



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

Чернявський М.Р.

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

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).

The diagram illustrates the following tables and their attributes:

- book**:
 - PK name_book(char)
 - author(char)
- book_fund**:
 - PK id_of_librarian(int)
 - book_numebr(int)
- reader**:
 - PK reader_id(int)
 - data(char)
- librarian**:
 - PK id_of_librarian(int)
 - data(char)
 - cashbox_number(int)

The relationships are as follows:

- SEARCH**: A many-to-one relationship between **book** (N) and **book_fund** (1).
- DELIVERY**: A many-to-one relationship between **book_fund** (1) and **reader** (N).
- librarian**: A table that shares the primary key **id_of_librarian(int)** with **book_fund**, indicating a one-to-one relationship.

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

```
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Enter command : 7
You have to enter the command from 1 to 5 ERROR
```

Ілюстрації валідації даних при введенні користувачем:

```
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Enter command : 1
Your table name: book , librarian , book_fund , reader
Enter table name book
Enter column name WRONGCOLUMN
42703
WARNING:Error ОШИБКА: столбец "wrongcolumn" не существует
LINE 1: SELECT WRONGCOLUMN FROM book
          ^
```

Вимоги до пункту №2 деталізованого завдання:

Меню генерації:

```
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Enter command : 4
Your table name: book , librarian , book_fund , reader
Enter table name book_fund
Enter value: 3
WITH table_m AS(INSERT INTO book_fund SELECT trunc(random()*1000)::int, trunc(random()*1000)::int FROM generate_series(1,3) RETURNING id_of_librarian)INSERT INTO librarian SELECT
id_of_librarian FROM table_m
```

Копії екрану з фрагментами згенерованих даних таблиць:

9977	熾	熾
9978	熾	熾
9979	熾	熾
9980	熾	熾
9981	熾	熾
9982	熾	熾
9983	熾	熾
9984	熾	熾
9985	熾	熾
9986	熾	熾
9987	熾	熾
9988	熾	熾
9989	熾	熾
9990	熾	熾
9991	熾	熾
9992	熾	熾
9993	熾	熾
9994	熾	熾
9995	熾	熾
9996	熾	熾
9997	熾	熾
9998	熾	熾
9999	熾	熾
10000	熾	熾

Копії SQLзапитів, що ілюструють генерацію при визначених вхідних параметрах:

```

Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Enter command : 4
Your table name: book , librarian , book_fund , reader
Enter table name book_fund
Enter value: 3
WITH table_m AS(INSERT INTO book_fund SELECT trunc(random()*1000)::int, trunc(random()*1000)::int FROM generate_series(1,3) RETURNING id_of_librarian)INSERT INTO librarian SELECT
id_of_librarian FROM table_m

```

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації введення пошукового запиту та результатів виконання запитів:

```

Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Enter command : 5
Input quantity of attributes to search by >>> 1
Input name of the attribute number 1 to search by >>> cashbox_number
['cashbox_number']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name
LIKE 'cashbox_number'
['integer']
Enter left limit1
Enter right limit35
[(1, 'Ivanov Ivan', 34), (2, 'Andiy Olegovich', 18), (3, 'David Nikolaevich', 22), (4, 'Max Antonov', 13)]
Time:0.00200653076171875 seconds

```

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

```

Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Enter command : 5
Input quantity of attributes to search by >>> 1
Input name of the attribute number 1 to search by >>> id_of_librarian
['id_of_librarian']

col_names_str: SELECT table_name FROM INFORMATION_SCHEMA.COLUMNS WHERE information_schema.columns.column_name LIKE 'id_of
librarian'
['integer', 'integer']
Enter left limit2
Enter right limit4
[(2,), (3,), (2,), (3,)]
SELECT id_of_librarian FROM librarian WHERE id_of_librarian>='2' AND id_of_librarian<'4' UNION ALL SELECT id_of_librarian
FROM book_fund WHERE id_of_librarian>='2' AND id_of_librarian<'4'
Time:0.0029921531677246094 seconds

```

Вимоги до пункту №4 деталізованого завдання:

Ілюстрації програмного коду з репозиторію Git: