

**Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”**

**Факультет прикладної математики
Кафедра системного програмування і спеціалізованих
комп'ютерних систем**

ЛАБОРАТОРНА РОБОТА № 2

з дисципліни

“Бази даних і засоби управління”

Група: KB-03

Виконав: Палажченко Максим

Оцінка:

Київ – 2022

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.

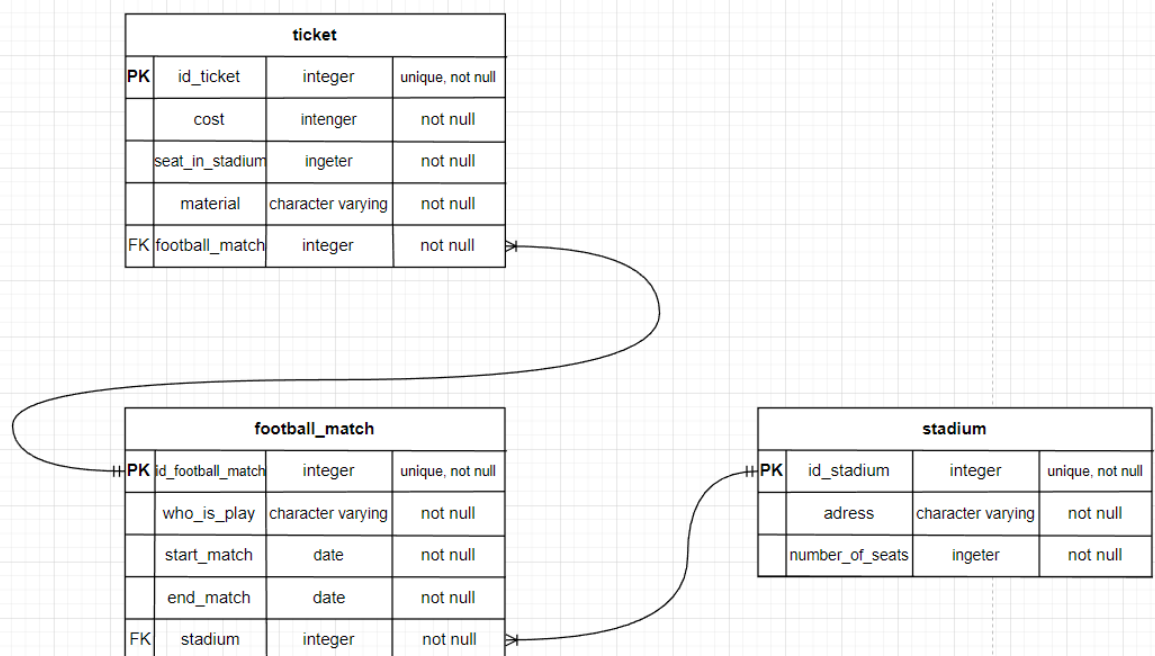
Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності foreign key).

- 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>)



Сутність	Атрибут	Опис Атрибути	Тип	Обмеження
ticket	id	unique identifier	integer	not null unique
	cost	ціна в ₴	integer	not null
	seat_in_the_stadium	Місце на стадіоні для вболівальник	integer	not null

	material	Матеріал з якого виготовлено квиток	character varying	not null
	football_match	посилання на характеристики	integer	not null
football_match	id	unique identifier	integer	not null unique
	who_is_play	які команди грають?	character varying	not null
	start_match	час початку матчу	date	not null
	end_match	час кінця матчу	date	not null
	stadium	посилання на характеристики	integer	not null
stadium	id	unique identifier	integer	not null unique
	adress	адреса стадіону	character varying	not null
	number_of_seats	кількість місць	integer	not null

Опис функціоналу меню:

Update – оновлення даних в певній колонці,

Add – додання нової колонки,

Delete – видалення нової колонки

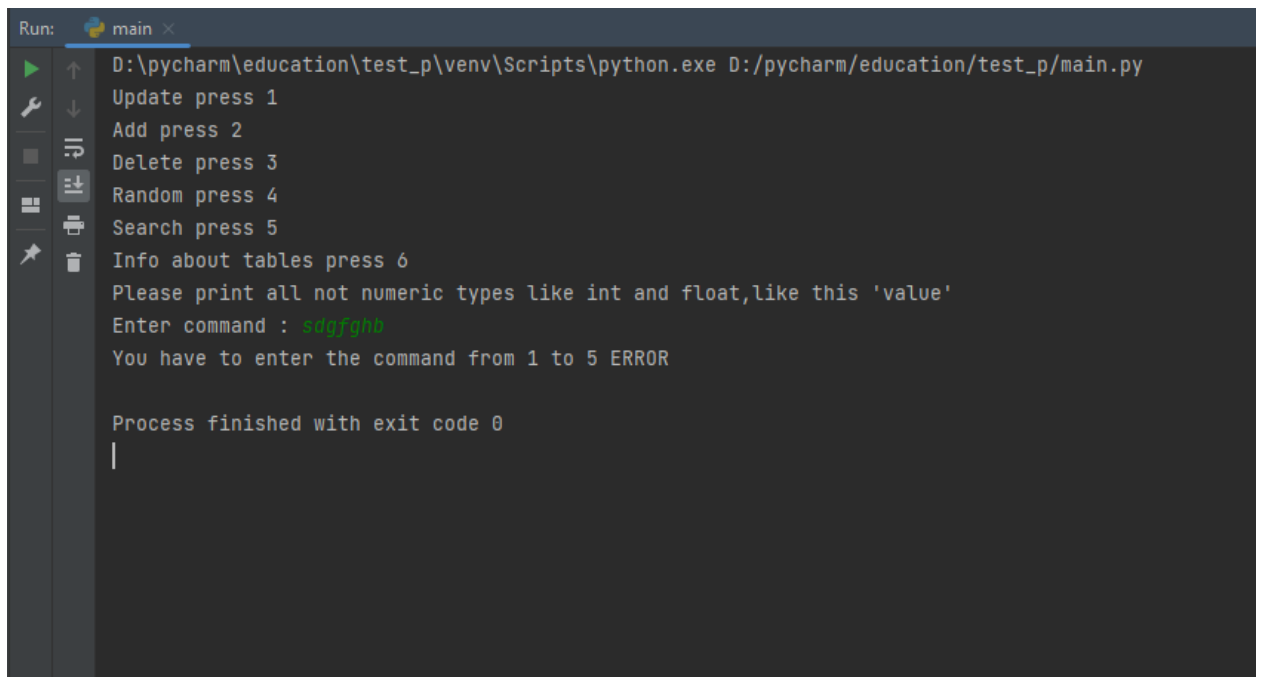
Random - рандомна генерація нових значень для колонок на n раз

Search – пошук по базовим даним якоїсь колонки

Info about tables – повна інформація по бд

Відповідь на вимоги до пункту №1 деталізованого завдання:

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



```
Run: main x
D:\pycharm\education\test_p\venv\Scripts\python.exe D:/pycharm/education/test_p/main.py
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command : sdgfgghb
You have to enter the command from 1 to 5 ERROR

Process finished with exit code 0
|
```

Видалення даних

Данні стадіона до видалення:

	id_stadium [PK] integer	adress character varying	number_of_seats integer
1	1	U	981
2	2	R	461
3	3	X	546
4	4	P	864
5	5	S	504
6	6	P	306
7	7	G	630
8	8	E	530
9	9	L	648
10	10	Q	710

Данні матчі до видалення:

Data output Messages Notifications						
	id_football_match [PK] integer	who_is_play character varying	start_match date	end_match date	stadium integer	
1	1	B	1973-12-11	1973-12-11	1	
2	2	H	1988-04-28	1988-04-28	3	
3	3	B	1973-10-19	1973-10-19	9	

```
D:\pycharm\education\test_p\venv\Scripts\python.exe D:/pycharm/education/test_p/main.py
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command :
Your table_name name: stadium, ticket, football_match
Enter table_name name
Enter column attribute,based on which you want to change this field like 'Column_name = 'value'' and press Enter,when you want to stop enter columns press '-' and press Enter:
Enter column attribute,based on which you want to change this field like 'Column_name = 'value'' and press Enter,when you want to stop enter columns press '-' and press Enter:
delete FROM stadium WHERE id_stadium = 1

Process finished with exit code 0
```

Стадіони після видалення:

	id_stadium [PK] integer	adress character varying	number_of_seats integer	
1	2	R	461	
2	3	X	546	
3	4	P	864	
4	5	S	504	
5	6	P	306	
6	7	G	630	
7	8	E	530	
8	9	L	648	
9	10	Q	710	

Матчі після видалення:

	id_football_match [PK] integer	who_is_play character varying	start_match date	end_match date	stadium integer	
1	2	H	1988-04-28	1988-04-28	3	
2	3	B	1973-10-19	1973-10-19	9	

Додавання даних:

Data outputMessagesNotifications

	id_ticket [PK] integer	cost integer	seat_in_the_stadium integer	material character varying (100)	football_match integer

D:\pycharm\education\test_p\venv\Scripts\python.exe D:/pycharm/education/test_p/main.py
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command : 2
Your table_name name: stadium, ticket, football_match
Enter table_name name ticket
cost:100
seat_in_the_stadium:1
material:ugleplastic
football_match:2
INSERT INTO ticket (cost,seat_in_the_stadium,material,football_match) VALUES ('100','1','ugleplastic','2')

Process finished with exit code 0

	id_ticket [PK] integer	cost integer	seat_in_the_stadium integer	material character varying (100)	football_match integer
1	2	100	1	ugleplastic	2

Вимоги до пункту №2 деталізованого завдання:
Меню генерації:

Data outputMessagesNotifications

	id_ticket [PK] integer	cost integer	seat_in_the_stadium integer	material character varying (100)	football_match integer
1	2	100	1	ugleplastic	2

D:\pycharm\education\test_p\venv\Scripts\python.exe D:/pycharm/education/test_p/main.py
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command : 4
Your table_name name: stadium, ticket, football_match
Enter table_name name ticket
Enter value:
INSERT INTO ticket (cost,seat_in_the_stadium,material,football_match) SELECT trunc(random()*1000)::int,trunc(random()*1000)::int,chr(trunc(65+random()*25)::int),id_football_match FROM football_match order by random() limit 1

Data output Messages Notifications						
	id_ticket [PK] integer	cost integer	seat_in_the_stadium integer	material character varying (100)	football_match integer	
1	2	100	1	ugleplastic	2	
2	3	235	235	F	2	
3	4	244	244	G	2	
4	5	251	251	G	2	
5	6	37	37	A	3	
6	7	452	452	L	3	
7	8	17	17	A	3	
8	9	216	216	F	2	
9	10	327	327	I	2	
10	11	867	867	V	3	
11	12	423	423	K	3	
12	13	524	524	N	3	
13	14	316	316	H	3	
14	15	513	513	M	2	
15	16	774	774	T	3	
16	17	556	556	N	2	

Вимоги до пункту №3 деталізованого завдання:

Ілюстрації введення пошукового запиту та результатів виконання запитів:

```

D:\pycharm\education\test_p\venv\Scripts\python.exe D:/pycharm/education/test_p/main.py
Update press 1
Add press 2
Delete press 3
Random press 4
Search press 5
Info about tables press 6
Please print all not numeric types like int and float,like this 'value'
Enter command :
Choose presearch scenario which you want:
Enter column where you want to search values separated by '|':id_stadium
please input range int numbers for id_stadium separated by '|' 1 10
SELECT * FROM football_match,ticket,stadium WHERE id_stadium > -1 AND id_stadium < 10
[(2, 'H', datetime.date(1988, 4, 28), datetime.date(1988, 4, 28), 3, 2, 100, 1, 'ugleplastic', 2, 2, 'R', 461), (2, 'H', datetime.date(1988, 4, 28), datetime.date(1988, 4, 28), 3, 2, 100, 1,
'ugleplastic', 2, 4, 'P', 864), (2, 'H', datetime.date(1988, 4, 28), datetime.date(1988, 4, 28), 3, 2, 100, 1, 'ugleplastic', 2, 5, 'S', 504), (2, 'H', datetime.date(1988, 4, 28), datetime
datetime.date(1988, 4, 28), 3, 2, 100, 1, 'ugleplastic', 2, 7, 'I', 119), (2, 'H', datetime.date(1988, 4, 28), datetime.date(1988, 4, 28), 3, 2, 100, 1, 'ugleplastic', 2, 8, 'I', 513), (2, 'H',

```

Вимоги до пункту №4 деталізованого завдання:

Ілюстрації програмного коду з репозиторію Git:

main.py:

```

from model import *
from view import *
import sys

def request():

```

```

input_command = comand_identification()

if (input_command == '1'):
    table = table_name()
    str_of_columns = input_columns_str_upd()
    str_of_updating_column = take_inf_about_param(str_of_columns)
    str_of_based_column = take_inf_based()
    update(table, str_of_updating_column, str_of_based_column)
elif (input_command == '2'):
    table = table_name()
    inf = take_inf_for_adding(table)
    add_information(table, inf)
elif (input_command == '3'):
    table = table_name()
    str_of_based_column = take_inf_based()
    delete(table, str_of_based_column)
elif (input_command == '4'):
    table = table_name()
    data = Data()
    random(table, data)
elif (input_command == '5'):
    scenario = choose_scenario_search()
    str_of_columns = input_columns_str_search()
    dict_of_searching_var = take_searching_rows(str_of_columns)
    list_of_searching = search(scenario, dict_of_searching_var)
    print_searching_values(list_of_searching)
elif (input_command == '6'):
    print_info(information())
elif (input_command is not str(range(6))):
    comndErr()
    sys.exit()

```

menu()

request()

my_inf(словник для інформації)

```

parametr = {
    "stadium": [("id_stadium", "int"), ("adress", "str"),
    ("number_of_seats", "int")],
    "football_match": [("id_football_match", "int"),
    ("who_is_play", "str"), ("start_match", "time"), ("end_match", "time"),
    ("stadium", "int")],
    "ticket": [("id_ticket", "int"), ("cost", "int"),
    ("seat_in_the_stadium", "int"), ("material", "str"), ("football_match",
    "int")]
}

my_inf = {
    "random":
    {
        "stadium": "INSERT INTO stadium (parametr) SELECT
chr(trunc(65+random()*25)::int), trunc(random()*1000)::int",
        "football_match": "INSERT INTO football_match (parametr) SELECT
chr(trunc(65+random()*25)::int), "
        "timestamp '1970-01-10 20:00:00'+ random() *
(timestamp '2033-01-20 20:00:00' - timestamp '1970-01-10 10:00:00'), "
        "timestamp '1970-01-10 20:00:00'+ random() *
(timestamp '2033-01-20 20:00:00' - timestamp '1970-01-10 10:00:00'), "
        "id_stadium From stadium order by random()
limit 1",

```

```

        "ticket": "INSERT INTO ticket (params) SELECT
trunc(random()*1000)::int, trunc(random()*1000)::int, "
        "chr(trunc(65+random()*25)::int), id_football_match
FROM football_match order by random() limit 1"
    },
    "delete": "delete FROM table_name WHERE str_of_based_column",
    "add_inf": "INSERT INTO table_name (params) VALUES (values)",
    "upd_inf": "UPDATE table_name SET str_of_updating_column WHERE
str_of_based_column;",
    "inf": "SELECT param FROM table",
    "presearch":
    [
        "SELECT * FROM stadium, ticket WHERE ",
        "SELECT id_football_match, id_ticket, material, cost FROM
football_match, ticket WHERE ",
        "SELECT * FROM football_match, ticket, stadium WHERE "
    ]
}

```

model.py(логіка програми)

```

import psycopg2
import my_inf

def decorator_for_go_to_database(func):
    def wrapper(*args, **kwargs):
        con = psycopg2.connect(
            database="football ticket sales service",
            user="postgres",
            password="password",
            host="localhost",
            port="5432"
        )
        con.set_session(autocommit=True)
        cursor_r = con.cursor()
        some_shit = func(con, cursor_r, *args, **kwargs)
        cursor_r.close()
        con.close()
        return some_shit
    return wrapper

@decorator_for_go_to_database
def random(con, cursor_r, table_name, n):
    param = ",".join([*map(lambda x: my_inf.params[table_name][x][0],
range(1, len(my_inf.params[table_name])))])
    try:
        for i in range(int(n)):
            take_sql_string_to_two_stream(cursor_r,
my_inf.my_inf["random"][table_name].replace("params", param))
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')

@decorator_for_go_to_database
def add_information(con, cursor_r, table_name, mass):
    param = ",".join([*map(lambda x: my_inf.params[table_name][x][0],
range(1, len(my_inf.params[table_name])))])
    mass = [*map(lambda x: "{0}".format(x), mass)]
    values = ",".join(mass)
    try:

```

```

        take_sql_string_to_two_stream(cursor_r,
my_inf.my_inf["add_inf"].replace("table_name",
table_name).replace("params", param).replace("values", values))
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')

@decorator_for_go_to_database
def delete(con, cursor_r, table_name, str_of_based_column):
    try:
        take_sql_string_to_two_stream(cursor_r,
my_inf.my_inf["delete"].replace("str_of_based_column",
str_of_based_column).replace("table_name", table_name))
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')

@decorator_for_go_to_database
def search(con, cursor_r, scenario, dict_of_searching_var):
    list_of_commands = []
    for key, value in dict_of_searching_var.items():
        if(isinstance(value, list)):
            list_of_commands.append(key + " > " + value[0] + " AND " + key +
" < " + value[1])
        else:
            list_of_commands.append(key + " LIKE " + value)
    commands = ' AND '.join(list_of_commands)
    try:
        take_sql_string_to_two_stream(cursor_r,
my_inf.my_inf["presearch"][scenario] + commands)
        searching_values = cursor_r.fetchall()
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')
    else:
        return searching_values

@decorator_for_go_to_database
def update(con, cursor_r, table_name, str_of_updating_column,
str_of_based_column):
    try:
        take_sql_string_to_two_stream(cursor_r,
my_inf.my_inf["upd_inf"].replace("table_name",
table_name).replace("str_of_updating_column",
str_of_updating_column).replace("str_of_based_column", str_of_based_column))
    except psycopg2.Error as err:
        print(err.pgcode)
        print(f'WARNING:Error {err}')

@decorator_for_go_to_database
def information(con, cursor_r):
    dict_of_all_tables = {}
    try:
        for table in my_inf.params:
            dict_of_all_tables[table] = []
            flag_of_take_memory = 1
            for param in my_inf.params[table]:

                param = param[0]
                cursor_r.execute(my_inf.my_inf["inf"].replace("table",
table).replace("param", param))
                list_of_all_params = cursor_r.fetchall()

```

```

        if (flag_of_take_memory):
            for memory_take in range(len(list_of_all_params)):
                dict_of_all_tables[table].append({})
                flag_of_take_memory = 0
            for i in range(len(list_of_all_params)):
                dict_of_all_tables[table][i][param] =
list_of_all_params[i][0]
                flag_of_take_memory = 1
        except psycopg2.Error as err:
            print(err.pgcode)
            print(f'WARNING:Error {err}')
        else:
            return dict_of_all_tables

def take_sql_string_to_two_stream(cursor_r, sql_stream):
    print(sql_stream)
    cursor_r.execute(sql_stream)

```

requirements.txt(використанні ліби)

```

attrs==22.1.0
colorama==0.4.6
exceptiongroup==1.0.1
iniconfig==1.1.1
packaging==21.3
pluggy==1.0.0
psycopg2==2.9.5
pyparsing==3.0.9
tomli==2.0.1

```