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**

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

Виконав:
студент III курсу
групи КВ-92
Гула М.Р.
Перевірив:
Петрашенко А.В.

Київ – 2021

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції перегляду, 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).

URL репозиторію з вихідним кодом: https://github.com/HulaMR/BD

Результати виконання операції вилучення запису батьківської таблиці:

З таблиці Cinema:

```
D:\PY\bd2\venv\Scripts\python.exe D:/PY/bd2/main.py
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 2

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 1
Num of id: 18
[INFO] Data was successfully deleted
```

	id_cinema [PK] integer	name character varying	adress character varying
1	1	M	President Hrushevsky Avenue, 2, Lutsk, Volyn region, 43000
2	3	One Kino	Svitla Street, 2, Lutsk, Volyn region, 43000
3	4	NCNNJ	Kopernika Street, 2, Lutsk, Volyn region, 43000
4	5	Full relax	Kopernika Street, 2, Lutsk, Volyn region, 43000
5	6	CCFOM	Voli Avenue, 2, Lutsk, Volyn region, 43000
6	7	CLLCN	Svitla Street, 2, Lutsk, Volyn region, 43000
7	8	KMRRF	Svitla Street, 2, Lutsk, Volyn region, 43000
8	9	LNUZG	Voli Avenue, 2, Lutsk, Volyn region, 43000
9	10	OJOKN	Svitla Street, 2, Lutsk, Volyn region, 43000
10	11	CKFNN	Voli Avenue, 2, Lutsk, Volyn region, 43000
11	12	XMSUN	Svitla Street, 2, Lutsk, Volyn region, 43000
12	13	Game cin	Shopena 13 street
13	14	JZTPW	Voli Avenue, 2, Lutsk, Volyn region, 43000
14	15	NUQNQ	Svitla Street, 2, Lutsk, Volyn region, 43000
15	16	XLFW[	Voli Avenue, 2, Lutsk, Volyn region, 43000
16	17	CKCSZ	Voli Avenue, 2, Lutsk, Volyn region, 43000
17	19	NRGEJ	Svitla Street, 2, Lutsk, Volyn region, 43000

З таблиці Cash Register:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 2

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 3
Num of id: 19
[INFO] Data was successfully deleted
```

	id_cr [PK] integer	id_cinema integer	the_name_of_the_cashier character varying	number_of_available_tickets integer
1	1	1	John	79
2	2	1	Bill	84
3	3	5	Maks	400
4	4	13	IZXTT[GMNZNSKE[	215
5	8	5	NVMHI	268
6	9	4	LYXNL	209
7	10	19	EWOHJ	255
8	69	5	WOLQEMJ	242
9	80	5	ZLXVS	274

З таблиці Hall:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 2

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 2
Num of id: 70
[INFO] Data was successfully deleted
```

	id_hall [PK] integer	id_cinema integer	number_of_seats integer
1	1	1	100
2	2	1	200
3	3	1	250
4	4	13	155
5	6	1	264
6	7	1	217
7	8	1	269
8	9	1	299
9	10	13	210
10	11	1	234
11	12	13	245
12	13	15	203
13	14	10	244
14	15	15	215
15	16	13	205
16	91	1	282

Результат, коли вилучення неможливе:

В Cinema:

```
D:\PY\bd2\venv\Scripts\python.exe D:/PY/bd2/main.py
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 2

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 1
Num of id: 19
[INFO] Error while working with PostgreSQL ПОМИЛКА:  update або delete в таблиці "Cinema" порушує обмеження зовнішнього ключа "FK_cr_c" таблиці "Cash register"
DETAIL:  На ключ (id_cinema)=(19) все ще є посилання в таблиці "Cash register".

[INFO] PostgreSQL connection closed

Process finished with exit code 0
```

В Cash register:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 2

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 2
Num of id: 1
[INFO] Error while working with PostgreSQL ПОМИЛКА:  update або delete в таблиці "Hall" порушує обмеження зовнішнього ключа "FK_t_h" таблиці "Ticket"
DETAIL:  На ключ (id_hall)=(1) все ще є посилання в таблиці "Ticket".
```

В Hall:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 2

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 2
Num of id: 2
[INFO] Error while working with PostgreSQL ПОМИЛКА:  update або delete в таблиці "Hall" порушує обмеження зовнішнього ключа "FK_t_h" таблиці "Ticket"
DETAIL:  На ключ (id_hall)=(2) все ще є посилання в таблиці "Ticket".
```

Результати виконання операції вставки запису в дочірню таблицю:

В Hall:

```
D:\PY\bd2\venv\Scripts\python.exe D:/PY/bd2/main.
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 3

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 2
Input new id: 17
Input new cinema id: 12
Input new number of seats: 320
[INFO] Data was successfully inserted
```

	id_hall [PK] integer	id_cinema integer	number_of_seats integer
1	1	1	100
2	2	1	200
3	3	1	250
4	4	13	155
5	6	1	264
6	7	1	217
7	8	1	269
8	9	1	299
9	10	13	210
10	11	1	234
11	12	13	245
12	13	15	203
13	14	10	244
14	15	15	215
15	16	13	205
16	17	12	320
17	91	1	282

В Cash register:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 3

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 3
Input new id: 11
Input new cinema id: 7
Input new name of the cashier: Mike
Input new number of the available tickets: 210
[INFO] Data was successfully inserted
```

	id_cr [PK] integer	id_cinema integer	the_name_of_the_cashier character varying	number_of_available_tickets integer
1	1	1	John	79
2	2	1	Bill	84
3	3	5	Maks	400
4	4	13	IZXTT[GMNZNSKE[	215
5	8	5	NVMHI	268
6	9	4	LYXNL	209
7	10	19	EWOHJ	255
8	11	7	Mike	210
9	69	5	WOLQEMJ	242
10	80	5	ZLXVS	274

B Ticket:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 3

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 4
Input new id: 15
Input new cash register id: 4
Input new hall id: 9
Input new film: 6
Input new price: 79
Input a date in YYYY-MM-DD format: 2021-11-21
Input a time in HH:MM format: 13:00
[INFO] Data was successfully inserted
```

	id_ticket [PK] integer	id_cr integer	id_hall integer	film_id integer	price money	date_and_time timestamp without time zone
1	1	1	2	3	70,00 ?	2021-09-25 13:00:00
2	2	1	2	3	70,00 ?	2021-09-25 14:45:00
3	3	2	3	5	100,00 ?	2021-09-25 19:30:00
4	4	2	1	1	80,00 ?	2021-09-25 18:10:00
5	5	2	2	2	75,00 ?	2021-09-25 10:30:00
6	6	1	3	4	110,00 ?	2021-09-25 21:50:00
7	7	4	4	5	55,00 ?	2021-11-08 19:08:13.770427
8	8	4	4	1	150,00 ?	2021-11-10 00:00:00
9	9	4	4	2	111,00 ?	2021-11-12 17:30:00
10	10	4	4	3	133,00 ?	2021-12-12 21:50:00
11	11	9	9	5	243,00 ?	2022-01-07 05:33:52.73733
12	12	80	1	3	120,00 ?	2021-12-12 12:12:00
13	13	3	12	5	212,00 ?	2021-12-26 09:56:40.194671
14	14	2	91	10	221,00 ?	2022-02-19 09:02:14.162929
15	15	4	9	6	79,00 ?	2021-11-21 13:00:00

В Film:

```
D:\PY\bd2\venv\Scripts\python.exe D:/PY/bd2/main.p
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 3

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 5
Input new id: 25
Input new name: Rocky
Input a duration in HH:MM format: 2:40
[INFO] Data was successfully inserted
```

	id_film [PK] integer	name character varying	duration interval
8	8	DFWJUDVI	02:32:44.925235
9	9	LTFVRBGO	02:14:02.5351
10	10	MGLC[RQM	02:29:00.369381
11	11	SKSPQFLE	02:38:29.542816
12	12	XIIEVGQF	02:42:20.490398
13	13	FGSKPUON	02:35:07.900499
14	14	LQUKMCPK	02:47:48.909114
15	15	FPJUNURN	02:39:30.854841
16	16	UKICIFOI	02:30:14.186797
17	17	HSYLXEZV	02:15:07.874391
18	18	GYJKPZCS	02:25:58.139936
19	19	XKUEERSX	02:50:58.739668
20	20	SPIHZTHB	02:47:24.002707
21	21	MYMKMPXN	02:30:25.261768
22	22	IGCQOPMY	02:33:06.783744
23	23	CTVPCWQX	02:48:16.087188
24	24	BYILMFEM	02:59:56.513536
25	25	Rocky	02:40:00

Неможливість запису(в батьківській таблиці немає відповідного запису):
В Hall:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 3

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 2
Input new id: 17
Input new cinema id: 20
Input new number of seats: 400
[INFO] Error while working with PostgreSQL ПОМИЛКА:  insert або update в таблиці "Hall" порушує обмеження зовнішнього ключа "FK_h_c"
DETAIL:  Ключ (id_cinema)=(20) не присутній в таблиці "Cinema".

[INFO] PostgreSQL connection closed
```

В Cash register:

```
D:\PY\bd2\venv\Scripts\python.exe D:/PY/bd2/main.py
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 3

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 3
Input new id: 11
Input new cinema id: 20
Input new name of the cashier: Mike
Input new number of the available tickets: 45
[INFO] Error while working with PostgreSQL ПОМИЛКА:  insert або update в таблиці "Cash register" порушує обмеження зовнішнього ключа "FK_cr_c"
DETAIL:  Ключ (id_cinema)=(20) не присутній в таблиці "Cinema".
```

В Ticket:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 3

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 4
Input new id: 15
Input new cash register id: 11
Input new hall id: 5
Input new film: 3
Input new price: 120
Input a date in YYYY-MM-DD format: 2021-12-07
Input a time in HH:MM format: 16:30
[INFO] Error while working with PostgreSQL ПОМИЛКА:  insert або update в таблиці "Ticket" порушує обмеження зовнішнього ключа "FK_t_cr"
DETAIL:  Ключ (id_cr)=(11) не присутній в таблиці "Cash register".
```

Ілюстрації фрагментів згенерованих таблиць з відповідними SQL-запитами:

B Cinema:

	id_cinema [PK] integer	name character varying	adress character varying
15	16	XLFW[	Voli Avenue, 2, Lutsk, Volyn region, 43000
16	17	CKCSZ	Voli Avenue, 2, Lutsk, Volyn region, 43000
17	18	NWHEF	Svitla Street, 2, Lutsk, Volyn region, 43000
18	19	NRGEJ	Svitla Street, 2, Lutsk, Volyn region, 43000
19	21	BRZYS	Svitla Street, 2, Lutsk, Volyn region, 43000
20	22	TFIWM	Voli Avenue, 2, Lutsk, Volyn region, 43000
21	23	LNZEQ	Kopernika Street, 2, Lutsk, Volyn region, 43000
22	24	XCEVG	Svitla Street, 2, Lutsk, Volyn region, 43000
23	25	DU[UM	Kopernika Street, 2, Lutsk, Volyn region, 43000
24	26	IMLFN	Kopernika Street, 2, Lutsk, Volyn region, 43000
25	27	RGEQN	Voli Avenue, 2, Lutsk, Volyn region, 43000
26	28	SLTFH	Svitla Street, 2, Lutsk, Volyn region, 43000
27	29	SXOJD	Voli Avenue, 2, Lutsk, Volyn region, 43000

```
cursor.execute("""INSERT INTO "Cinema"(id_cinema, name, adress)
    VALUES((SELECT(select count(id_cinema) from "Cinema")+3::int),
      (select (chr(ascii('B') + (random() * 25)::int) ||
        chr(ascii('B') + (random() * 25)::int) ||
        chr(ascii('B') + (random() * 25)::int) ||
        chr(ascii('B') + (random() * 25)::int) ||
        chr(ascii('B') + (random() * 25)::int)
        )), %s);""", (choice(r_adr),))
```

B Hall:

	id_hall [PK] integer	id_cinema integer	number_of_seats integer
8	9	1	299
9	10	13	210
10	11	1	234
11	12	13	245
12	13	15	203
13	14	10	244
14	15	15	215
15	16	13	205
16	17	12	320
17	18	17	282
18	19	12	207
19	20	12	225
20	21	13	288
21	22	4	221
22	91	1	282

```
cursor.execute("""INSERT INTO "Hall"(id_hall, id_cinema, number_of_seats)
    VALUES((SELECT(select count(id_hall) from "Hall")+1::int),
    (SELECT id_cinema FROM "Cinema" OFFSET
    floor(random()*(select count(id_cinema) from "Cinema")) LIMIT 1),
    trunc(random()*100+200)::int);""")
```

B Cash register:

	id_cr [PK] integer	id_cinema integer	the_name_of_the_cashier character varying	number_of_available_tickets integer
1	1	1	John	79
2	2	1	Bill	84
3	3	5	Maks	400
4	4	13	IZXTT[GMNZNSKE[	215
5	8	5	NVMHI	268
6	9	4	LYXNL	209
7	10	19	EWOHJ	255
8	11	7	Mike	210
9	12	17	LFFSZ	230
10	13	17	NRNEY	215
11	14	18	GOVOM	271
12	15	24	NPIGR	201
13	16	25	EDTCF	204
14	69	5	WOLQEMJ	242
15	80	5	ZLXVS	274

```
    cursor.execute("""INSERT INTO "Cash register"(id_cr, id_cinema,
        the_name_of_the_cashier, number_of_available_tickets)
         VALUES(
         (SELECT(select count(id_cr) from "Cash register")+2::int),
         (SELECT id_cinema FROM "Cinema" OFFSET
         floor(random()*(select count(id_cinema) from "Cinema")) LIMIT 1),
        (select (chr(ascii('B') + (random() * 25)::integer) ||
 chr(ascii('B') + (random() * 25)::integer) ||
 chr(ascii('B') + (random() * 25)::integer) ||
 chr(ascii('B') + (random() * 25)::integer) ||
 chr(ascii('B') + (random() * 25)::integer)
)), trunc(random()*100+200)::int);""")
```

B Ticket:

	id_ticket [PK] integer	id_cr integer	id_hall integer	film_id integer	price money	date_and_time timestamp without time zone
4	4	2	1	1	80,00 ?	2021-09-25 18:10:00
5	5	2	2	2	75,00 ?	2021-09-25 10:30:00
6	6	1	3	4	110,00 ?	2021-09-25 21:50:00
7	7	4	4	5	55,00 ?	2021-11-08 19:08:13.770427
8	8	4	4	1	150,00 ?	2021-11-10 00:00:00
9	9	4	4	2	111,00 ?	2021-11-12 17:30:00
10	10	4	4	3	133,00 ?	2021-12-12 21:50:00
11	11	9	9	5	243,00 ?	2022-01-07 05:33:52.73733
12	12	80	1	3	120,00 ?	2021-12-12 12:12:00
13	13	3	12	5	212,00 ?	2021-12-26 09:56:40.194671
14	14	2	91	10	221,00 ?	2022-02-19 09:02:14.162929
15	15	4	9	6	79,00 ?	2021-11-21 13:00:00
16	16	14	14	5	201,00 ?	2022-02-16 22:57:53.414819
17	17	10	17	17	269,00 ?	2022-01-25 02:21:19.519935
18	18	15	9	15	264,00 ?	2022-01-31 18:42:26.040548

```
cursor.execute("""INSERT INTO "Ticket"(
    id_ticket, id_cr, id_hall, film_id, price, date_and_time)
    VALUES((SELECT(select count(id_ticket) from "Ticket")+1::int),
    (SELECT id_cr FROM "Cash register" OFFSET
     floor(random()*(select count(id_cr) from "Cash register")) LIMIT 1),
    (SELECT id_hall FROM "Hall" OFFSET
     floor(random()*(select count(id_hall) from "Hall")) LIMIT 1),
    (SELECT id_film FROM "Film" OFFSET
     floor(random()*(select count(id_film) from "Film")) LIMIT 1),
    trunc(random()*100+200)::int,
    (select NOW() + (random() * (interval '90 days')) + '30 days')
    );""")
```

B Film:

	id_film [PK] integer	name character varying	duration interval
24	24	[illegible]	[illegible]
25	25	Rocky	02:40:00
26	26	XKXZHVXG	02:17:31.06826
27	27	HCKBHKZQ	02:47:13.033986
28	28	ZOOCHWHO	02:06:17.575575
29	29	JLXHLUEJ	02:19:46.340203
30	30	PFESODQO	02:41:45.832271
31	31	RPYLJYWS	02:57:00.893595
32	32	WMXIHUYG	02:36:47.296109
33	33	HTBLJTDL	02:22:08.359579
34	34	SDOIUQOP	02:57:48.822733
35	35	MVITLWGS	02:08:03.678974
36	36	HRYOCXEP	02:17:40.803605
37	37	JGSHLLXM	02:53:37.538221

```
cursor.execute("""INSERT INTO "Film"(id_film, name, duration)
     VALUES(
    (SELECT(select count(id_Film) from "Film")+1::int),
    (select (chr(ascii('B') + (random() * 24)::integer) ||
            chr(ascii('B') + (random() * 24)::integer) ||
            chr(ascii('B') + (random() * 24)::integer) ||
            chr(ascii('B') + (random() * 24)::integer) ||
            chr(ascii('B') + (random() * 24)::integer) ||
            chr(ascii('B') + (random() * 24)::integer) ||
            chr(ascii('B') + (random() * 24)::integer) ||
            chr(ascii('B') + (random() * 24)::integer) )),
    (select (random() * (interval '1 hour')) + '2 hour'))
    ;""")
```

Ілюстрації результатів пошукових запитів з відповідними SQL-запитами:

В Cinema:

Пошук за id_cinema:

```
Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 1
1 = to search id
2 = to search name or adress
Input:1
Input start: 1
Input finish: 3

Search duration =  0.004288434982299805

id_cinema = 1
name = M
adress = President Hrushevsky Avenue, 2, Lutsk, Volyn region, 43000


id_cinema = 3
name = One Kino
adress = Svitla Street, 2, Lutsk, Volyn region, 43000
```

```
cursor.execute("""SELECT * FROM "Cinema" WHERE %s <= id_cinema and id_cinema <= %s;""", (a, b,))
```

Пошук за name та address:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 5

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 1
1 = to search id
2 = to search name or adress
Input:2
Input text: One kino

Search duration =  0.23225760459899902

id_cinema = 3
name = One Kino
adress = Svitla Street, 2, Lutsk, Volyn region, 43000
```

```
cursor.execute("""SELECT * FROM "Cinema" WHERE
    to_tsvector(name) || to_tsvector(adress)
    @@ plainto_tsquery(%s);;""", (txt,))
```

B Hall:

Пошук за id_hall:

```
Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 2
1 = to search hall id
2 = to search cinema id
3 = to search numbers of seats
Input:1
Input start: 1
Input finish: 2

Search duration =  0.0029976367950439453

id_hall = 1
id_cinema = 1
number_of_seats = 100


id_hall = 2
id_cinema = 1
number_of_seats = 200
```

```
cursor.execute("""SELECT * FROM "Hall" WHERE %s <= id_hall and id_hall <= %s;""", (a, b,))
```

Пошук за id_cinema:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 5

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 2
1 = to search hall id
2 = to search cinema id
3 = to search numbers of seats
Input:2
Input start: 3
Input finish: 5

Search duration =  0.002982616424560547

id_hall = 22
id_cinema = 4
number_of_seats = 221
```

```
cursor.execute("""SELECT * FROM "Hall" WHERE %s <= id_cinema and id_cinema <= %s;""", (a, b,))
```

Пошук за number_of_seats:

```
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 2
1 = to search hall id
2 = to search cinema id
3 = to search numbers of seats
Input:3
Input start: 100
Input finish: 200

Search duration =  0.0029578208923339844

id_hall = 1
id_cinema = 1
number_of_seats = 100


id_hall = 2
id_cinema = 1
number_of_seats = 200


id_hall = 4
id_cinema = 13
number_of_seats = 155
```

```
cursor.execute("""SELECT * FROM "Hall" WHERE %s <= number_of_seats and
      number_of_seats <= %s;""", (a, b,))
```

B Cash register:

Пошук за id_cr:

```
Input: 3
1 = to search cash register id
2 = to search cinema id
3 = to search the_name_of_the_cashier
4 = to search number_of_available_tickets
Input:1
Input start: 12
Input finish: 14

Search duration =  0.002990245819091797

id_cr = 12
id_cinema = 17
the_name_of_the_cashier = LFFSZ
number_of_available_tickets = 230


id_cr = 13
id_cinema = 17
the_name_of_the_cashier = NRNEY
number_of_available_tickets = 215


id_cr = 14
id_cinema = 18
the_name_of_the_cashier = GOVOM
number_of_available_tickets = 271
```

```
cursor.execute("""SELECT * FROM "Cash register" WHERE %s <= id_cr and id_cr <= %s;""", (a, b,))
```

Пошук за id_cinema:

```
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 3
1 = to search cash register id
2 = to search cinema id
3 = to search the_name_of_the_cashier
4 = to search number_of_available_tickets
Input:2
Input start: 5
Input finish: 7

Search duration =  0.0009582042694091797

id_cr = 3
id_cinema = 5
the_name_of_the_cashier = Maks
number_of_available_tickets = 400


id_cr = 69
id_cinema = 5
the_name_of_the_cashier = WOLQEMJ
number_of_available_tickets = 242
```

```
id_cr = 8
id_cinema = 5
the_name_of_the_cashier = NVMHI
number_of_available_tickets = 268


id_cr = 80
id_cinema = 5
the_name_of_the_cashier = ZLXVS
number_of_available_tickets = 274


id_cr = 11
id_cinema = 7
the_name_of_the_cashier = Mike
number_of_available_tickets = 210
```

```
cursor.execute("""SELECT * FROM "Cash register" WHERE %s <= id_cinema and id_cinema <= %s;""",
               (a, b,))
```

Пошук за the_name_of_the_cashier:

```
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 5

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 3
1 = to search cash register id
2 = to search cinema id
3 = to search the_name_of_the_cashier
4 = to search number_of_available_tickets
Input:3
Input name: Bill

Search duration =  0.0029671192169189453

id_cr = 2
id_cinema = 1
the_name_of_the_cashier = Bill
number_of_available_tickets = 84
```

```
cursor.execute("""SELECT * FROM "Cash register" WHERE to_tsvector(the_name_of_the_cashier)
    @@ plainto_tsquery(%s);;""", (txt,))
```

Пошук за number_of_available_tickets:

```
Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 3
1 = to search cash register id
2 = to search cinema id
3 = to search the_name_of_the_cashier
4 = to search number_of_available_tickets
Input:4
Input start: 50
Input finish: 100

Search duration =  0.0009610652923583984

id_cr = 1
id_cinema = 1
the_name_of_the_cashier = John
number_of_available_tickets = 79


id_cr = 2
id_cinema = 1
the_name_of_the_cashier = Bill
number_of_available_tickets = 84
```

```
cursor.execute("""SELECT * FROM "Cash register" WHERE %s <= number_of_available_tickets and
    number_of_available_tickets <= %s;""", (a, b,))
```

В Ticket:

Пошук за id_ticket:

```
Input: 4
1 = to search ticket id
2 = to search cash register id
3 = to search hall id
4 = to search film id
5 = to search price
6 = to search date and time
Input:1
Input start: 1
Input finish: 3

Search duration =  0.005962371826171875

id_ticket = 1
id_cr = 1
id_hall = 2
id_film = 3
price = 70,00 ?
date_and_time = 2021-09-25 13:00:00


id_ticket = 2
id_cr = 1
id_hall = 2
id_film = 3
price = 70,00 ?
date_and_time = 2021-09-25 14:45:00
```

```
id_ticket = 3
id_cr = 2
id_hall = 3
id_film = 5
price = 100,00 ?
date_and_time = 2021-09-25 19:30:00
```

```
cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= id_ticket and id_ticket <= %s;""", (a, b,))
```

Пошук за id_cr:

```
1 = to search ticket id
2 = to search cash register id
3 = to search hall id
4 = to search film id
5 = to search price
6 = to search date and time
Input:2
Input start: 1
Input finish: 1

Search duration =  0.0009622573852539062

id_ticket = 2
id_cr = 1
id_hall = 2
id_film = 3
price = 70,00 ?
date_and_time = 2021-09-25 14:45:00


id_ticket = 1
id_cr = 1
id_hall = 2
id_film = 3
price = 70,00 ?
date_and_time = 2021-09-25 13:00:00
```

```
id_ticket = 6
id_cr = 1
id_hall = 3
id_film = 4
price = 110,00 ?
date_and_time = 2021-09-25 21:50:00
```

```
cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= id_cr and id_cr <= %s;""", (a, b,))
```

Пошук за id_hall:

```
1 = to search ticket id
2 = to search cash register id
3 = to search hall id
4 = to search film id
5 = to search price
6 = to search date and time
Input:3
Input start: 4
Input finish: 4

Search duration =  0.003955364227294922

id_ticket = 7
id_cr = 4
id_hall = 4
id_film = 5
price = 55,00 ?
date_and_time = 2021-11-08 19:08:13.770427


id_ticket = 8
id_cr = 4
id_hall = 4
id_film = 1
price = 150,00 ?
date_and_time = 2021-11-10 00:00:00
```

```
id_ticket = 9
id_cr = 4
id_hall = 4
id_film = 2
price = 111,00 ?
date_and_time = 2021-11-12 17:30:00


id_ticket = 10
id_cr = 4
id_hall = 4
id_film = 3
price = 133,00 ?
date_and_time = 2021-12-12 21:50:00
```

```
cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= id_hall and id_hall <= %s;""", (a, b,))
```

Пошук за film_id:

```
1 = to search ticket id
2 = to search cash register id
3 = to search hall id
4 = to search film id
5 = to search price
6 = to search date and time
Input:4
Input start: 5
Input finish: 5

Search duration =  0.0059854984283447266

id_ticket = 3
id_cr = 2
id_hall = 3
id_film = 5
price = 100,00 ?
date_and_time = 2021-09-25 19:30:00


id_ticket = 7
id_cr = 4
id_hall = 4
id_film = 5
price = 55,00 ?
date_and_time = 2021-11-08 19:08:13.770427
```

```
id_ticket = 11
id_cr = 9
id_hall = 9
id_film = 5
price = 243,00 ?
date_and_time = 2022-01-07 05:33:52.737330


id_ticket = 13
id_cr = 3
id_hall = 12
id_film = 5
price = 212,00 ?
date_and_time = 2021-12-26 09:56:40.194671


id_ticket = 16
id_cr = 14
id_hall = 14
id_film = 5
price = 201,00 ?
date_and_time = 2022-02-16 22:57:53.414819
```

```
cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= film_id and film_id <= %s;""", (a, b,))
```

Пошук за price:

```
1 = to search ticket id
2 = to search cash register id
3 = to search hall id
4 = to search film id
5 = to search price
6 = to search date and time
Input:5
Input start: 70
Input finish: 90

Search duration =  0.001975536346435547

id_ticket = 2
id_cr = 1
id_hall = 2
id_film = 3
price = 70.00
date_and_time = 2021-09-25 14:45:00


id_ticket = 1
id_cr = 1
id_hall = 2
id_film = 3
price = 70.00
date_and_time = 2021-09-25 13:00:00
```

```
id_ticket = 4
id_cr = 2
id_hall = 1
id_film = 1
price = 80.00
date_and_time = 2021-09-25 18:10:00


id_ticket = 5
id_cr = 2
id_hall = 2
id_film = 2
price = 75.00
date_and_time = 2021-09-25 10:30:00


id_ticket = 15
id_cr = 4
id_hall = 9
id_film = 6
price = 79.00
date_and_time = 2021-11-21 13:00:00
```

```
cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= price and price <= %s;""", (a, b,))
```

Пошук за date_and_time:

```
1 = to search ticket id
2 = to search cash register id
3 = to search hall id
4 = to search film id
5 = to search price
6 = to search date and time
Input:6
New start date in YYYY-MM-DD format: 2021-11-12
New start time in HH:MM format: 00:00
New finish date in YYYY-MM-DD format: 2021-12-13
New finish time in HH:MM format: 00:00

Search duration =  0.0019960403442382812

id_ticket = 9
id_cr = 4
id_hall = 4
id_film = 2
price = 111.00
date_and_time = 2021-11-12 17:30:00


id_ticket = 10
id_cr = 4
id_hall = 4
id_film = 3
price = 133.00
date_and_time = 2021-12-12 21:50:00
```

```
id_ticket = 12
id_cr = 80
id_hall = 1
id_film = 3
price = 120.00
date_and_time = 2021-12-12 12:12:00


id_ticket = 15
id_cr = 4
id_hall = 9
id_film = 6
price = 79.00
date_and_time = 2021-11-21 13:00:00
```

```
cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= date_and_time and date_and_time <= %s;""",
               (a, b,))
```

В Film:

Пошук за id_film:

```
Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 5
1 = to search film id
2 = to search name
3 = to search duration
Input: 1
Input start: 2
Input finish: 3

Search duration =  0.0029909610748291016

id_film = 2
name = BOND 25: NO TIME TO DIE
duration = 2:00:00


id_film = 3
name = The Cord Counter
duration = 1:49:00
```

```
cursor.execute("""SELECT * FROM "Film" WHERE %s <= id_film and id_film <= %s;""", (a, b,))
```

Пошук за name:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input: 5

Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 5
1 = to search film id
2 = to search name
3 = to search duration
Input: 2
Input text: Rocky

Search duration =  0.006018400192260742

id_film = 25
name = Rocky
duration = 2:40:00
```

```
cursor.execute("""SELECT * FROM "Film" WHERE to_tsvector(name)
    @@ plainto_tsquery(%s);;""", (txt,))
```

Пошук за duration:

```
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input: 5
1 = to search film id
2 = to search name
3 = to search duration
Input: 3
New start duration in HH:MM format: 02:00
New finish duration in HH:MM format: 02:10

Search duration =  0.00159454345703125

id_film = 2
name = BOND 25: NO TIME TO DIE
duration = 2:00:00


id_film = 28
name = ZOOCHWHO
duration = 2:06:17.575575


id_film = 35
name = MVITLWGS
duration = 2:08:03.678974
```

```
cursor.execute("""SELECT * FROM "Film" WHERE %s <= duration and duration <= %s;""", (a, b,))
```

Ілюстрації програмного коду модуля “Model”:

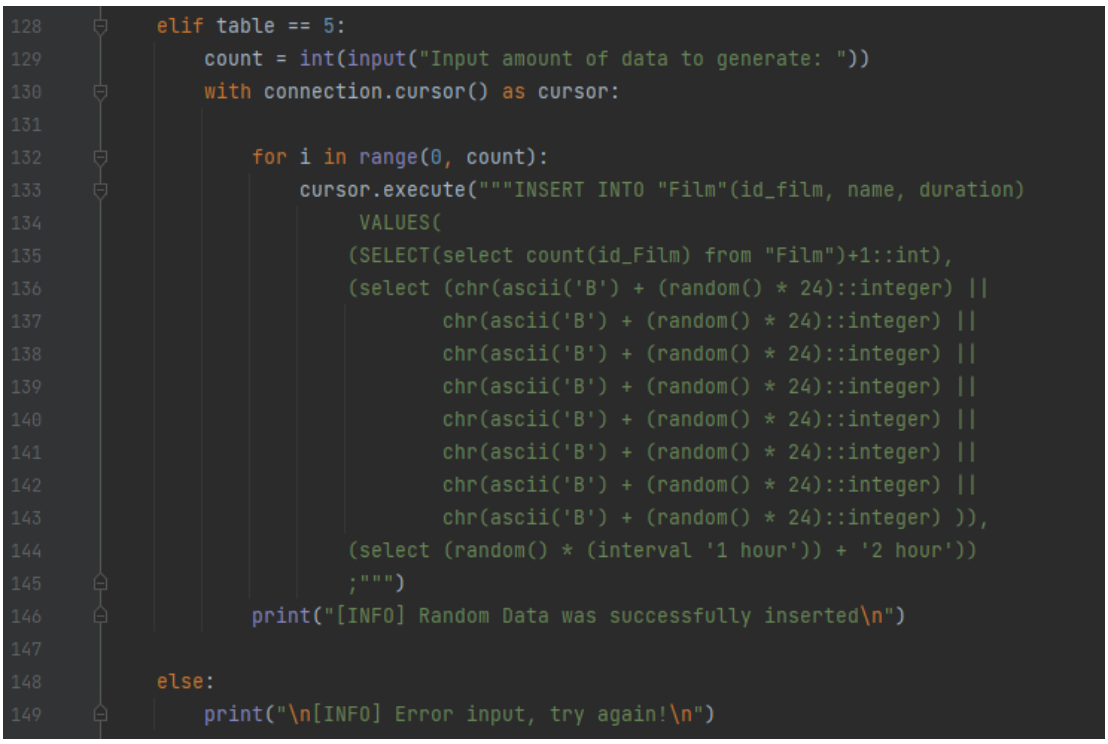
Функція insert:

```
def insert(connection, table):

    if table == 1:
        id_c = int(input("Input new id: "))
        name = str(input("Input new name: "))
        adr = str(input("Input new adress: "))
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Cinema"(id_cinema, name, adress) VALUES(%s, %s, %s);""",
                           (id_c, name, adr,))
            print("[INFO] Data was successfully inserted\n")
    elif table == 2:
        id_h = int(input("Input new id: "))
        id_c = int(input("Input new cinema id: "))
        seats = str(input("Input new number of seats: "))
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Hall"(id_hall, id_cinema, number_of_seats) VALUES(%s, %s, %s);""",
                           (id_h, id_c, seats,))
            print("[INFO] Data was successfully inserted\n")
    elif table == 3:
        id_cr = int(input("Input new id: "))
        id_c = int(input("Input new cinema id: "))
        name = str(input("Input new name of the cashier: "))
        tickets = int(input("Input new number of the available tickets: "))
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Cash register"(id_cr, id_cinema,
                the_name_of_the_cashier, number_of_available_tickets)
                VALUES(%s, %s, %s, %s);""", (id_cr, id_c, name, tickets,))
            print("[INFO] Data was successfully inserted\n")
    elif table == 4:
        id_t = int(input("Input new id: "))
        id_cr = int(input("Input new cash register id: "))
        id_hall = int(input("Input new hall id: "))
        id_film = int(input("Input new film: "))
        price = int(input("Input new price: "))
        date_entry = input("Input a date in YYYY-MM-DD format: ")
        year, month, day = map(int, date_entry.split('-'))
        time_entry = input("Input a time in HH:MM format: ")
        hour, minute = map(int, time_entry.split(':'))
        date_time = datetime.datetime(year, month, day, hour, minute)
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Ticket"(id_ticket, id_cr, id_hall, film_id, price, date_and_time)
                VALUES(%s, %s, %s, %s, %s, %s);""", (id_t, id_cr, id_hall, id_film, price, date_time,))
            print("[INFO] Data was successfully inserted\n")
    elif table == 5:
        id_f = int(input("Input new id: "))
        name = str(input("Input new name: "))
        time_entry = input("Input a duration in HH:MM format: ")
        hour, minute = map(int, time_entry.split(':'))
        duration = datetime.time(hour, minute)
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Film"(id_film, name, duration) VALUES(%s, %s, %s);""",
                           (id_f, name, duration,))
            print("[INFO] Data was successfully inserted\n")
    else:
        print("\n[INFO] Error input, try again!\n")
```

Забезпечує можливість введення даних до таблиць баз даних.

Функція insert_rand:

```
def insert_rand(connection, table):

    if table == 1:
        count = int(input("Input amount of data to generate: "))
        with connection.cursor() as cursor:

            for i in range(0, count):
                cursor.execute("""INSERT INTO "Cinema"(id_cinema, name, adress)
                    VALUES((SELECT(select count(id_cinema) from "Cinema")+3::int),
                      (select (chr(ascii('B') + (random() * 25)::int) ||
                        chr(ascii('B') + (random() * 25)::int) ||
                        chr(ascii('B') + (random() * 25)::int) ||
                        chr(ascii('B') + (random() * 25)::int) ||
                        chr(ascii('B') + (random() * 25)::int)
                        )), %s);""", (choice(r_adr),))
            print("[INFO] Random Data was successfully inserted\n")

    elif table == 2:
        with connection.cursor() as cursor:

            cursor.execute("""INSERT INTO "Hall"(id_hall, id_cinema, number_of_seats)
                 VALUES((SELECT(select count(id_hall) from "Hall")+1::int),
                 (SELECT id_cinema FROM "Cinema" OFFSET
                 floor(random()*(select count(id_cinema) from "Cinema")) LIMIT 1),
                 trunc(random()*100+200)::int);""")
            print("[INFO] Random Data was successfully inserted\n")

    elif table == 3:
        with connection.cursor() as cursor:

            cursor.execute("""INSERT INTO "Cash register"(id_cr, id_cinema,
                the_name_of_the_cashier, number_of_available_tickets)
                 VALUES(
                 (SELECT(select count(id_cr) from "Cash register")+2::int),
                 (SELECT id_cinema FROM "Cinema" OFFSET
                 floor(random()*(select count(id_cinema) from "Cinema")) LIMIT 1),
                (select (chr(ascii('B') + (random() * 25)::integer) ||
        chr(ascii('B') + (random() * 25)::integer) ||
        chr(ascii('B') + (random() * 25)::integer) ||
        chr(ascii('B') + (random() * 25)::integer) ||
        chr(ascii('B') + (random() * 25)::integer)
       )), trunc(random()*100+200)::int);""")
        print("[INFO] Random Data was successfully inserted\n")

    elif table == 4:
        with connection.cursor() as cursor:

            cursor.execute("""INSERT INTO "Ticket"(
                id_ticket, id_cr, id_hall, film_id, price, date_and_time)
                VALUES((SELECT(select count(id_ticket) from "Ticket")+1::int),
                (SELECT id_cr FROM "Cash register" OFFSET
                 floor(random()*(select count(id_cr) from "Cash register")) LIMIT 1),
                (SELECT id_hall FROM "Hall" OFFSET
                 floor(random()*(select count(id_hall) from "Hall")) LIMIT 1),
                (SELECT id_film FROM "Film" OFFSET
                 floor(random()*(select count(id_film) from "Film")) LIMIT 1),
                trunc(random()*100+200)::int,
                (select NOW() + (random() * (interval '90 days')) + '30 days')
                );""")
        print("[INFO] Random Data was successfully inserted\n")

```



```
elif table == 5:
    count = int(input("Input amount of data to generate: "))
    with connection.cursor() as cursor:

        for i in range(0, count):
            cursor.execute("""INSERT INTO "Film"(id_film, name, duration)
                 VALUES(
                (SELECT(select count(id_Film) from "Film")+1::int),
                (select (chr(ascii('B') + (random() * 24)::integer) ||
                        chr(ascii('B') + (random() * 24)::integer) ||
                        chr(ascii('B') + (random() * 24)::integer) ||
                        chr(ascii('B') + (random() * 24)::integer) ||
                        chr(ascii('B') + (random() * 24)::integer) ||
                        chr(ascii('B') + (random() * 24)::integer) ||
                        chr(ascii('B') + (random() * 24)::integer) ||
                        chr(ascii('B') + (random() * 24)::integer) )),
                (select (random() * (interval '1 hour')) + '2 hour'))
                ;""")
            print("[INFO] Random Data was successfully inserted\n")

else:
    print("\n[INFO] Error input, try again!\n")
```

Забезпечує можливість заповнення таблиць баз даних випадковими даними.

Функція update:

```
def update(connection, table):

    if table == 1:
        id_c = int(input("Num of id: "))
        column = int(input("1 = to upd name\n"
                           "2 = to upd adress\n"
                           "Input:"))
        if column == 1:
            new = str(input("New name: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cinema" set name = %s where id_cinema = %s;""", (new, id_c,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            new = str(input("New adress: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cinema" set adress = %s where id_cinema = %s;""", (new, id_c,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 1)

    elif table == 2:
        id_h = int(input("Num of id: "))
        column = int(input("1 = to upd id_cinema\n"
                           "2 = to upd number of seats\n"
                           "Input:"))
        if column == 1:
            new = str(input("New cinema id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Hall" set id_cinema = %s where id_hall = %s;""", (new, id_h,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            new = str(input("New number of seats: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Hall" set number_of_seats = %s where id_hall = %s;""", (new, id_h,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 2)

    elif table == 3:
        id_cr = int(input("Num of id: "))
        column = int(input("1 = to upd id_cinema\n"
                           "2 = to upd the_name_of_the_cashier\n"
                           "3 = to upd number_of_available_tickets\n"
                           "Input:"))
        if column == 1:
            new = int(input("New cinema id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cash register" set id_cinema = %s where id_cr = %s;""", (new, id_cr,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            new = str(input("New number of seats: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cash register" set the_name_of_the_cashier = %s where id_cr = %s;""",
                               (new, id_cr,))
                print("[INFO] Data was successfully upd\n")
        elif column == 3:
            new = str(input("New number of seats: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cash register" set number_of_available_tickets = %s where id_cr = %s;""",
                               (new, id_cr,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 3)
```

```
    elif table == 4:
        id_t = int(input("Num of id: "))
        column = int(input("1 = to upd cash register id\n"
                           "2 = to upd hall id\n"
                           "3 = to upd film id\n"
                           "4 = to upd price\n"
                           "5 = to upd date and time\n"
                           "Input:"))
        if column == 1:
            new = int(input("New cash register id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set id_cr = %s where id_ticket = %s;""", (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            new = int(input("New hall id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set id_hall = %s where id_ticket = %s;""",
                               (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        elif column == 3:
            new = int(input("New film id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set film_id = %s where id_ticket = %s;""",
                               (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        elif column == 4:
            new = int(input("New price: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set price = %s where id_ticket = %s;""",
                               (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        elif column == 5:
            date_entry = input("New date in YYYY-MM-DD format: ")
            year, month, day = map(int, date_entry.split('-'))
            time_entry = input("New time in HH:MM format: ")
            hour, minute = map(int, time_entry.split(':'))
            new = datetime.datetime(year, month, day, hour, minute)
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set date_and_time = %s where id_ticket = %s;""",
                               (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 4)
```

```
    elif table == 5:
        id_f = int(input("Num of id: "))
        column = int(input("1 = to upd name\n"
                           "2 = to upd duration\n"
                           "Input:"))
        if column == 1:
            new = str(input("New name: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Film" set name = %s where id_film = %s;""", (new, id_f,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            time_entry = input("New duration in HH:MM format: ")
            hour, minute = map(int, time_entry.split(':'))
            new = datetime.time(hour, minute)
            with connection.cursor() as cursor:
                cursor.execute("""update "Film" set duration = %s where id_film = %s;""", (new, id_f,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 5)

    else:
        print("\n[INFO] Error input, try again!\n")
```

Забезпечує можливість оновлення потрібних даних в таблиці баз даних.

Функція delete:

```
def delete(connection, table):

    if table == 1:
        id_c = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Cinema" WHERE id_cinema = %s;""", (id_c,))
            print("[INFO] Data was successfully deleted\n")
    elif table == 2:
        id_h = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Hall" WHERE id_hall = %s;""", (id_h,))
            print("[INFO] Data was successfully deleted\n")
    elif table == 3:
        id_cr = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Cash register" WHERE id_cr = %s;""", (id_cr,))
            print("[INFO] Data was successfully deleted\n")
    elif table == 4:
        id_t = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Ticket" WHERE id_ticket = %s;""", (id_t,))
            print("[INFO] Data was successfully deleted\n")
    elif table == 5:
        id_f = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Film" WHERE id_film = %s;""", (id_f,))
            print("[INFO] Data was successfully deleted\n")
    else:
        print("\n[INFO] Error input, try again!\n")
```

Забезпечує можливість видалення потрібних таблиць в таблиці баз даних.

Функція select_table:

```
def select_table(connection, table):

    if table == 1:
        string = int(input("1 = to one str\n"
                           "2 = to all str\n"
                           "Input:"))
        if string == 1:
            id_c = int(input("Num of id: "))
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Cinema" where id_cinema = %s;""", (id_c,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Cinema";""")
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            select_table(connection, 1)

    elif table == 2:
        string = int(input("1 = to one str\n"
                           "2 = to all str\n"
                           "Input:"))
        if string == 1:
            id_c = int(input("Num of id: "))
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Hall" where id_hall = %s;""", (id_c,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Hall";""")
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            select_table(connection, 2)

    elif table == 3:
        string = int(input("1 = to one str\n"
                           "2 = to all str\n"
                           "Input:"))
        if string == 1:
            id_c = int(input("Num of id: "))
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Cash register" where id_cr = %s;""", (id_c,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Cash register";""")
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            select_table(connection, 3)
```

```
376     elif table == 4:
377         string = int(input("1 = to one str\n"
378                            "2 = to all str\n"
379                            "Input:"))
380         if string == 1:
381             id_c = int(input("Num of id: "))
382             with connection.cursor() as cursor:
383                 cursor.execute("""SELECT * FROM "Ticket" where id_ticket = %s;""", (id_c,))
384                 return cursor.fetchmany(1)
385 
386         elif string == 2:
387             with connection.cursor() as cursor:
388                 cursor.execute("""SELECT * FROM "Ticket";""")
389                 return cursor.fetchall()
390         else:
391             print("\n[INFO] Error input, try again!\n")
392             select_table(connection, 4)
393 
394     elif table == 5:
395         string = int(input("1 = to one str\n"
396                            "2 = to all str\n"
397                            "Input:"))
398         if string == 1:
399             id_f = int(input("Num of id: "))
400             with connection.cursor() as cursor:
401                 cursor.execute("""SELECT * FROM "Film" where id_film = %s;""", (id_f,))
402                 return cursor.fetchmany(1)
403 
404         elif string == 2:
405             with connection.cursor() as cursor:
406                 cursor.execute("""SELECT * FROM "Film";""")
407                 return cursor.fetchall()
408         else:
409             print("\n[INFO] Error input, try again!\n")
410             select_table(connection, 5)
411 
412     else:
413         print("\n[INFO] Error input, try again!\n")
414 
```

Забезпечує можливість виведення потрібних таблиць баз даних, де ми можемо вивести, як всі таблиці, так і одну

Функція search:

```
def search(connection, table):

    if table == 1:
        column = int(input("1 = to search id\n"
                           "2 = to search name or adress\n"
                           "Input:"))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cinema" WHERE %s <= id_cinema and id_cinema <= %s;""", (a, b,))
                finish = time.time()
                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            txt = str(input("Input text: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cinema" WHERE
                    to_tsvector(name) || to_tsvector(adress)
                    @@ plainto_tsquery(%s);;""", (txt,))
                finish = time.time()
                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 1)
```

```
    elif table == 2:
        column = int(input("1 = to search hall id\n"
                           "2 = to search cinema id\n"
                           "3 = to search numbers of seats\n"
                           "Input:"))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Hall" WHERE %s <= id_hall and id_hall <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Hall" WHERE %s <= id_cinema and id_cinema <= %s;""", (a, b,))
                finish = time.time()
                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 3:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Hall" WHERE %s <= number_of_seats and
                     number_of_seats <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 2)
```

```
    elif table == 3:
        column = int(input("1 = to search cash register id\n"
                           "2 = to search cinema id\n"
                           "3 = to search the_name_of_the_cashier\n"
                           "4 = to search number_of_available_tickets\n"
                           "Input:"))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cash register" WHERE %s <= id_cr and id_cr <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cash register" WHERE %s <= id_cinema and id_cinema <= %s;""",
                               (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 3:
            txt = str(input("Input name: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cash register" WHERE to_tsvector(the_name_of_the_cashier)
                    @@ plainto_tsquery(%s);;""", (txt,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 4:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cash register" WHERE %s <= number_of_available_tickets and
                    number_of_available_tickets <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 3)
```

```
elif table == 4:
    column = int(input("1 = to search ticket id\n"
                       "2 = to search cash register id\n"
                       "3 = to search hall id\n"
                       "4 = to search film id\n"
                       "5 = to search price\n"
                       "6 = to search date and time\n"
                       "Input:"))
    if column == 1:
        a = int(input("Input start: "))
        b = int(input("Input finish: "))
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= id_ticket and id_ticket <= %s;""", (a, b,))
            finish = time.time()

            print("\nSearch duration = ", finish - start)
            return cursor.fetchall()

    elif column == 2:
        a = int(input("Input start: "))
        b = int(input("Input finish: "))
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= id_cr and id_cr <= %s;""", (a, b,))
            finish = time.time()

            print("\nSearch duration = ", finish - start)
            return cursor.fetchall()

    elif column == 3:
        a = int(input("Input start: "))
        b = int(input("Input finish: "))
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= id_hall and id_hall <= %s;""", (a, b,))
            finish = time.time()

            print("\nSearch duration = ", finish - start)
            return cursor.fetchall()

    elif column == 4:
        a = int(input("Input start: "))
        b = int(input("Input finish: "))
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= film_id and film_id <= %s;""", (a, b,))
            finish = time.time()

            print("\nSearch duration = ", finish - start)
            return cursor.fetchall()
```

```
    elif column == 5:
        a = float(input("Input start: "))
        b = float(input("Input finish: "))
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= price and price <= %s;""", (a, b,))
            finish = time.time()

            print("\nSearch duration = ", finish - start)
            return cursor.fetchall()

    elif column == 6:
        date_entry_a = input("New start date in YYYY-MM-DD format: ")
        year, month, day = map(int, date_entry_a.split('-'))
        time_entry_a = input("New start time in HH:MM format: ")
        hour, minute = map(int, time_entry_a.split(':'))
        a = datetime.datetime(year, month, day, hour, minute)

        date_entry_b = input("New finish date in YYYY-MM-DD format: ")
        year, month, day = map(int, date_entry_b.split('-'))
        time_entry_b = input("New finish time in HH:MM format: ")
        hour, minute = map(int, time_entry_b.split(':'))
        b = datetime.datetime(year, month, day, hour, minute)
        with connection.cursor() as cursor:
            start = time.time()
            cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= date_and_time and date_and_time <= %s;""",
                           (a, b,))
            finish = time.time()

            print("\nSearch duration = ", finish - start)
            return cursor.fetchall()
    else:
        print("\n[INFO] Error input, try again!\n")
        search(connection, 4)
```

```
    elif table == 5:
        column = int(input("1 = to search film id\n"
                           "2 = to search name\n"
                           "3 = to search duration\n"
                           "Input: "))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Film" WHERE %s <= id_film and id_film <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            txt = str(input("Input text: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Film" WHERE to_tsvector(name)
                    @@ plainto_tsquery(%s);;""", (txt,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 3:

            time_entry_a = input("New start duration in HH:MM format: ")
            hour, minute = map(int, time_entry_a.split(':'))
            a = datetime.time(hour, minute)

            time_entry_b = input("New finish duration in HH:MM format: ")
            hour, minute = map(int, time_entry_b.split(':'))
            b = datetime.time(hour, minute)
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Film" WHERE %s <= duration and duration <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 5)

    else:
        print("\n[INFO] Error input, try again!\n")
```

Забезпечує можливість пошуку потрібних даних по всі базі даних.

.

Модель «сутність-зв’язок»:

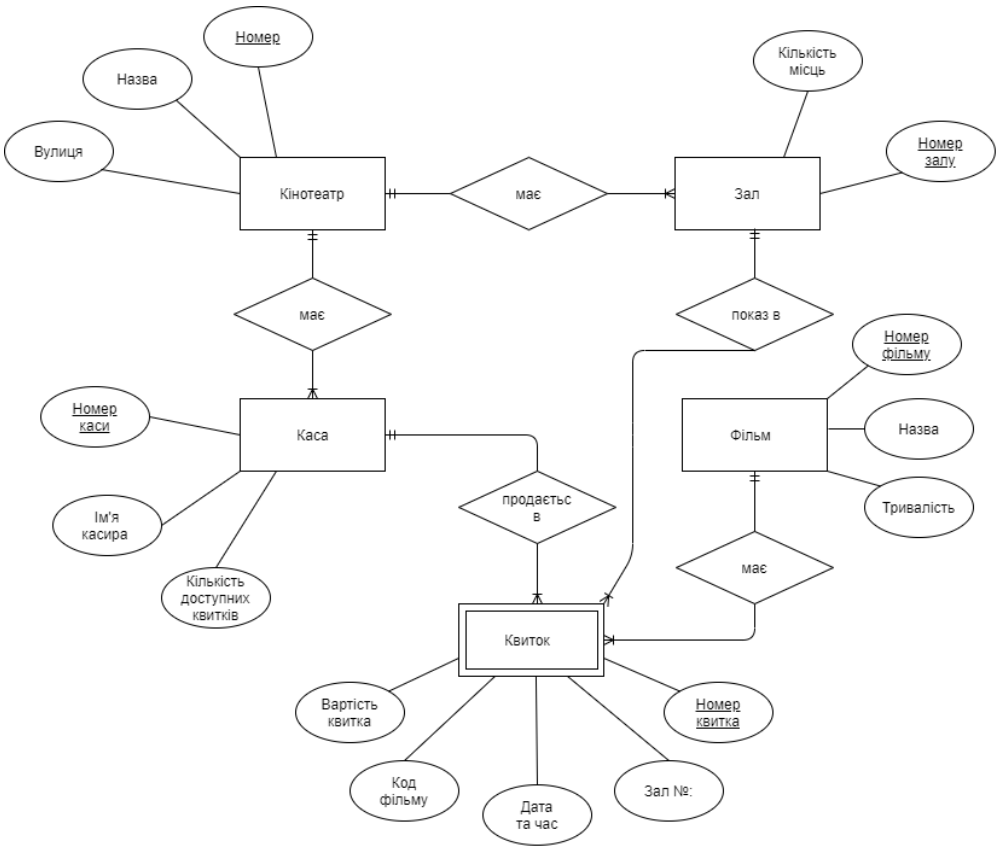


Рис1. ER-діаграма за нотацією Чена

Схеми бази даних:

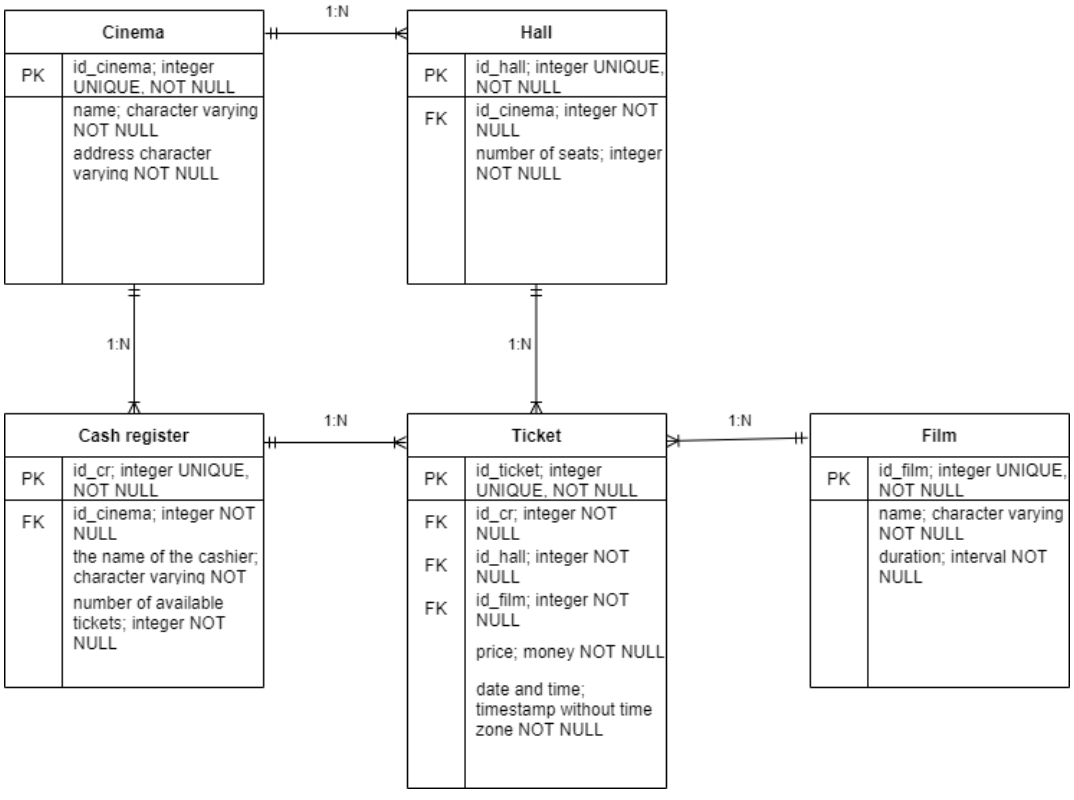


Рис2. Схема бази даних

Сутність	Атрибут	Тип атрибуту
Cinema – містить інформацію про кінотеатр	**id_cinema** – унікальний ідентифікатор кінотеатру **name** – назва кінотеатру **address** – адреса кінотеатру	**integer** **character varying** **character varying**
Cash register – містить інформацію про продаж квитків	**id_cr** – унікальний ідентифікатор каси **id_cinema** – ідентифікатор кінотеатру в якому знаходить каса **the_name_of_the_cashier** – ім'я касира **number_of_available_tickets** – кількість доступних квитків	**integer** **integer** **character varying** **integer**
Hall – містить у собі дані залів	**id_hall** – унікальний ідентифікатор залу **id_cinema** – ідентифікатор кінотеатру в якому знаходить зал **number_of_seats** – кількість місць в залі	**integer** **integer** **integer**
Ticket – містить дані про сеанс на фільм	**id_ticket** – унікальний ідентифікатор квитка **id_cr** – ідентифікатор каси, де було куплено квиток **id_hall** – ідентифікатор залу, в якому відбудеться сеанс **film_id** – ідентифікатор фільм, на який куплено квиток **price** – ціна квитка **date_and_time** – дата та час початку сеансу	**integer** **integer** **integer** **integer** **numeric** **timestamp without time zone**
Film – містить в собі всі фільми, які доступні до перегляду	**id_film** – унікальний ідентифікатор фільму **name** – назва фільму **duration** – тривалість фільму	**integer** **character varying** **interval**

Опис зв'язків:

У Кінотеатрі є декілька Кас та Залів, тому між сутностями Кінотеатр і Каса та сутностями Кінотеатр і Зал зв'язок 1:N.

Так як в одній Касі можна купити багато Квитків, то сутності Каса і квиток мають зв’язок 1:N.
При купівлі Квитка вибирається один Фільм, з декількох доступних, тому сутності Квиток і Фільм мають зв’язок 1:N.
За день в Залі відбувається показ багатьох різних Фільмів, тому сутності Зал і Фільм мають зв’язок 1:N.

Опис меню:
Головне меню:

```
1 to update
2 to delete
3 to insert
4 to select table
5 to search
6 to insert random
7 to end
Input:
```

1 – оновлення даних в вибраній таблиці
2 – видалення даних в вибраній таблиці
3 – введення даних в вибрану таблицю
4 – вивід вибраної таблиці
5 – пошук по таблицями бази даних
6 – ввід випадкових даних у таблицю
7 – вихід з меню

Після вибору одного з пунктів 1 – 6, ми побачимо наступне меню для вибору таблиці бази даних:

```
Select a table:
1 = Cinema
2 = Hall
3 = Cash register
4 = Ticket
5 = Film
Input:
```

Якщо, в першому меню ми вибрали пункт 1 та в другому вибрали таблицю, то побачимо:

```
Num of id: 1
1 = to upd name
2 = to upd adress
Input:
```

меню для вибору даних для оновлення.

Якщо, в першому меню ми вибрали пункт 2 та в другому вибрали таблицю, то побачимо:

```
Num of id: 1
```

меню де вводимо id для видалення рядка.

Якщо, в першому меню ми вибрали пункт 3 та в другому вибрали таблицю, то побачимо:

```
Input new id: 30
Input new name: Premier
Input new adress: Svitla 13
```

меню для вводу нових даних

Якщо, в першому меню ми вибрали пункт 4 та в другому вибрали таблицю, то побачимо:

```
1 = to one str
2 = to all str
Input:
```

меню для вибору вивести один рядок чи всі,

```
Input:1
Num of id:
```

якщо, вибираємо 1 то вводимо id рядка для виводу.

Якщо, в першому меню ми вибрали пункт 5 та в другому вибрали таблицю, то побачимо:

```
1 = to search id
2 = to search name or adress
Input:
```

меню для вибору атрибути, та пошуку конкретних даних

Якщо, в першому меню ми вибрали пункт 6 та в другому вибрали таблицю, то побачимо:

- для таблиці 1 та 5

```
Input amount of data to generate:
```

меню для вводу кількості даних для генерації

- для інших таблиць буде проведена одна генерація

Код програми:

main.py

```
import psycopg2
from Controller import menu

try:
    connection = psycopg2.connect(
        database="Labs",
        user="postgres",
        password="4214",
        host="127.0.0.1")
    connection.autocommit = True

    menu(connection)

except Exception as _ex:
    print("[INFO] Error while working with PostgreSQL", _ex)
finally:
    if connection:
        connection.close()
        print("[INFO] PostgreSQL connection closed")
```

Model.py

```
import time
import datetime
from random import *

r_adr = ['Voli Avenue, 2, Lutsk, Volyn region, 43000',
         'Svitla Street, 2, Lutsk, Volyn region, 43000',
         'Kopernika Street, 2, Lutsk, Volyn region, 43000']


def insert(connection, table):

    if table == 1:
        id_c = int(input("Input new id: "))
        name = str(input("Input new name: "))
        adr = str(input("Input new adress: "))
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Cinema"(id_cinema, name, adress)
VALUES(%s, %s, %s);""",
                           (id_c, name, adr,))
            print("[INFO] Data was successfully inserted\n")
    elif table == 2:
        id_h = int(input("Input new id: "))
        id_c = int(input("Input new cinema id: "))
        seats = str(input("Input new number of seats: "))
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Hall"(id_hall, id_cinema,
number_of_seats) VALUES(%s, %s, %s);""",
                           (id_h, id_c, seats,))
            print("[INFO] Data was successfully inserted\n")
    elif table == 3:
        id_cr = int(input("Input new id: "))
        id_c = int(input("Input new cinema id: "))
        name = str(input("Input new name of the cashier: "))
        tickets = int(input("Input new number of the available tickets: "))
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Cash register"(id_cr, id_cinema,
                the_name_of_the_cashier, number_of_available_tickets)
                VALUES(%s, %s, %s, %s);""", (id_cr, id_c, name, tickets,))
            print("[INFO] Data was successfully inserted\n")
    elif table == 4:
```

```
        id_t = int(input("Input new id: "))
        id_cr = int(input("Input new cash register id: "))
        id_hall = int(input("Input new hall id: "))
        id_film = int(input("Input new film: "))
        price = int(input("Input new price: "))
        date_entry = input("Input a date in YYYY-MM-DD format: ")
        year, month, day = map(int, date_entry.split('-'))
        time_entry = input("Input a time in HH:MM format: ")
        hour, minute = map(int, time_entry.split(':'))
        date_time = datetime.datetime(year, month, day, hour, minute)
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Ticket"(id_ticket, id_cr, id_hall,
film_id, price, date_and_time)
                VALUES(%s, %s, %s, %s, %s, %s);""", (id_t, id_cr, id_hall,
id_film, price, date_time,))
            print("[INFO] Data was successfully inserted\n")
    elif table == 5:
        id_f = int(input("Input new id: "))
        name = str(input("Input new name: "))
        time_entry = input("Input a duration in HH:MM format: ")
        hour, minute = map(int, time_entry.split(':'))
        duration = datetime.time(hour, minute)
        with connection.cursor() as cursor:
            cursor.execute("""INSERT INTO "Film"(id_film, name, duration)
VALUES(%s, %s, %s);""",
                           (id_f, name, duration,))
            print("[INFO] Data was successfully inserted\n")
    else:
        print("\n[INFO] Error input, try again!\n")


def insert_rand(connection, table):

    if table == 1:
        count = int(input("Input amount of data to generate: "))
        with connection.cursor() as cursor:

            for i in range(0, count):
                cursor.execute("""INSERT INTO "Cinema"(id_cinema, name,
adress)
                    VALUES((SELECT(select count(id_cinema) from
"Cinema")+3::int),
                      (select (chr(ascii('B') + (random() * 25)::int) ||
                        chr(ascii('B') + (random() * 25)::int) ||
                        chr(ascii('B') + (random() * 25)::int) ||
                        chr(ascii('B') + (random() * 25)::int) ||
                        chr(ascii('B') + (random() * 25)::int)
                        )), %s);""", (choice(r_adr),))
            print("[INFO] Random Data was successfully inserted\n")

    elif table == 2:
        with connection.cursor() as cursor:

            cursor.execute("""INSERT INTO "Hall"(id_hall, id_cinema,
number_of_seats)
                VALUES((SELECT(select count(id_hall) from "Hall")+1::int),
                (SELECT id_cinema FROM "Cinema" OFFSET
                floor(random()*(select count(id_cinema) from "Cinema"))
LIMIT 1),
                trunc(random()*100+200)::int);""")
            print("[INFO] Random Data was successfully inserted\n")

    elif table == 3:
        with connection.cursor() as cursor:
```

```
            cursor.execute("""INSERT INTO "Cash register"(id_cr, id_cinema,
                the_name_of_the_cashier, number_of_available_tickets)
                 VALUES(
                 (SELECT(select count(id_cr) from "Cash register")+2::int),
                 (SELECT id_cinema FROM "Cinema" OFFSET
                 floor(random()*(select count(id_cinema) from "Cinema"))
LIMIT 1),
                (select (chr(ascii('B') + (random() * 25)::integer) ||
        chr(ascii('B') + (random() * 25)::integer) ||
        chr(ascii('B') + (random() * 25)::integer) ||
        chr(ascii('B') + (random() * 25)::integer) ||
        chr(ascii('B') + (random() * 25)::integer)
       )), trunc(random()*100+200)::int);""")
        print("[INFO] Random Data was successfully inserted\n")

    elif table == 4:
        with connection.cursor() as cursor:

            cursor.execute("""INSERT INTO "Ticket"(
                id_ticket, id_cr, id_hall, film_id, price, date_and_time)
                VALUES((SELECT(select count(id_ticket) from
"Ticket")+1::int),
                (SELECT id_cr FROM "Cash register" OFFSET
                 floor(random()*(select count(id_cr) from "Cash register"))
LIMIT 1),
                (SELECT id_hall FROM "Hall" OFFSET
                 floor(random()*(select count(id_hall) from "Hall")) LIMIT
1),
                (SELECT id_film FROM "Film" OFFSET
                 floor(random()*(select count(id_film) from "Film")) LIMIT
1),
                trunc(random()*100+200)::int,
                (select NOW() + (random() * (interval '90 days')) + '30
days')
                );""")
        print("[INFO] Random Data was successfully inserted\n")

    elif table == 5:
        count = int(input("Input amount of data to generate: "))
        with connection.cursor() as cursor:

            for i in range(0, count):
                cursor.execute("""INSERT INTO "Film"(id_film, name, duration)
                     VALUES(
                    (SELECT(select count(id_Film) from "Film")+1::int),
                    (select (chr(ascii('B') + (random() * 24)::integer) ||
                            chr(ascii('B') + (random() * 24)::integer) ||
                            chr(ascii('B') + (random() * 24)::integer) ||
                            chr(ascii('B') + (random() * 24)::integer) ||
                            chr(ascii('B') + (random() * 24)::integer) ||
                            chr(ascii('B') + (random() * 24)::integer) ||
                            chr(ascii('B') + (random() * 24)::integer) ||
                            chr(ascii('B') + (random() * 24)::integer) )),
                    (select (random() * (interval '1 hour')) + '2 hour'))
                    ;""")
            print("[INFO] Random Data was successfully inserted\n")

    else:
        print("\n[INFO] Error input, try again!\n")


def update(connection, table):
```

```
    if table == 1:
        id_c = int(input("Num of id: "))
        column = int(input("1 = to upd name\n"
                           "2 = to upd adress\n"
                           "Input:"))
        if column == 1:
            new = str(input("New name: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cinema" set name = %s where
id_cinema = %s;""", (new, id_c,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            new = str(input("New adress: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cinema" set adress = %s where
id_cinema = %s;""", (new, id_c,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 1)

    elif table == 2:
        id_h = int(input("Num of id: "))
        column = int(input("1 = to upd id_cinema\n"
                           "2 = to upd number of seats\n"
                           "Input:"))
        if column == 1:
            new = str(input("New cinema id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Hall" set id_cinema = %s where
id_hall = %s;""", (new, id_h,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            new = str(input("New number of seats: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Hall" set number_of_seats = %s
where id_hall = %s;""", (new, id_h,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 2)

    elif table == 3:
        id_cr = int(input("Num of id: "))
        column = int(input("1 = to upd id_cinema\n"
                           "2 = to upd the_name_of_the_cashier\n"
                           "3 = to upd number_of_available_tickets\n"
                           "Input:"))
        if column == 1:
            new = int(input("New cinema id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cash register" set id_cinema = %s
where id_cr = %s;""", (new, id_cr,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            new = str(input("New number of seats: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Cash register" set
the_name_of_the_cashier = %s where id_cr = %s;""",
                               (new, id_cr,))
                print("[INFO] Data was successfully upd\n")
        elif column == 3:
            new = str(input("New number of seats: "))
            with connection.cursor() as cursor:
```

```
                cursor.execute("""update "Cash register" set
number_of_available_tickets = %s where id_cr = %s;""",
                               (new, id_cr,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 3)

    elif table == 4:
        id_t = int(input("Num of id: "))
        column = int(input("1 = to upd cash register id\n"
                           "2 = to upd hall id\n"
                           "3 = to upd film id\n"
                           "4 = to upd price\n"
                           "5 = to upd date and time\n"
                           "Input:"))
        if column == 1:
            new = int(input("New cash register id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set id_cr = %s where
id_ticket = %s;""", (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            new = int(input("New hall id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set id_hall = %s where
id_ticket = %s;""",
                               (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        elif column == 3:
            new = int(input("New film id: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set film_id = %s where
id_ticket = %s;""",
                               (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        elif column == 4:
            new = int(input("New price: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set price = %s where
id_ticket = %s;""",
                               (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        elif column == 5:
            date_entry = input("New date in YYYY-MM-DD format: ")
            year, month, day = map(int, date_entry.split('-'))
            time_entry = input("New time in HH:MM format: ")
            hour, minute = map(int, time_entry.split(':'))
            new = datetime.datetime(year, month, day, hour, minute)
            with connection.cursor() as cursor:
                cursor.execute("""update "Ticket" set date_and_time = %s
where id_ticket = %s;""",
                               (new, id_t,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 4)

    elif table == 5:
        id_f = int(input("Num of id: "))
        column = int(input("1 = to upd name\n"
                           "2 = to upd duration\n"
                           "Input:"))
        if column == 1:
```

```
            new = str(input("New name: "))
            with connection.cursor() as cursor:
                cursor.execute("""update "Film" set name = %s where id_film =
%s;""", (new, id_f,))
                print("[INFO] Data was successfully upd\n")
        elif column == 2:
            time_entry = input("New duration in HH:MM format: ")
            hour, minute = map(int, time_entry.split(':'))
            new = datetime.time(hour, minute)
            with connection.cursor() as cursor:
                cursor.execute("""update "Film" set duration = %s where
id_film = %s;""", (new, id_f,))
                print("[INFO] Data was successfully upd\n")
        else:
            print("\n[INFO] Error input, try again!\n")
            update(connection, 5)

    else:
        print("\n[INFO] Error input, try again!\n")


def delete(connection, table):

    if table == 1:
        id_c = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Cinema" WHERE id_cinema = %s;""",
(id_c,))
            print("[INFO] Data was successfully deleted\n")
    elif table == 2:
        id_h = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Hall" WHERE id_hall = %s;""",
(id_h,))
            print("[INFO] Data was successfully deleted\n")
    elif table == 3:
        id_cr = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Cash register" WHERE id_cr =
%s;""", (id_cr,))
            print("[INFO] Data was successfully deleted\n")
    elif table == 4:
        id_t = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Ticket" WHERE id_ticket = %s;""",
(id_t,))
            print("[INFO] Data was successfully deleted\n")
    elif table == 5:
        id_f = int(input("Num of id: "))
        with connection.cursor() as cursor:
            cursor.execute("""DELETE from "Film" WHERE id_film = %s;""",
(id_f,))
            print("[INFO] Data was successfully deleted\n")
    else:
        print("\n[INFO] Error input, try again!\n")


def select_table(connection, table):

    if table == 1:
        string = int(input("1 = to one str\n"
                           "2 = to all str\n"
                           "Input:"))
        if string == 1:
```

```
            id_c = int(input("Num of id: "))
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Cinema" where id_cinema =
%s;""", (id_c,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Cinema";""")
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            select_table(connection, 1)

    elif table == 2:
        string = int(input("1 = to one str\n"
                           "2 = to all str\n"
                           "Input:"))
        if string == 1:
            id_c = int(input("Num of id: "))
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Hall" where id_hall =
%s;""", (id_c,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Hall";""")
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            select_table(connection, 2)

    elif table == 3:
        string = int(input("1 = to one str\n"
                           "2 = to all str\n"
                           "Input:"))
        if string == 1:
            id_c = int(input("Num of id: "))
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Cash register" where id_cr =
%s;""", (id_c,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Cash register";""")
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            select_table(connection, 3)

    elif table == 4:
        string = int(input("1 = to one str\n"
                           "2 = to all str\n"
                           "Input:"))
        if string == 1:
            id_c = int(input("Num of id: "))
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Ticket" where id_ticket =
%s;""", (id_c,))
                return cursor.fetchmany(1)

        elif string == 2:
```

```
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Ticket";""")
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            select_table(connection, 4)

    elif table == 5:
        string = int(input("1 = to one str\n"
                           "2 = to all str\n"
                           "Input:"))
        if string == 1:
            id_f = int(input("Num of id: "))
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Film" where id_film =
%s;""", (id_f,))
                return cursor.fetchmany(1)

        elif string == 2:
            with connection.cursor() as cursor:
                cursor.execute("""SELECT * FROM "Film";""")
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            select_table(connection, 5)

    else:
        print("\n[INFO] Error input, try again!\n")


def search(connection, table):

    if table == 1:
        column = int(input("1 = to search id\n"
                           "2 = to search name or adress\n"
                           "Input:"))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cinema" WHERE %s <=
id_cinema and id_cinema <= %s;""", (a, b,))
                finish = time.time()
                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            txt = str(input("Input text: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cinema" WHERE
                    to_tsvector(name) || to_tsvector(adress)
                    @@ plainto_tsquery(%s);;""", (txt,))
                finish = time.time()
                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 1)

    elif table == 2:
        column = int(input("1 = to search hall id\n"
                           "2 = to search cinema id\n"
```

```
                            "3 = to search numbers of seats\n"
                            "Input:"))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Hall" WHERE %s <= id_hall
and id_hall <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Hall" WHERE %s <= id_cinema
and id_cinema <= %s;""", (a, b,))
                finish = time.time()
                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 3:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Hall" WHERE %s <=
number_of_seats and
                     number_of_seats <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 2)

    elif table == 3:
        column = int(input("1 = to search cash register id\n"
                            "2 = to search cinema id\n"
                            "3 = to search the_name_of_the_cashier\n"
                            "4 = to search number_of_available_tickets\n"
                            "Input:"))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cash register" WHERE %s <=
id_cr and id_cr <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
```

```
                cursor.execute("""SELECT * FROM "Cash register" WHERE %s <=
id_cinema and id_cinema <= %s;""",
                               (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 3:
            txt = str(input("Input name: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cash register" WHERE
to_tsvector(the_name_of_the_cashier)
                    @@ plainto_tsquery(%s);;""", (txt,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 4:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Cash register" WHERE %s <=
number_of_available_tickets and
                    number_of_available_tickets <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 3)

    elif table == 4:
        column = int(input("1 = to search ticket id\n"
                           "2 = to search cash register id\n"
                           "3 = to search hall id\n"
                           "4 = to search film id\n"
                           "5 = to search price\n"
                           "6 = to search date and time\n"
                           "Input:"))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Ticket" WHERE %s <=
id_ticket and id_ticket <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= id_cr
and id_cr <= %s;""", (a, b,))
                finish = time.time()
```

```
                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 3:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= id_hall
and id_hall <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 4:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= film_id
and film_id <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 5:
            a = float(input("Input start: "))
            b = float(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Ticket" WHERE %s <= price
and price <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 6:
            date_entry_a = input("New start date in YYYY-MM-DD format: ")
            year, month, day = map(int, date_entry_a.split('-'))
            time_entry_a = input("New start time in HH:MM format: ")
            hour, minute = map(int, time_entry_a.split(':'))
            a = datetime.datetime(year, month, day, hour, minute)

            date_entry_b = input("New finish date in YYYY-MM-DD format: ")
            year, month, day = map(int, date_entry_b.split('-'))
            time_entry_b = input("New finish time in HH:MM format: ")
            hour, minute = map(int, time_entry_b.split(':'))
            b = datetime.datetime(year, month, day, hour, minute)
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Ticket" WHERE %s <=
date_and_time and date_and_time <= %s;""",
                               (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 4)
```

```
    elif table == 5:
        column = int(input("1 = to search film id\n"
                           "2 = to search name\n"
                           "3 = to search duration\n"
                           "Input: "))
        if column == 1:
            a = int(input("Input start: "))
            b = int(input("Input finish: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Film" WHERE %s <= id_film
and id_film <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 2:
            txt = str(input("Input text: "))
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Film" WHERE
to_tsvector(name)
                    @@ plainto_tsquery(%s);;""", (txt,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()

        elif column == 3:

            time_entry_a = input("New start duration in HH:MM format: ")
            hour, minute = map(int, time_entry_a.split(':'))
            a = datetime.time(hour, minute)

            time_entry_b = input("New finish duration in HH:MM format: ")
            hour, minute = map(int, time_entry_b.split(':'))
            b = datetime.time(hour, minute)
            with connection.cursor() as cursor:
                start = time.time()
                cursor.execute("""SELECT * FROM "Film" WHERE %s <= duration
and duration <= %s;""", (a, b,))
                finish = time.time()

                print("\nSearch duration = ", finish - start)
                return cursor.fetchall()
        else:
            print("\n[INFO] Error input, try again!\n")
            search(connection, 5)

    else:
        print("\n[INFO] Error input, try again!\n")
```

Controller.py

```
from Model import *
from View import *


def menu(connection):
    x = int(input("1 to update\n"
                  "2 to delete\n"
                  "3 to insert\n"
                  "4 to select table\n"
```

```
                  "5 to search\n"
                  "6 to insert random\n"
                  "7 to end\n"
                  "Input: "))
    print("")
    if x == 1:
        table = tables()
        update(connection, table)

    elif x == 2:
        table = tables()
        delete(connection, table)

    elif x == 3:
        table = tables()
        insert(connection, table)

    elif x == 4:
        table = tables()
        f_t = select_table(connection, table)
        fetch(table, f_t)

    elif x == 5:
        table = tables()
        f_t = search(connection, table)
        fetch(table, f_t)

    elif x == 6:
        table = tables()

        insert_rand(connection, table)

    elif x == 7:
        return None

    else:
        print("\n[INFO] Error input, try again!\n")

    menu(connection)
```

View.py

```
def tables():
    choice_table = int(input("Select a table:\n"
                             "1 = Cinema \n"
                             "2 = Hall\n"
                             "3 = Cash register\n"
                             "4 = Ticket\n"
                             "5 = Film\n"
                             "Input: "))
    return choice_table


def fetch(table, f_table):
    if table == 1:
        for i in f_table:
            print("\n""id_cinema =", i[0])
            print("name =", i[1])
            print("adress =", i[2], "\n")
    elif table == 2:
        for i in f_table:
            print("\n""id_hall =", i[0])
            print("id_cinema =", i[1])
            print("number_of_seats =", i[2], "\n")
    elif table == 3:
```

```
        for i in f_table:
            print("\n""id_cr =", i[0])
            print("id_cinema =", i[1])
            print("the_name_of_the_cashier =", i[2])
            print("number_of_available_tickets =", i[3], "\n")
    elif table == 4:
        for i in f_table:
            print("\n""id_ticket =", i[0])
            print("id_cr =", i[1])
            print("id_hall =", i[2])
            print("id_film =", i[3])
            print("price =", i[4])
            print("date_and_time =", i[5], "\n")
    elif table == 5:
        for i in f_table:
            print("\n""id_film =", i[0])
            print("name =", i[1])
            print("duration =", i[2], "\n")
```

Мова програмування – Python 3.8
Середовище розробки програмного забезпечення – PyCharm Community Edition.
Середовище для відлагодження SQL-запитів до бази даних – PgAdmin4.

Використані бібліотеки:
psycopg – для роботи з PostgresSQL
time – для роботи з часом, а саму визначення швидкодії пошуку по базі даних
datetime – для приведення введених даних в типи для роботи з датою та часом
random – для роботи з рандомізацією, а саме вибір випадкової вулиці з масиву.