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

Нігматшаєв М. А.

Перевірив:

Київ – 2020

Завдання:

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).

Посилання на репозиторій:

https://github.com/mnihmatshaiev/db-and-ms

Структура бази даних:

Пункту №:1

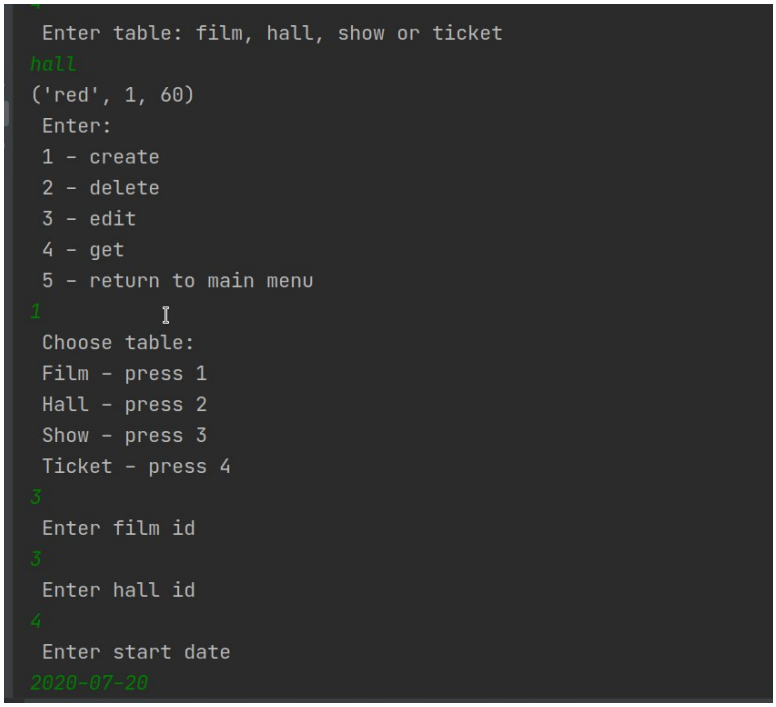
Ілюстрації обробки виняткових ситуацій (помилок) при уведенні/вилучення даних:

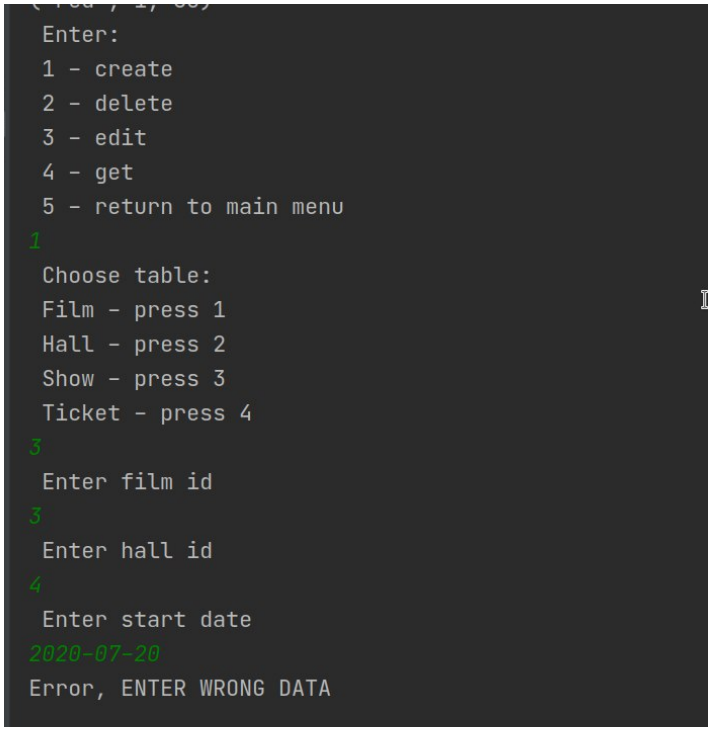
При додаванні даних(додавання вже існуючих ключів первинних ключів)

```
 Enter table: film, hall, show or ticket
ticket
(1, 3, 2)
(1, 2, 2)
(1, 3, 3)
(1, 22, 1)
 Enter:
 1 - create
 2 - delete
 3 - edit
 4 - get
 5 - return to main menu

1
 Choose table:
 Film - press 1
 Hall - press 2
 Show - press 3
 Ticket - press 4
4
 Enter show id
```

```
 Enter:
 1 - create
 2 - delete
 3 - edit
 4 - get
 5 - return to main menu
1
 Choose table:
 Film - press 1
 Hall - press 2
 Show - press 3
 Ticket - press 4
4
 Enter show id
1
 Enter row
2
 Enter seat
2
Error, ENTER WRONG DATA

 Enter:
```

Для редагування(спроба замінити hall_id на значення якого немає в таблиці hall):





Приклади успішного виконання запитів:

Редагування:

```
5 - return to main menu
4
 Enter table: film, hall, show or ticket
film
(3, 't')
(4, 'fhks')
 Enter:
 1 - create
 2 - delete
 3 - edit
 4 - get
 5 - return to main menu
3
 Choose table:
 Film - press 1
 Hall - press 2
 Show - press 3
 Ticket - press 4
1
 Enter film name
Tor
Enter the id of the film you want to edit
4
```

```
 Choose table:
 Film - press 1
 Hall - press 2
 Show - press 3
 Ticket - press 4
1
 Enter film name
Tor
Enter the id of the film you want to edit
4
 Enter:
 1 - create
 2 - delete
 3 - edit
 4 - get
 5 - return to main menu
4
 Enter table: film, hall, show or ticket
film
(3, 't')
(4, 'Tor')
 Enter:
```

Вилучення:

```
ticket
(1, 3, 2)
(1, 2, 2)
(1, 3, 3)
(1, 22, 1)
(4, 1, 6)
 Enter:
 1 - create
 2 - delete
 3 - edit
 4 - get
 5 - return to main menu
2
 Choose table:
 Film - press 1
 Hall - press 2
 Show - press 3
 Ticket - press 4
4
 Enter the id of the show, row and set you want to delete
```

```
Film - press 1
Hall - press 2
Show - press 3
Ticket - press 4
4
Enter the id of the show, row and set you want to delete
1
3
3
Enter:
1 - create
2 - delete
3 - edit
4 - get
5 - return to main menu
4
Enter table: film, hall, show or ticket
ticket
(1, 3, 2)
(1, 2, 2)
(1, 22, 1)
(4, 1, 6)
Enter:
```

Конїї SQL-запитів:

```python
def create(cursor, table, parameter_1, parameter_2, parameter_3):
    try:
        if table == 1:
            cursor.execute("INSERT INTO film (film_name) VALUES (%s)", (str(parameter_1),))
        elif table == 2:
            cursor.execute("INSERT INTO hall (hall_name, capacity) VALUES (%s, %s)", (parameter_1, parameter_2,))
        elif table == 3:
            cursor.execute("INSERT INTO show (film_id, hall_id, start_date)"
                           " VALUES (%s, %s, %s)", (parameter_1, parameter_2, parameter_3))
        elif table == 4:
            cursor.execute("INSERT INTO ticket (show_id, row, seat)"
                           " VALUES (%s, %s, %s)", (parameter_1, parameter_2, parameter_3,))
    except errors.ForeignKeyViolation:
        print("Error, ENTER WRONG DATA\n")
```

```python
def delete(cursor, table, parameter, parameter_2, parameter_3):
    if table == 1:
        cursor.execute("DELETE FROM film WHERE film_id = %s", [parameter])
    elif table == 2:
        cursor.execute("DELETE FROM hall WHERE hall_id = %s", [parameter])
    elif table == 3:
        cursor.execute("DELETE FROM show WHERE show_id = %s", [parameter])
    elif table == 4:
        cursor.execute("DELETE FROM ticket WHERE show_id = %s and row = %s and seat = %s",
                       (parameter, parameter_2, parameter_3,))
```

```python
def edit(cursor, table, parameter_1, parameter_2, parameter_3, parameter_4, parameter_5):
    try:
        if table == 1:
            cursor.execute("UPDATE film SET film_name = %s"
                           " WHERE film_id = %s", (parameter_1, parameter_2,))
        elif table == 2:
            cursor.execute("UPDATE hall SET hall_name = %s, capacity = %s"
                           " WHERE hall_id = %s", (parameter_1, parameter_2, parameter_3,))
        elif table == 3:
            cursor.execute("UPDATE show SET film_id = %s, hall_id = %s, start_date = %s"
                           " WHERE show_id = %s", (parameter_1, parameter_2, parameter_3, parameter_4,))
        elif table == 4:
            cursor.execute("UPDATE ticket SET row = %s, seat = %s"
                           " WHERE show_id = %s and row = %s and seat = %s",
                           (parameter_1, parameter_2, parameter_3, parameter_4, parameter_5,))
    except errors.ForeignKeyViolation:
        print("Error, ENTER WRONG DATA\n")
```

Пункт №2:

Приклад генерації 100 000 записів для таблиць film:

```
Enter:
1 - task one
2 - task two
3 - task tree
4 - exit
2
Choose table:
Film - press 1
Hall - press 2
Show - press 3
Ticket - press 4
1
Input number of lines in tables
100000
Enter:
1 - task one
2 - task two
3 - task tree
4 - exit
```

Результат:

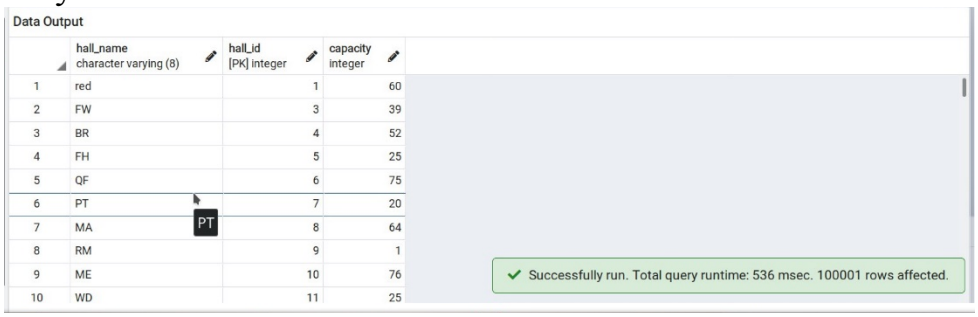
	film_id [PK] integer	film_name character varying (32)
1	3	t
2	4	Tor
3	5	JG
4	6	TB
5	7	DH
6	8	EE
7	9	MK
8	10	BF
9	11	VV
10	12	KR

Successfully run. Total query runtime: 822 msec. 100002 rows affected.

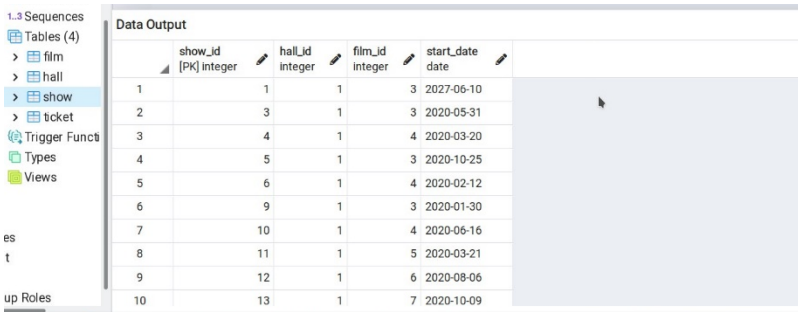
Приклад генерації 100 000 записів для таблиці hall:

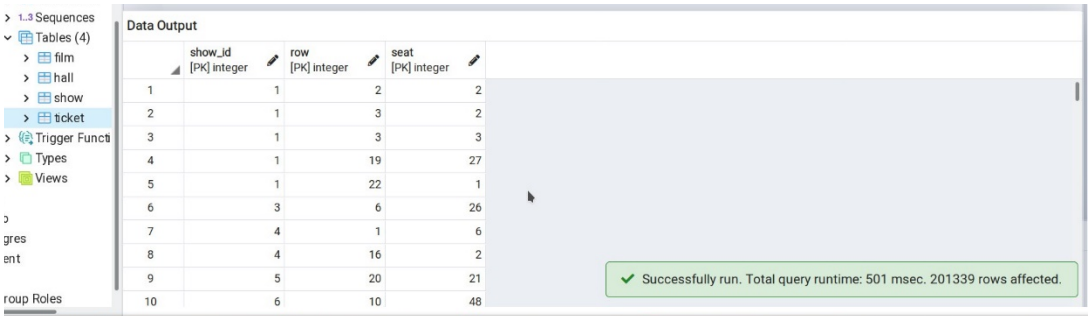
```
2 - task two
3 - task tree
4 - exit
2
Choose table:
Film - press 1
Hall - press 2
Show - press 3
Ticket - press 4
2
Input number of lines in tables
100000
Enter:
1 - task one
2 - task two
3 - task tree
4 - exit
4
```

Результат:



Приклад генерації даних для таблиць show та ticket – вибірка 100% від вхідних таблиць:



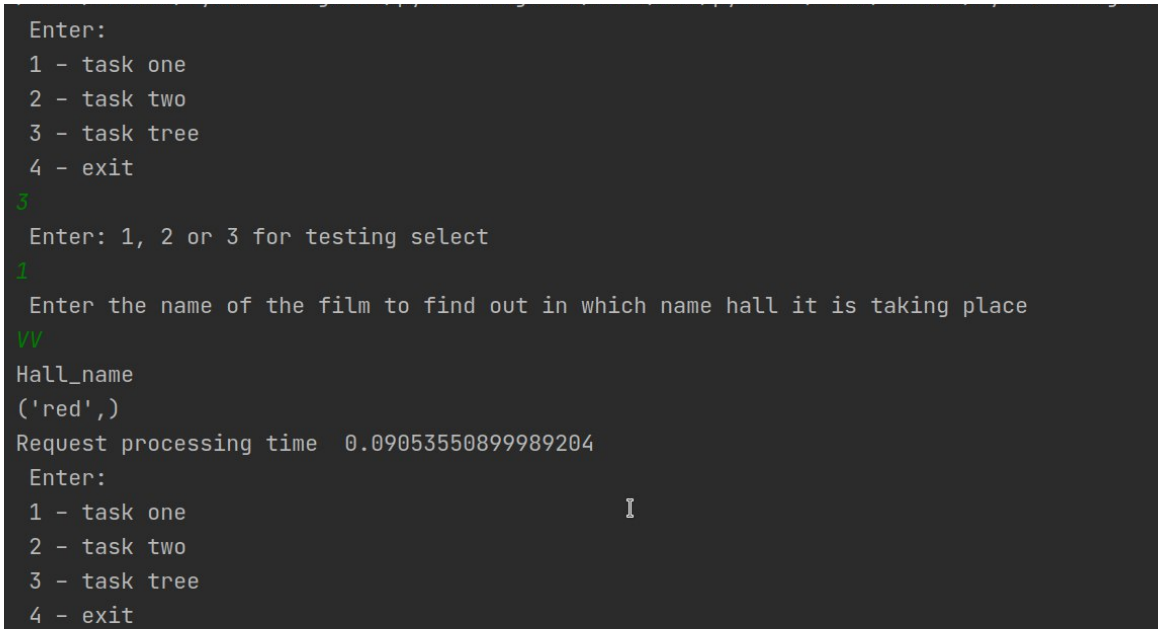


Копії SQL-запитів:

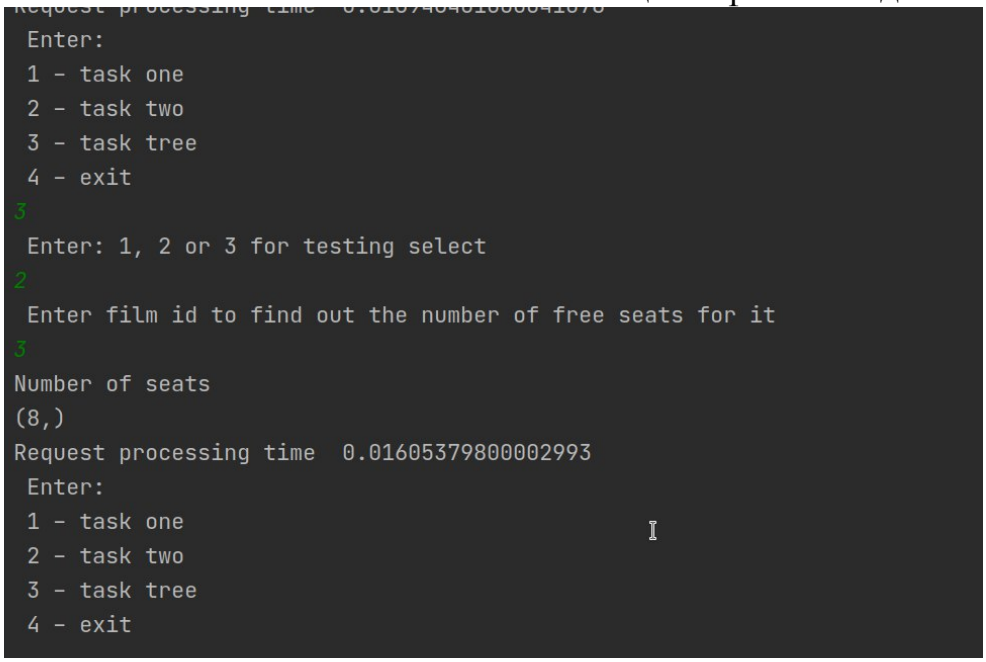
```
def random_generation(cursor, table, number):
    try:
        if table == 1:
            cursor.execute("INSERT INTO film (film_name) SELECT chr(trunc(65 + random()*25)::int)||"
                           " chr(trunc(65 + random()*25)::int) FROM generate_series(1, %s)", [number])
        elif table == 2:
            cursor.execute("INSERT INTO hall (hall_name, capacity) "
                           " SELECT chr(trunc(65 + random()*25)::int)|| chr(trunc(65 + random()*25)::int),"
                           " trunc(random()*100::int) FROM generate_series(1, %s)", [number])
        elif table == 3:
            cursor.execute("INSERT INTO show (start_date, film_id, hall_id) "
                           "SELECT timestamp '2020-01-20 20:00:00' "
                           "+ random() * (timestamp '2020-10-20 20:00:00' - timestamp '2020-01-10 10:00:00'),"
                           "film_id, hall_id FROM film TABLESAMPLE BERNOULLI (100), hall TABLESAMPLE BERNOULLI (100)")
        elif table == 4:
            cursor.execute("INSERT INTO ticket (row, seat, show_id) SELECT  trunc(random()*25::int), "
                           "trunc(random()*100::int), show_id FROM show TABLESAMPLE bernoulli(100) ")
    except psycopg2.errors.UniqueViolation:
        print("Error\n")
```

Пункт №3:

Запит №1 – вивести ім’я залу в якому буде показано фільм з заданим ім’ям:



Запит 2 – вивести кількість вільних місць на фільм з заданим айди:



Запит 3 – вивести загальну кількість місць в залі, в якому буде проходити перегляд з заданим айді:

```
 Enter:
 1 - task one
 2 - task two
 3 - task tree
 4 - exit
3
 Enter: 1, 2 or 3 for testing select
3
 Enter the id of the show to find out the capacity of the hall in which it takes place
3
Capacity
(60,)
Request processing time  0.002330844999960391
 Enter:
 1 - task one
 2 - task two
 3 - task tree
 4 - exit
```

Конїї SQL-запитів:

```
cursor.execute("with a as (SELECT film_id FROM film"
               " where film.film_name = %s), "
               "b as (SELECT hall_id FROM show inner join a on show.film_id = a.film_id)"
               " SELECT hall_name from hall join b on hall.hall_id = b.hall_id", (str(parameter),))
```

```
    cursor.execute("with a as (SELECT show_id FROM show"
                   " where show.film_id = %s)"
                   " SELECT count(seat) FROM ticket join a on ticket.show_id = a.show_id",
                   [parameter])
```

```
    cursor.execute("with a as (SELECT hall_id FROM show"
                     " where show.show_id = %s)"
                     " SELECT capacity FROM hall join a on hall.hall_id = a.hall_id",
                     [parameter])
```