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»**

**Тема**: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав:   
студент групи КВ-82

Мороз А.С.  
Перевірив:  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Київ 2020

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL  
  
  
 **Завдання**

*Загальне завдання* роботи полягає у наступному:

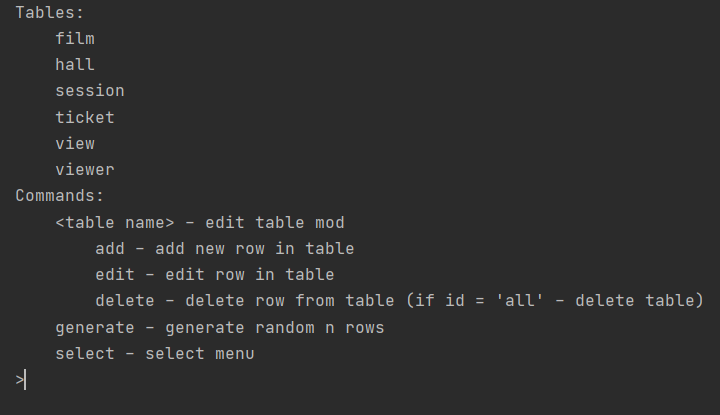
1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та 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).

**Завдання 1**

У даній програмі передбачені такі дії із сутностями бази даних:

* Внесення стовпців
* Редагування
* Видалення

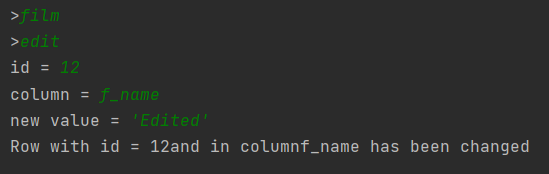
Головне меню програми:



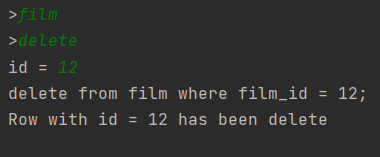
Внесення даних у таблицю:



Оновлення даних таблиці film:

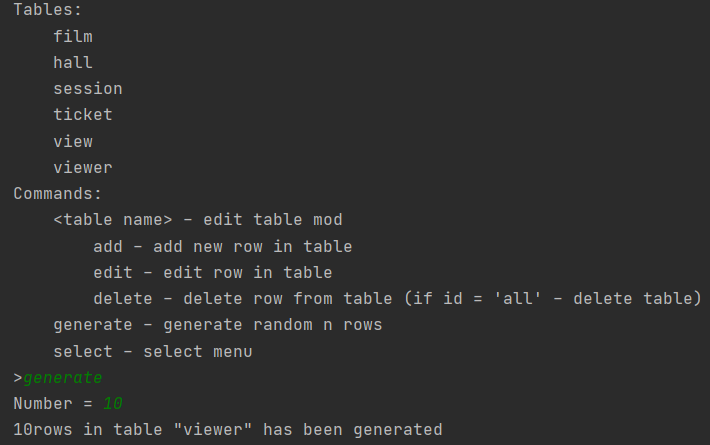


Видалення запису з таблиці film:



**Завдання 2**

Дана програма передбачає створення числових даних, рядкових даних або даних, що відповідають датам. Приклад вводу даних для генерації записів у таблиці viewer.



Результати:



Запит:

insert into viewer (v\_name, v\_age)

select substr(characters, (random()\*length(characters)+1 )::integer, 10),

(12+random()\*70)::int

from (values ('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) as symbols(characters),

generate\_series(1, 10);

**Завдання 3**

У програмі реалізовано два запиту пошуку:

1. Обрати глядача, за спеційним віком, який дивиться фільм за спеційною датою випуску
2. Обрати залу із спеційною кількістю місць і рядків

**Запит 1.**

"with usr as ( "\  
 "select \* from viewer where v\_age > 10 and v\_age < 17 "\  
 "), fm as ( "\  
 "select \* from film where (year > \'2001-01-01\') and (year < \'2005-01-01\') ) "\  
 "select \* from usr;"

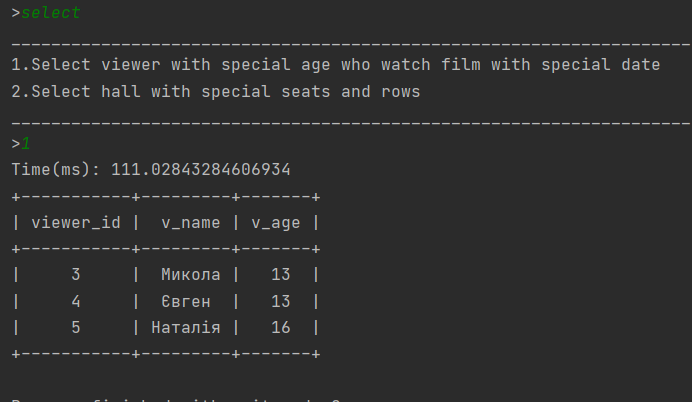
**Запит 2.**

"select \* from hall where (seats\_in\_row > 10 and row\_count < 22)"

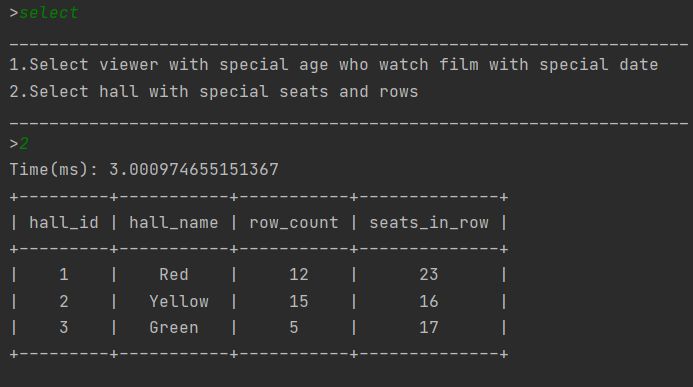
Далі наведено результати роботи програми з використанням відповідних

запитів.

Перший запит:



Другий запит:



**Завдання 4**

Код програми:

model.py

import psycopg2  
import time  
  
connection = psycopg2.connect(  
 dbname='cinema',  
 user='postgres',  
 password='3miners1creeper',  
 host='127.0.0.1'  
)  
  
cursor = connection.cursor()  
  
tables = ["film", "hall", "session", "ticket", "view", "viewer"]  
  
def create\_new\_row\_in\_table(name, args):  
 sql\_query = "insert into "  
 for el in args:  
 if el == args[0]:  
 sql\_query = sql\_query + name + " values( " + str(el) + ", "  
 elif el != args[len(args)-1]:  
 sql\_query = sql\_query + str(el) + ", "  
 else:  
 sql\_query = sql\_query + str(el) + ");"  
 try:  
 cursor.execute(sql\_query)  
 print("In table" + name + " has been aded new row")  
 except psycopg2.Error as err:  
 print(err.pgerror)  
  
def delete\_row\_from\_table(name, id):  
 sql\_query = "delete from " + name  
 if id != "all":  
 sql\_query = sql\_query + " where " + name +"\_id = " + str(id) + ";"  
 else:  
 sql\_query = sql\_query + ";"  
 print(sql\_query)  
 try:  
 cursor.execute(sql\_query)  
 print("Row with id = " + id + " has been delete")  
 except psycopg2.Error as err:  
 print(err.pgerror)

def edit\_row\_in\_table(name, id, column, newvalue):  
 if id == 0:  
 sql\_query\  
 = "update " + name + " set " + column + " = " + newvalue + ";"  
 else:  
 sql\_query = "update " + name + " set " + column + " = " + newvalue + " where " + name+"\_id = " + str(id) + ";"  
 try:  
 cursor.execute(sql\_query)  
 print("Row with id = " + id + "and in column" + column + " has been changed")  
 except psycopg2.Error as err:  
 print(err.pgerror)  
  
def generate\_random\_rows():  
 num = input("Number = ")  
 sql\_query = "insert into viewer (v\_name, v\_age) " \  
 "select substr(characters, (random()\*length(characters)+1 )::integer, 10), " \  
 "(12+random()\*70)::int " \  
 "from (values ('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) as symbols(characters), " \  
 "generate\_series(1," + num + ");"  
 try:  
 cursor.execute(sql\_query)  
 print(num + "rows in table \"viewer\" has been generated")  
 except psycopg2.Error as err:  
 print(err.pgerror)  
  
def select(num):  
 sql\_query = " "  
 start\_time = time.time()  
 if num == '1':  
 sql\_query = "with usr as ( "\  
 "select \* from viewer where v\_age > 10 and v\_age < 17 "\  
 "), fm as ( "\  
 "select \* from film where (year > \'2001-01-01\') and (year < \'2005-01-01\') ) "\  
 "select \* from usr;"  
 else:  
 sql\_query = "select \* from hall where (seats\_in\_row > 10 and row\_count < 22)"  
 try:  
 cursor.execute(sql\_query)  
 print("Time(ms):", 1000\*(time.time() - start\_time))  
 return cursor.fetchall()  
 except psycopg2.Error as err:  
 print(err.pgerror)

view.py:

import model  
from prettytable import PrettyTable  
  
def show\_main\_screen():  
 print("Tables:")  
 for elem in model.tables:  
 print("\t"+elem)  
 show\_commands\_menu()  
  
def show\_commands\_menu():  
 print("Commands:")  
 print("\t<table name> - edit table mod")  
 print("\t\tadd - add new row in table")  
 print("\t\tedit - edit row in table")  
 print("\t\tdelete - delete row from table (if id = 'all' - delete table)")  
 print("\tgenerate - generate random n rows")  
 print("\tselect - select menu")  
  
def show\_select\_menu():  
 print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  
 print("1.Select viewer with special age who watch film with special date")  
 print("2.Select hall with special seats and rows")  
 print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  
  
  
def show\_result\_of\_select\_first(rows):  
 th = ["viewer\_id", "v\_name", "v\_age"]  
 table = PrettyTable(th)  
 for elem in rows:  
 td = [elem[0],elem[1],elem[2]]  
 table.add\_row(td)  
 print(table)  
  
def show\_result\_of\_select\_secound(rows):  
 th = ["hall\_id", "hall\_name", "row\_count", "seats\_in\_row"]  
 table = PrettyTable(th)  
 for elem in rows:  
 td = [elem[0], elem[1], elem[2], elem[3]]  
 table.add\_row(td)  
 print(table)

controller.py:

import view  
table\_size = {  
 "film" or "hall" or "session" or "ticket": 4,  
 "view" or "viewer": 3,  
}  
heads = {  
 "film": ["film\_id", "f\_name", "f\_genre", "f\_date"],  
 "hall": ["hall\_id", "hall\_name", "row\_count", "seats\_in\_row"],  
 "session": ["session\_id", "time", "s\_film\_id", "s\_hall\_id"],  
 "ticket": ["ticket\_id", "row", "seat", "t\_session\_id"],  
 "view": ["view\_id", "v\_viewer\_id", "v\_film\_id"],  
 "viewer": ["viewer\_id", "v\_name", "v\_age"]  
}  
  
def try\_mode\_command():  
 command = input(">")  
 if command == search\_table(command):  
 table\_eit\_mode(command)  
 elif command == "generate":  
 view.model.generate\_random\_rows()  
 elif command == "select":  
 \_select()  
  
def table\_eit\_mode(table\_name):  
 command = input(">")  
 if command == "add":  
 \_add(table\_name)  
 elif command == "edit":  
 \_edit(table\_name)  
 elif command == "delete":  
 \_delete(table\_name)  
  
def \_add(table\_name):  
 array = []  
 list = heads.get(table\_name)  
 i = 0  
 while i < table\_size.get(table\_name):  
 array.append(input(str(list)+"\n>"))  
 i = i + 1  
 view.model.create\_new\_row\_in\_table(table\_name, array)  
  
def \_edit(table\_name):  
 id = input("id = ")  
 column = input("column = ")  
 newvalue = input("new value = ")  
 view.model.edit\_row\_in\_table(table\_name, id, column, newvalue)  
  
def \_delete(table\_name):  
 id = input("id = ")  
 view.model.delete\_row\_from\_table(table\_name, id)  
  
def search\_table(command):  
 for name in view.model.tables:  
 if command == name:  
 return name

def \_select():  
 view.show\_select\_menu()  
 command = input(">")  
 if command == '1':  
 view.show\_result\_of\_select\_first(view.model.select(command))  
 elif command == '2':  
 view.show\_result\_of\_select\_secound(view.model.select(command))

main.py:

import controller  
  
controller.view.show\_main\_screen()  
controller.try\_mode\_command()  
controller.view.model.connection.commit()  
controller.view.model.connection.close()

Посилання на репозиторій GitHub:  
  
<https://github.com/RemusArtemus/db_labs>