



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

Бугайов Д.С.

Перевірив:

Київ – 2020

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).

- 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 уведенні/вилучення даних:

```
Main menu:
1. Insert row
2. Delete row
3. Update row
4. Generate random rows
5. Search rows
6. Quit
>> 1

Choose the table you want to insert the row:

trolleybus
depot
route
what_services
service
where_serviced
phonebook

>> SOME_UNEXISTING_TABLE

ERROR: No table with the name "SOME_UNEXISTING_TABLE"

Main menu:
1. Insert row
```

Ілюстрації валідації даних при введенні користувачем:

```
Main menu:
1. Insert row
2. Delete row
3. Update row
4. Generate random rows
5. Search rows
6. Quit
>> 5

Input quantity of attributes to search by >> 300

Input name of the attribute number 1 to search by >> id

Input left integer limit of id to search by >> TEST

Input right integer limit of id to search by >>
ERROR: Wrong input, must be integer above 0
```

Вимоги до пункту №2 деталізованого завдання:

Меню генерації:

```
Main menu:
1. Insert row
2. Delete row
3. Update row
4. Generate random rows
5. Search rows
6. Quit
>> 4

Choose the table you want to generate rows:

trolleybus
where_serviced
depot
service
route
what_services
phonebook

>> route

Input quantity of rows to generate >> 2

Generating 2 rows in route successful

Main menu:
1. Insert row
2. Delete row
3. Update row
4. Generate random rows
5. Search rows
6. Quit
>>
```

копії екрану з фрагментами згенерованих даних таблиць:

Q: <Filter Criteria>		
	name	address
99968	ㄣ琚号	鰐□塚
99969	ㄣ濫ㄣ	ㄣ鰐□
99970	ㄣ蟬ㄣ	ㄣ挿○
99971	ㄣ諫棋	ㄣ欸欸○
99972	ㄣ穉	ㄣ欸欸甘
99973	ㄣ欸	ㄣ湾
99974	ㄣ悉經	ㄣ欸欸
99975	ㄣ嬌□	ㄣ□ ^{2A}
99976	ㄣ ^ㄣ	ㄣ割密
99977	ㄣ瘰	ㄣ糸 ^ㄣ
99978	ㄣ岷□	ㄣ ^ㄣ 限
99979	ㄣ箐	ㄣ塢季
99980	ㄣ辰陸	ㄣ ^ㄣ
99981	ㄣ叠□	ㄣ欸許
99982	ㄣ ^ㄣ	ㄣ忘湮
99983	ㄣ ^ㄣ	ㄣ欸體
99984	ㄣ ^ㄣ	ㄣ□
99985	ㄣ指	ㄣ□
99986	ㄣ ^ㄣ	ㄣ(ㄣ)
99987	ㄣ□	ㄣ ^ㄣ
99988	ㄣ ^ㄣ	ㄣ□
99989	ㄣ	ㄣ□
99990	ㄣ	ㄣ ^ㄣ
99991	ㄣ ^ㄣ	ㄣ ^ㄣ
99992	ㄣ	ㄣ ^ㄣ
99993	ㄣ ^ㄣ	ㄣ ^ㄣ
99994	ㄣ ^ㄣ	ㄣ ^ㄣ
99995	ㄣ ^ㄣ	ㄣ ^ㄣ
99996	ㄣ ^ㄣ	ㄣ ^ㄣ
99997	ㄣ ^ㄣ	ㄣ ^ㄣ
99998	ㄣ ^ㄣ	ㄣ ^ㄣ
99999	ㄣ ^ㄣ	ㄣ ^ㄣ
100000	ㄣ ^ㄣ	ㄣ ^ㄣ

*для генерації великої кількості унікальних ключів було взято широкий алфавіт

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

```
Main menu:
1. Insert row
2. Delete row
3. Update row
4. Generate random rows
5. Search rows
6. Quit
>> 4

Choose the table you want to generate rows:

trolleybus
where_serviced
depot
service
route
what_services
phonebook

>> depot

Input quantity of rows to generate >> 4

INSERT INTO depot SELECT  chr(trunc(65 + random()
*50000)::int) || chr(trunc(65 + random()*50000)
::int) || chr(trunc(65 + random()*50000)::int),
chr(trunc(65 + random()*50000)::int) ||
chr(trunc(65 + random()*50000)::int) ||
chr(trunc(65 + random()*50000)::int) FROM
generate_series(1,4)

Generating 4 rows in depot successful

Main menu:
1. Insert row
2. Delete row
3. Update row
4. Generate random rows
```

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації введення пошукового запиту та результатів виконання запитів:

```
Main menu:
1. Insert row
2. Delete row
3. Update row
4. Generate random rows
5. Search rows
6. Quit
>> 5

Input quantity of attributes to search by >> 1

Input name of the attribute number 1 to search by >> trolleybus_id

Input left integer limit of trolleybus_id to search by >> 1

Input right integer limit of trolleybus_id to search by >> 9999

Table: "where_serviced"

trolleybus_id      depot_name
1111               Trolleybus repair-operational depot #2
2345               Trolleybus repair-operational depot #1
6363               Kurenivs'ke Trolleybusne Depo

Table: "service"

trolleybus_id      next
1111               2020-11-28
2345               2020-09-25
6363               2020-09-07

Time taken: 11[ms]
```


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

```
Main menu:
1. Insert row
2. Delete row
3. Update row
4. Generate random rows
5. Search rows
6. Quit
>> 5

Input quantity of attributes to search by >> 2

Input name of the attribute number 1 to search by >> trolleybus_id

Input left integer limit of trolleybus_id to search by >> 1

Input right integer limit of trolleybus_id to search by >> 9999

Input name of the attribute number 2 to search by >> type

Input string for type to search by >> long

DECLARE myportal CURSOR FOR SELECT trolleybus_id FROM what_services
WHERE trolleybus_id >= '1' AND trolleybus_id < '9999' AND type
LIKE 'long' UNION ALL SELECT trolleybus_id FROM service WHERE
trolleybus_id >= '1' AND trolleybus_id < '9999' AND type LIKE
'long' UNION ALL SELECT trolleybus_id FROM where_serviced WHERE
trolleybus_id >= '1' AND trolleybus_id < '9999' AND type LIKE
'long'

trolleybus_id

6363
2345
1111
6363
2345
1111

Time taken: 1[ms]
```

Вимоги до пункту №4 деталізованого завдання:

Ілюстрації програмного коду з репозиторію Git:

BuhaiovDmytro / Studies

Unwatch2Star0Fork1

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

...

master

Studies / Databases_and_controls / Lab2 /

Go to file

Add file

BuhaiovDmytro

Merge branch 'master' of github.com:BuhaiovDmytro/Studies

10 minutes ago

History

..		
CMakeLists.txt	Renamed directory Databases and controls	1 hour ago
README.md	Update README.md	1 hour ago
db_operations.cpp	Renamed directory Databases and controls	1 hour ago
db_operations.h	Renamed directory Databases and controls	1 hour ago
main.cpp	Renamed directory Databases and controls	1 hour ago
psql_getters.cpp	Renamed directory Databases and controls	1 hour ago
psql_getters.h	Renamed directory Databases and controls	1 hour ago
submenu.cpp	Lab2: Fixed Generation menu	10 minutes ago
submenu.h	Renamed directory Databases and controls	1 hour ago
trolleybus_schema.png	Renamed directory Databases and controls	1 hour ago