'character varying']
Input string for film_title to search by >>> Shrek2
[('Ivanov Ivan', 'Shrek2'), ('Shrek2', '1h21m')]
Time for search is: 0.0029993057250976562 sec

```

```

C:\Users\Boss\Desktop\БД\lab2\final>python control_func.py
Press 1 if you want to ADD
Press 2 if you want to DELETE
Press 3 if you want to UPDATE
Press 4 if you want to SEARCH
Press 5 if you want to RANDOM
Enter command --> 4
Input quantity of attributes to search by >>> 2
Input name of the attribute number 1 to search by >>> film_title
Input name of the attribute number 2 to search by >>> duration
['film_title', 'duration']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'film_title' INTERSECT ALL SELECT table_name FROM infor
mation_schema.columns WHERE information_schema.columns.column_name LIKE 'duration'
['character varying', 'character varying']
Input string for film_title to search by >>> Shrek2
Input string for duration to search by >>> 1h21m
[('Shrek2', '1h21m')]
Time for search is: 0.00299835205078125 sec

```

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

```

Press 1 if you want to ADD
Press 2 if you want to DELETE
Press 3 if you want to UPDATE
Press 4 if you want to SEARCH
Press 5 if you want to RANDOM
Enter command --> 4
Input quantity of attributes to search by >>> 1
Input name of the attribute number 1 to search by >>> type_of_discount
['type_of_discount']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'type_of_discount'
['character varying', 'character varying']
Input string for type_of_discount to search by >>> Student
[('Student'), ('Student'),]
Time for search is: 0.003998279571533203 sec
SELECT type_of_discount FROM discount WHERE type_of_discount LIKE 'Student' UNION ALL SELECT type_of_discount FROM cashier WHERE type_of_discount LIKE 'Student'
C:\Users\Boss\Desktop\БД\lab2\

```

## Вимоги до 4-го пункту:


main

database\_discpl / lab2 /

Go to file







Add file

...

 illiabuba lab1 lab2

60e691e 15 seconds ago [History](#)

..

 .DS_Store	lab1 lab2	15 seconds ago
 Bubnov_Illia_Lab2.docx	lab1 lab2	15 seconds ago
 SCHEMA.png	lab1 lab2	15 seconds ago
 control_func.py	lab1 lab2	15 seconds ago
 model.py	lab1 lab2	15 seconds ago
 view.py	lab1 lab2	15 seconds ago