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**"

Виконала: студентка 3 курсу

групи КВ-93

Федорова А.Ю.

Перевірив:

Павловський В.І.

Київ – 2021

Мета роботи

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).

Інформація про програму

Посилання на репозиторій у GitHub з вихідним кодом програми та звітом: https://github.com/nutasanchik/DB_Lab2

Використана мова програмування: Python 3.10

Використані бібліотеки: psycopg2 (для зв’язку з СУБД), time (для виміру часу запиту пошуку для завдання 3), sys (для реалізації консольного інтерфейсу).

Інформація про обрану базу даних

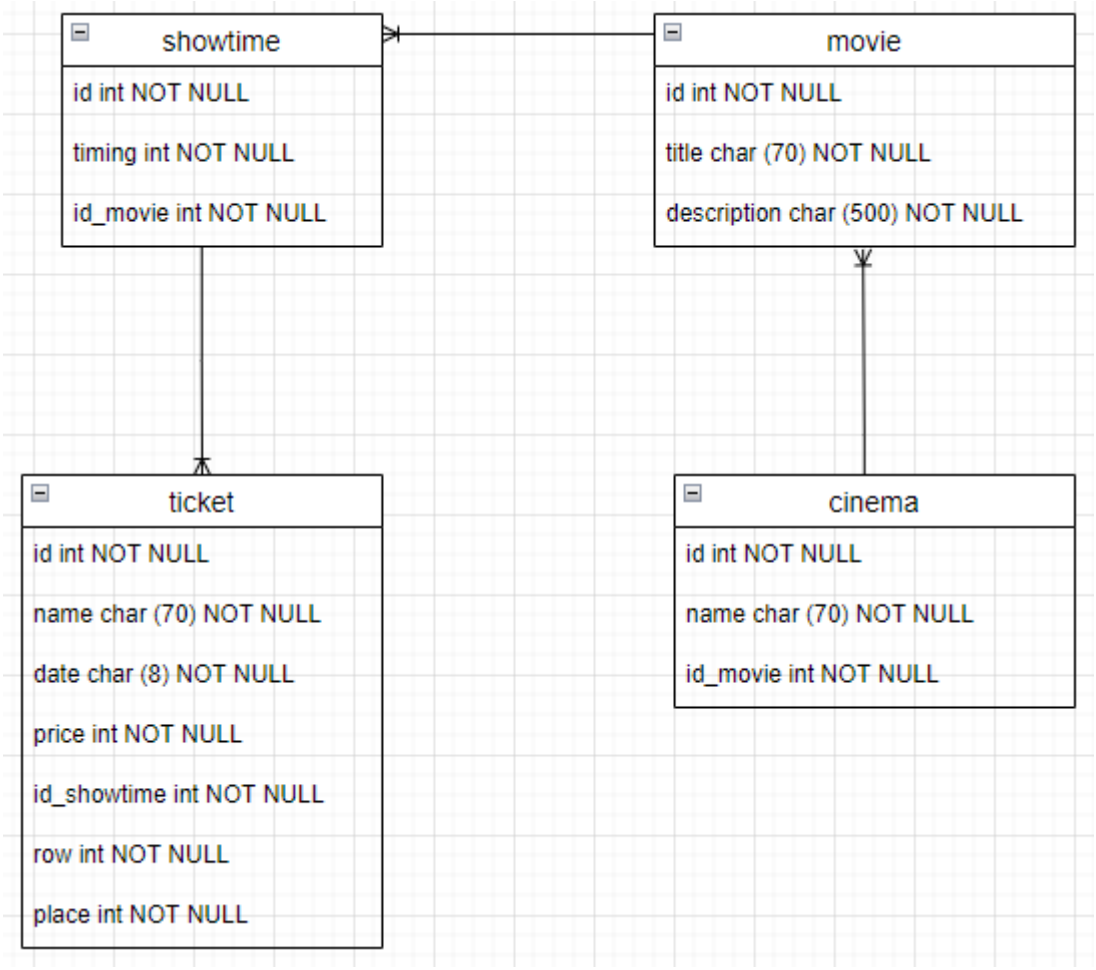

Рисунок 1. Схема бази даних

Таблиця 1. Опис структури БД

Відношення	**Атрибут**	**Тип**
Відношення «movie» Містить інформація про фільм	id – персональний номер фільму title – назва фільму description – короткий опис фільму	Числовий Текстовий (70) Текстовий (500)
Відношення «showtime» Містить інформацію про розклад сеансів	id – персональний номер сеансу timing – тривалість сеансу id_movie – номер фільму на даний сеанс	Числовий Числовий Числовий
Відношення «cinema» Містить інформацію про кінотеатр	id – персональний номер кінотеатру name – назва кінотеатру id_movie – персональний номер сеансу	Числовий Текстовий (70) Числовий
Відношення «ticket» Містить інформацію про придбаний квиток	id – персональний номер квитка name – ім’я глядача date – дата перегляду фільму price – вартість квитка id_showtime – персональний номер сеансу row – номер ряду у залі place – номер місця у залі	Числовий Текстовий (70) Текстовий (8) Числовий Числовий Числовий Числовий

Короткий опис структури БД «Кінотеатр»

Модель «сутність-зв’язок» галузі продажу квитків у кіно.

Модель має чотири сутності: movie, showtime, cinema, ticket.

Сутність movie – описує фільм, на який був куплений квиток. Кожен фільм має власний id, name та description.

Сутність showtime – описує час сеансу. Атрибутами даної сутності є id, timing та id_movie.

Сутність cinema – описує кінотеатр. Кожен кінотеатр має власний id, name, id_movie.

Сутність ticket – описує квиток, куплений глядачем. Кожен квиток має власний id, name, date, price, id_showtime, row, place.

Реалізація пункту 1 завдання до ЛР

print_table – виводить вміст заданої таблиці у вікно терміналу. У разі некоректності введених даних, у вікно терміналу виводиться помилка. Можливі аргументи: Cinema, Movie, Showtime, Ticket.

delete_record – видаляє запис з вказаним первинним ключем. У разі некоректності введених даних, у вікно терміналу виводиться помилка. Аргументи: table_name, key_name, key_value .

update_record – змінює поля, за заданим первинним ключем. У разі некоректності введених даних, у вікно терміналу виводиться помилка. Можливі аргументи:

Cinema id(int) name(str) id_movie(int)

Movie id(int) title(str) description(str)

Showtime id(int) timing(int) id_movie(int)

Ticket id(int) name(str) date(int) price(int) id_showtime(int) row(int) place(int)

insert_record – вставляє новий рядок у задану таблицю. У разі некоректності введених даних, у вікно терміналу виводиться помилка. Можливі аргументи:

Cinema id(int) name(str) id_movie(int)

Movie id(int) title(str) description(str)

Showtime id(int) timing(int) id_movie(int)

Ticket id(int) name(str) date(int) price(int) id_showtime(int) row(int) place(int)

Реалізація пункту 2 завдання до ЛР

generate_randomly –здійснює генерування n псевдорандомізованих записів у заданій таблиці. У разі некоректності введених даних, у вікно терміналу виводиться помилка. Аргументи: table_name, число записів, що мають бути створені.

Реалізація пункту 3 завдання до ЛР

search_records –реалізує пошук за 1 та більше атрибутами з заданих таблиць (від двох до чотирьох) і виводить у вікно терміналу результат пошуку (або нічого, якщо пошук не дав результатів) та час, за який було проведено запит. У разі некоректності введених даних, у вікно терміналу виводиться помилка. Початково потрібно вказати аргументи: table1_name table2_name table1_key table2_key або table1_name table2_name table3_name table1_key table2_key table3_key table13_key або table1_name table2_name table3_name table4_name table1_key table2_key table3_key table13_key table4_key table24_key де table13_key, table24_key – це зовнішні ключі, що зв’язують 1 та 3 таблицю, або 2 та 4. Після вказання цієї інформації потрібно буде вказати кількість атрибутів для пошуку, а тип пошуку, ім’я атрибуту (обов’язково з вказанням до якої таблиці з перелічених аргументів він відноситься: one.key_name, two.key_name, three.key_name або four.key_name), та значення (спочатку лівий кінець інтервалу, потім правий для числового пошуку та пошуку за датою, або рядок для пошуку за ключовим словом). Спочатку вказуються всі дані для першого атрибуту, потім для другого і т.д. до введеної кількості атрибутів

Завдання 1

Видалення:

Таблиця Cinema до видалення запису 5:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Cinema
SELECT * FROM public."Cinema"
Cinema table:
id: 1   name: Multiplex         id_movie: 1
-----------------------------------
id: 2   name: Multiplex         id_movie: 3
-----------------------------------
id: 3   name: Multiplex         id_movie: 2
-----------------------------------
id: 4   name: Multiplex         id_movie: 4
-----------------------------------
id: 5   name: Multiplex         id_movie: 1
-----------------------------------
```

Таблиця Cinema після видалення запису 5:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py delete_record Cinema id 5
select count(*) from public."Cinema" where id=5
select count(*) from public."Movie" where id=5
DELETE FROM public."Cinema" WHERE id=5;
```

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Cinema
SELECT * FROM public."Cinema"
Cinema table:
id: 1   name: Multiplex         id_movie: 1
-----------------------------------
id: 2   name: Multiplex         id_movie: 3
-----------------------------------
id: 3   name: Multiplex         id_movie: 2
-----------------------------------
id: 4   name: Multiplex         id_movie: 4
-----------------------------------
```

Таблиця Showtime до видалення запису 8:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Showtime
SELECT * FROM public."Showtime"
Showtime table:
id: 2   timing: 12      id_movie: 1

-------------------------------------
id: 3   timing: 14      id_movie: 2

-------------------------------------
id: 4   timing: 16      id_movie: 2

-------------------------------------
id: 5   timing: 18      id_movie: 3

-------------------------------------
id: 6   timing: 20      id_movie: 3

-------------------------------------
id: 7   timing: 21      id_movie: 4

-------------------------------------
id: 8   timing: 23      id_movie: 4

-------------------------------------
id: 1   timing: 16      id_movie: 1

-------------------------------------
```

Таблиця Showtime після спроби видалення запису 8:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py delete_record Showtime id 8
select count(*) from public."Showtime" where id=8
select count(*) from public."Movie" where id=8
DELETE FROM public."Showtime" WHERE id=8;
ОШИБКА:  UPDATE или DELETE в таблице "Showtime" нарушает ограничение внешнего ключа "fk_ticket_showtime" таблицы
 "Ticket"
DETAIL:  На ключ (id)=(8) всё ещё есть ссылки в таблице "Ticket".
```

Вставка:

Таблиця Movie до вставки запису 5:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Movie
SELECT * FROM public."Movie"
Movie table:
id: 1   title: The Eternals     description: Marvel Studios' Eternals features an exciting new team of Super Her
oes in the Marvel Cinematic Universe, ancient aliens who have been living on Earth in secret for thousands of ye
ars. Following the events of Avengers: Endgame, an unexpected tragedy forces them out of the shadows to reunite
against mankind's most ancient enemy, the Deviants.
-------------------------------------
id: 2   title: House of Gucci   description: House of Gucci is inspired by the shocking true story of the family
 behind the Italian fashion empire. When Patrizia Reggiani, an outsider from humble beginnings, marries into the
 Gucci family, her unbridled ambition begins to unravel the family legacy and triggers a reckless spiral of betr
ayal, decadence, revenge, and ultimately... murder.
-------------------------------------
id: 3   title: Dune     description: Paul Atreides, a brilliant and gifted young man born into a great destiny b
eyond his understanding, must travel to the most dangerous planet in the universe to ensure the future of his fa
mily and his people. As malevolent forces explode into conflict over the planet's exclusive supply of the most p
recious resource in existence, only those who can conquer their own fear will survive.
-------------------------------------
id: 4   title: Ghostbusters: From Beyond        description: When a single mom and her two kids arrive in a smal
l town, they begin to discover their connection to the original Ghostbusters and the secret legacy their grandfa
ther left behind.
-------------------------------------
```

Таблиця Movie після вставки запису 7:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py insert_record Movie 5 "Black Widow" "In Marvel Studios' action-packed spy thriller Black Widow, Natasha Romanoff aka Black Widow confronts the darker parts of her ledger when a dangerous conspiracy with ties to her past arises."
select count(*) from public."Movie" where id=5
insert into public."Movie" (id, title, description) VALUES (5, 'Black Widow', 'In Marvel Studios’ action-packed spy thriller Black Widow, Natasha Romanoff aka Black Widow confronts the darker parts of her ledger when a dangerous conspiracy with ties to her past arises.');
```

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Movie
SELECT * FROM public."Movie"
Movie table:
id: 1   title: The Eternals     description: Marvel Studios' Eternals features an exciting new team of Super Heroes in the Marvel Cinematic Universe, ancient aliens who have been living on Earth in secret for thousands of years. Following the events of Avengers: Endgame, an unexpected tragedy forces them out of the shadows to reunite against mankind's most ancient enemy, the Deviants.
-------------------------------------
id: 2   title: House of Gucci   description: House of Gucci is inspired by the shocking true story of the family behind the Italian fashion empire. When Patrizia Reggiani, an outsider from humble beginnings, marries into the Gucci family, her unbridled ambition begins to unravel the family legacy and triggers a reckless spiral of betrayal, decadence, revenge, and ultimately... murder.
-------------------------------------
id: 3   title: Dune     description: Paul Atreides, a brilliant and gifted young man born into a great destiny beyond his understanding, must travel to the most dangerous planet in the universe to ensure the future of his family and his people. As malevolent forces explode into conflict over the planet's exclusive supply of the most precious resource in existence, only those who can conquer their own fear will survive.
-------------------------------------
id: 4   title: Ghostbusters: From Beyond        description: When a single mom and her two kids arrive in a small town, they begin to discover their connection to the original Ghostbusters and the secret legacy their grandfather left behind.
-------------------------------------
id: 5   title: Black Widow      description: In Marvel Studios’ action-packed spy thriller Black Widow, Natasha Romanoff aka Black Widow confronts the darker parts of her ledger when a dangerous conspiracy with ties to her past arises.
-------------------------------------
```

Таблиця Ticket до вставки запису 7:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Ticket
SELECT * FROM public."Ticket"
Ticket table:
id: 1   name: Teodor Haas       date: 17.11     price: 100      id_showtime: 2  row: 3  place: 12
-------------------------------------
id: 2   name: Katina Blue       date: 17.11     price: 170      id_showtime: 2  row: 7  place: 8
-------------------------------------
id: 3   name: Edda Brooke       date: 18.11     price: 120      id_showtime: 4  row: 10         place: 11
-------------------------------------
id: 4   name: Vida Beck         date: 19.11
        price: 190      id_showtime: 5  row: 5  place: 9
-------------------------------------
id: 5   name: Eleonora Magyar   date: 19.11     price: 175      id_showtime: 7  row: 6  place: 14
-------------------------------------
id: 6   name: Humbert Mathers   date: 21.11     price: 110      id_showtime: 8  row: 5  place: 5
-------------------------------------
```

Таблиця Ticket після спроби вставки запису 7:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py insert_record Ticket 7 Zendaya 16.11 120 12 1 6
select count(*) from public."Showtime" where id=12
insert into public."Ticket" (id, name, date, price, id_showtime, row, place) VALUES (7, 'Zendaya', '16.11', '120
', '0', '1', '6');
ОШИБКА:  INSERT или UPDATE в таблице "Ticket" нарушает ограничение внешнего ключа "fk_ticket_showtime"
DETAIL:  Ключ (id_showtime)=(0) отсутствует в таблице "Showtime".
```

Зміна:

Таблиця Cinema до зміни запису 5:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Cinema
SELECT * FROM public."Cinema"
Cinema table:
id: 1   name: Multiplex         id_movie: 1

-------------------------------------
id: 2   name: Multiplex         id_movie: 3

-------------------------------------
id: 3   name: Multiplex         id_movie: 2

-------------------------------------
id: 4   name: Multiplex         id_movie: 4

-------------------------------------
id: 5   name: Multiplex         id_movie: 3

-------------------------------------
```

Таблиця Cinema після зміни запису 5:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py update_record Cinema 5 Multiplex 1
select count(*) from public."Cinema" where id=5
select count(*) from public."Movie" where id=1
UPDATE public."Cinema" SET name='Multiplex', id_movie='1' WHERE id=5;
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Cinema
SELECT * FROM public."Cinema"
Cinema table:
id: 1   name: Multiplex         id_movie: 1

-------------------------------------
id: 2   name: Multiplex         id_movie: 3

-------------------------------------
id: 3   name: Multiplex         id_movie: 2

-------------------------------------
id: 4   name: Multiplex         id_movie: 4

-------------------------------------
id: 5   name: Multiplex         id_movie: 1

-------------------------------------
```

Таблиця Showtime до зміни запису 8:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Showtime
SELECT * FROM public."Showtime"
Showtime table:
id: 2   timing: 12      id_movie: 1

-------------------------------------
id: 3   timing: 14      id_movie: 2

-------------------------------------
id: 4   timing: 16      id_movie: 2

-------------------------------------
id: 5   timing: 18      id_movie: 3

-------------------------------------
id: 6   timing: 20      id_movie: 3

-------------------------------------
id: 7   timing: 21      id_movie: 4

-------------------------------------
id: 8   timing: 23      id_movie: 4

-------------------------------------
id: 1   timing: 16      id_movie: 1

-------------------------------------
```

Таблиця Showtime після спроби зміни запису 8:

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py update_record Showtime 8 24 7
select count(*) from public."Showtime" where id=8
select count(*) from public."Movie" where id=7
Something went wrong (record with such id does not exist or inappropriate foreign key value)
```

Завдання 2

Рандомізація даних:

Cinema до змін

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Cinema
SELECT * FROM public."Cinema"
Cinema table:
id: 1   name: Multiplex         id_movie: 1

-----------------------------------
id: 2   name: Multiplex         id_movie: 3

-----------------------------------
id: 3   name: Multiplex         id_movie: 2

-----------------------------------
id: 4   name: Multiplex         id_movie: 4

-----------------------------------
id: 5   name: Multiplex         id_movie: 1

-----------------------------------
```

Cinema після рандомізації даних

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py generate_randomly Cinema 1
insert into public."Cinema"select (SELECT MAX(id)+1 FROM public."Cinema"), array_to_string(ARRAY(SELECT chr((97 + round
(random() * 25)) :: integer)                        FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), '')
, (SELECT id FROM public."Movie" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."Movie")-1))));
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Cinema
SELECT * FROM public."Cinema"
Cinema table:
id: 1   name: Multiplex         id_movie: 1
id: 2   name: Multiplex         id_movie: 3
-------------------------------------
id: 3   name: Multiplex         id_movie: 2
-------------------------------------
id: 4   name: Multiplex         id_movie: 4
-------------------------------------
id: 5   name: Multiplex         id_movie: 1
-------------------------------------
id: 6   name: laaloci   id_movie: 3
-------------------------------------
```

Movie до змін

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Movie
Movie table:
id: 1   title: The Eternals     description: Marvel Studios' Eternals features an exciting new team of Super Heroes in
the Marvel Cinematic Universe, ancient aliens who have been living on Earth in secret for thousands of years. Following
 the events of Avengers: Endgame, an unexpected tragedy forces them out of the shadows to reunite against mankind's mos
-------------------------------------
id: 2   title: House of Gucci   description: House of Gucci is inspired by the shocking true story of the family behind
 the Italian fashion empire. When Patrizia Reggiani, an outsider from humble beginnings, marries into the Gucci family,
 her unbridled ambition begins to unravel the family legacy and triggers a reckless spiral of betrayal, decadence, reve
nge, and ultimately... murder.
-------------------------------------
id: 3   title: Dune     description: Paul Atreides, a brilliant and gifted young man born into a great destiny beyond h
is understanding, must travel to the most dangerous planet in the universe to ensure the future of his family and his p
eople. As malevolent forces explode into conflict over the planet's exclusive supply of the most precious resource in e
xistence, only those who can conquer their own fear will survive.
-------------------------------------
id: 4   title: Ghostbusters: From Beyond        description: When a single mom and her two kids arrive in a small town,
 they begin to discover their connection to the original Ghostbusters and the secret legacy their grandfather left behi
nd.
-------------------------------------
id: 5   title: Black Widow      description: In Marvel Studios’ action-packed spy thriller Black Widow, Natasha Romanof
f aka Black Widow confronts the darker parts of her ledger when a dangerous conspiracy with ties to her past arises.
-------------------------------------
```

Movie після рандомізації даних

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py generate_randomly Movie 1
insert into public."Movie" select (SELECT (MAX(id)+1) FROM public."Movie"), array_to_string(ARRAY(SELECT chr((97 + roun
d(random() * 25)) :: integer)                          FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), ''
), array_to_string(ARRAY(SELECT chr((150 + round(random() * 25)) :: integer)                          FROM generate_ser
ies(1, FLOOR(RANDOM()*(10-4)+4):: integer)), '');
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Movie
SELECT * FROM public."Movie"
Movie table:
the Marvel Cinematic Universe, ancient aliens who have been living on Earth in secret for thousands of years. Following
 the events of Avengers: Endgame, an unexpected tragedy forces them out of the shadows to reunite against mankind's mos
-----------------------------------
id: 2   title: House of Gucci   description: House of Gucci is inspired by the shocking true story of the family behind
 the Italian fashion empire. When Patrizia Reggiani, an outsider from humble beginnings, marries into the Gucci family,
 her unbridled ambition begins to unravel the family legacy and triggers a reckless spiral of betrayal, decadence, reve
-----------------------------------
id: 3   title: Dune     description: Paul Atreides, a brilliant and gifted young man born into a great destiny beyond h
is understanding, must travel to the most dangerous planet in the universe to ensure the future of his family and his p
eople. As malevolent forces explode into conflict over the planet's exclusive supply of the most precious resource in e
xistence, only those who can conquer their own fear will survive.
-----------------------------------
id: 4   title: Ghostbusters: From Beyond        description: When a single mom and her two kids arrive in a small town,
 they begin to discover their connection to the original Ghostbusters and the secret legacy their grandfather left behi
nd.
-----------------------------------
id: 5   title: Black Widow      description: In Marvel Studios' action-packed spy thriller Black Widow, Natasha Romanof
f aka Black Widow confronts the darker parts of her ledger when a dangerous conspiracy with ties to her past arises.
-----------------------------------
id: 6   title: jllqw    description:  °¥¤
-----------------------------------
```

Showtime до змін

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Showtime
SELECT * FROM public."Showtime"
Showtime table:
id: 2   timing: 12      id_movie: 1

---------------------------------------
id: 3   timing: 14      id_movie: 2

---------------------------------------
id: 4   timing: 16      id_movie: 2

---------------------------------------
id: 5   timing: 18      id_movie: 3

---------------------------------------
id: 6   timing: 20      id_movie: 3

---------------------------------------
id: 7   timing: 21      id_movie: 4

---------------------------------------
id: 1   timing: 16      id_movie: 1

---------------------------------------
id: 8   timing: 24      id_movie: 2

---------------------------------------
```

Showtime після рандомізації даних

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py generate_randomly Showtime 1
insert into public."Showtime" select (SELECT MAX(id)+1 FROM public."Showtime"), FLOOR(RANDOM()*(100000-1)+1),(SELECT id
 FROM public."Movie" LIMIT 1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."Movie")-1))));
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Showtime
SELECT * FROM public."Showtime"
Showtime table:
id: 2   timing: 12      id_movie: 1
-----------------------------------
id: 3   timing: 14      id_movie: 2
-----------------------------------
id: 4   timing: 16      id_movie: 2
-----------------------------------
id: 5   timing: 18      id_movie: 3
-----------------------------------
id: 6   timing: 20      id_movie: 3
-----------------------------------
id: 7   timing: 21      id_movie: 4
-----------------------------------
id: 1   timing: 16      id_movie: 1
-----------------------------------
id: 8   timing: 24      id_movie: 2
-----------------------------------
id: 9   timing: 32910   id_movie: 4
-----------------------------------
```

Ticket до змін

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Ticket
SELECT * FROM public."Ticket"
Ticket table:
id: 1   name: Teodor Haas       date: 17.11     price: 100      id_showtime: 2  row: 3  place: 12
-----------------------------------
id: 2   name: Katina Blue       date: 17.11     price: 170      id_showtime: 2  row: 7  place: 8
-----------------------------------
id: 3   name: Edda Brooke       date: 18.11     price: 120      id_showtime: 4  row: 10         place: 11
-----------------------------------
id: 4   name: Vida Beck         date: 19.11
        price: 190      id_showtime: 5  row: 5  place: 9
-----------------------------------
id: 5   name: Eleonora Magyar   date: 19.11     price: 175      id_showtime: 7  row: 6  place: 14
-----------------------------------
id: 6   name: Humbert Mathers   date: 21.11     price: 110      id_showtime: 8  row: 5  place: 5
-----------------------------------
```

Ticket після рандомізації даних

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py generate_randomly Ticket 1
insert into public."Ticket" select (SELECT MAX(id)+1 FROM public."Ticket"), array_to_string(ARRAY(SELECT chr((97 + roun
d(random() * 25)) :: integer)                          FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4):: integer)), ''
), array_to_string(ARRAY(SELECT chr((150 + round(random() * 25)) :: integer)                          FROM generate_ser
ies(1, FLOOR(RANDOM()*(10-4)+4):: integer)), ''), FLOOR(RANDOM()*(100000-1)+1),(SELECT id FROM public."Showtime" LIMIT
1 OFFSET (round(random() *((SELECT COUNT(id) FROM public."Showtime")-1)))), FLOOR(RANDOM()*(100000-1)+1),FLOOR(RANDOM()
*(100000-1)+1);
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py print_table Ticket
SELECT * FROM public."Ticket"
Ticket table:
id: 1   name: Teodor Haas       date: 17.11     price: 100      id_showtime: 2  row: 3  place: 12
------------------------------------
id: 2   name: Katina Blue       date: 17.11     price: 170      id_showtime: 2  row: 7  place: 8
------------------------------------
id: 3   name: Edda Brooke       date: 18.11     price: 120      id_showtime: 4  row: 10         place: 11
------------------------------------
id: 4   name: Vida Beck         date: 19.11
        price: 190      id_showtime: 5  row: 5  place: 9
------------------------------------
id: 5   name: Eleonora Magyar   date: 19.11     price: 175      id_showtime: 7  row: 6  place: 14
------------------------------------
id: 6   name: Humbert Mathers   date: 21.11     price: 110      id_showtime: 8  row: 5  place: 5
------------------------------------
id: 7   name: wadpvol   date: ¡¨¬▯▯     price: 95980    id_showtime: 1  row: 21385      place: 23989
------------------------------------
```

Завдання 3

Пошук:

Пошук за двома таблицями

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py search_records Ticket Showtime id id
specify the number of attributes you`d like to search by: 2
specify the type of data you want to search for (numeric, string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.price
specify the left end of search interval: 90
specify the right end of search interval: 125
specify the type of data you want to search for (numeric, string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.timing
specify the left end of search interval: 11
specify the right end of search interval: 15
select * from public."Ticket" as one inner join public."Showtime" as two on one."id"=two."id" where 90<one.pric
e and one.price<125 and 11<two.timing and two.timing<15
--- 0.019988536834716797 seconds ---
search result:
3
Edda Brooke
18.11
120
4
10
11
3
14
2
------------------------------------
```

Пошук за трьома таблицями

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py search_records Showtime Movie Cinema id id id id
specify the number of attributes you`d like to search by: 3
specify the type of data you want to search for (numeric, string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.id_movie
specify the left end of search interval: 1
specify the right end of search interval: 3
specify the type of data you want to search for (numeric, string): string
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.title
specify the string you`d like to search for: Dune
specify the type of data you want to search for (numeric, string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: three.id
specify the left end of search interval: 0
specify the right end of search interval: 4
select * from public."Showtime" as one inner join public."Movie" as two on one."id"=two."id" inner join public.
"Cinema" as three on three."id"=one."id"where 1<one.id_movie and one.id_movie<3 and two.title LIKE 'Dune' and 0
<three.id and three.id<4
--- 0.003999233245849609 seconds ---
search result:
3
14
2
3
Dune
Paul Atreides, a brilliant and gifted young man born into a great destiny beyond his understanding, must travel
 to the most dangerous planet in the universe to ensure the future of his family and his people. As malevolent
forces explode into conflict over the planet's exclusive supply of the most precious resource in existence, onl
y those who can conquer their own fear will survive.
3
Multiplex
2
------------------------------------
```

Пошук за чотирма таблицями

```
PS F:\Ann\Uni\Third_year\BD\Lab2> python main.py search_records Showtime Movie Cinema Ticket id id id id id id
specify the number of attributes you`d like to search by: 4
specify the type of data you want to search for (numeric, string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: one.timing
specify the left end of search interval: 10
specify the right end of search interval: 20
specify the type of data you want to search for (numeric, string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: two.id
specify the left end of search interval: 0
specify the right end of search interval: 4
specify the type of data you want to search for (numeric, string): string
specify the name of key by which you`d like to perform search in form: table_number.key_name: three.name
specify the string you`d like to search for: Multiplex
```

```
specify the type of data you want to search for (numeric, string): numeric
specify the name of key by which you`d like to perform search in form: table_number.key_name: four.id_showtime
specify the left end of search interval: 0
specify the right end of search interval: 3
select * from public."Showtime" as one inner join public."Movie" as two on one."id"=two."id" inner join public.
"Cinema" as three on three."id"=one."id" inner join public."Ticket" as four on four."id"=two."id"where 10<one.t
iming and one.timing<20 and 0<two.id and two.id<4 and three.name LIKE 'Multiplex' and 0<four.id_showtime and fo
ur.id_showtime<3
--- 0.004996538162231445 seconds ---
search result:
1
16
1
1
The Eternals
Marvel Studios' Eternals features an exciting new team of Super Heroes in the Marvel Cinematic Universe, ancien
t aliens who have been living on Earth in secret for thousands of years. Following the events of Avengers: Endg
ame, an unexpected tragedy forces them out of the shadows to reunite against mankind's most ancient enemy, the
Deviants.
1
Multiplex
1
1
Teodor Haas
17.11
100
2
3
12
-----------------------------------
2
12
1
2
House of Gucci
House of Gucci is inspired by the shocking true story of the family behind the Italian fashion empire. When Pat
rizia Reggiani, an outsider from humble beginnings, marries into the Gucci family, her unbridled ambition begin
s to unravel the family legacy and triggers a reckless spiral of betrayal, decadence, revenge, and ultimately..
. murder.
2
Multiplex
3
2
Katina Blue
17.11
170
2
7
8
-----------------------------------
```

Завдання 4

Код програми:

Model.py

```
import psycopg2 as ps


class Model:
    def __init__(self):
```

```
        self.conn = None
        try:
            self.conn = ps.connect(
                dbname="Cinema_1",
                user='postgres',
                password="12345",
                host='127.0.0.1',
                port="5432",
            )
        except(Exception, ps.DatabaseError) as error:
            print("[INFO] Error while working with Postgresql", error)

    def request(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return True
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as error:
            print(error)
            self.conn.rollback()
            return False

    def get(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return cursor.fetchall()
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as error:
            print(error)
            self.conn.rollback()
            return False

    def get_el(self, req: str):
        try:
            cursor = self.conn.cursor()
            print(req)
            cursor.execute(req)
            self.conn.commit()
            return cursor.fetchone()
        except(Exception, ps.DatabaseError, ps.ProgrammingError) as error:
            print(error)
            self.conn.rollback()
            return False

    def count(self, table_name: str):
        return self.get_el(f"select count(*) from public.\"{table_name}\"")

    def find(self, table_name: str, key_name: str, key_value: int):
        return self.get_el(f"select count(*) from public.\"{table_name}\" where
{key_name}={key_value}")

    def max(self, table_name: str, key_name: str):
```

```
        return self.get_el(f"select max({key_name}) from
public.\"{table_name}\"")

    def min(self, table_name: str, key_name: str):
        return self.get_el(f"select min({key_name}) from
public.\"{table_name}\"")

    def print_cinema(self) -> None:
        return self.get(f"SELECT * FROM public.\"Cinema\"")

    def print_movie(self) -> None:
        return self.get(f"SELECT * FROM public.\"Movie\"")

    def print_showtime(self) -> None:
        return self.get(f"SELECT * FROM public.\"Showtime\"")

    def print_ticket(self) -> None:
        return self.get(f"SELECT * FROM public.\"Ticket\"")

    def delete_data(self, table_name: str, key_name: str, key_value) -> None:
        self.request(f"DELETE FROM public.\"{table_name}\" WHERE
{key_name}={key_value};")

    def update_data_cinema(self, key_value: int, name: str, id_movie: int) ->
None:
        self.request(f"UPDATE public.\"Cinema\" SET name=\'{name}\',
id_movie=\'{id_movie}\' WHERE id={key_value};")

    def update_data_movie(self, key_value: int, title: str, description: str) -
> None:
        self.request(f"UPDATE public.\"Movie\" SET title=\'{title}\',
description=\'{description}\' "
                     f"WHERE id={key_value};")

    def update_data_showtime(self, key_value: int, timing: int, id_movie: int)
-> None:
        self.request(f"UPDATE public.\"Showtime\" SET timing=\'{timing}\',
id_movie=\'{id_movie}\' WHERE id={key_value};")

    def update_data_ticket(self, key_value: int, name: str, date: str, price:
int, id_showtime: int, row: int,
                           place: int) -> None:
        self.request(f"UPDATE public.\"Ticket\" SET name=\'{name}\',
date=\'{date}\', price=\'{price}\', "
                     f"id_showtime=\'{id_showtime}\', row=\'{row}\',
place=\'{place}\' WHERE id={key_value};")

    def insert_data_cinema(self, key_value: int, name: str, id_movie: int) ->
None:
        self.request(f"insert into public.\"Cinema\" (id, name, id_movie) "
                     f"VALUES ({key_value}, \'{name}\', \'{id_movie}\');")

    def insert_data_movie(self, key_value: int, title: str, description: str) -
> None:
        self.request(f"insert into public.\"Movie\" (id, title, description) "
                     f"VALUES ({key_value}, \'{title}\', \'{description}\');")
```

```
    def insert_data_showtime(self, key_value: int, timing: int, id_movie: int)
-> None:
        self.request(f"insert into public.\"Showtime\" (id, timing, id_movie) "
                     f"VALUES ({key_value}, \'{timing}\', \'{id_movie}\');")

    def insert_data_ticket(self, key_value: int, name: str, date: str, price:
int, id_showtime: int, row: int,
                           place: int) -> None:
        self.request(f"insert into public.\"Ticket\" (id, name, date, price,
id_showtime, row, place) "
                     f"VALUES ({key_value}, \'{name}\', \'{date}\',
\'{price}\', \'{id_showtime}\', \'{row}\', \'{place}\');")

    def cinema_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Cinema\""
                         "select (SELECT MAX(id)+1 FROM public.\"Cinema\"), "
                         "array_to_string(ARRAY(SELECT chr((97 + round(random()
* 25)) :: integer) \
                         FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4)::
integer)), ''), "
                         "(SELECT id FROM public.\"Movie\" LIMIT 1 OFFSET "
                         "(round(random() *((SELECT COUNT(id) FROM
public.\"Movie\")-1))));")

    def movie_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Movie\" "
                         "select (SELECT (MAX(id)+1) FROM public.\"Movie\"), "
                         "array_to_string(ARRAY(SELECT chr((97 + round(random()
* 25)) :: integer) \
                         FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4)::
integer)), ''), "
                         "array_to_string(ARRAY(SELECT chr((97 + round(random()
* 25)) :: integer) \
                         FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4)::
integer)), ''); ")

    def showtime_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Showtime\" "
                         "select (SELECT MAX(id)+1 FROM public.\"Showtime\"), "
                         "FLOOR(RANDOM()*(100000-1)+1),"
                         "(SELECT id FROM public.\"Movie\" LIMIT 1 OFFSET "
                         "(round(random() *((SELECT COUNT(id) FROM
public.\"Movie\")-1))));")

    def ticket_data_generator(self, times: int) -> None:
        for i in range(times):
            self.request("insert into public.\"Ticket\" "
                         "select (SELECT MAX(id)+1 FROM public.\"Ticket\"), "
                         "array_to_string(ARRAY(SELECT chr((97 + round(random()
* 25)) :: integer) \
                         FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4)::
integer)), ''), "
```

```
                          "array_to_string(ARRAY(SELECT chr((150 +
round(random() * 25)) :: integer) \
                          FROM generate_series(1, FLOOR(RANDOM()*(10-4)+4)::
integer)), ''), "
                          "FLOOR(RANDOM()*(100000-1)+1),"
                          "(SELECT id FROM public.\"Showtime\" LIMIT 1 OFFSET "
                          "(round(random() *((SELECT COUNT(id) FROM
public.\"Showtime\")-1)))), "
                          "FLOOR(RANDOM()*(100000-1)+1),"
                          "FLOOR(RANDOM()*(100000-1)+1);")

    def search_data_two_tables(self, table1_name: str, table2_name: str,
table1_key, table2_key,
                               search: str):
        return self.get(f"select * from public.\"{table1_name}\" as one inner
join public.\"{table2_name}\" as two "
                        f"on one.\"{table1_key}\"=two.\"{table2_key}\" "
                        f"where {search}")

    def search_data_three_tables(self, table1_name: str, table2_name: str,
table3_name: str,
                                 table1_key, table2_key, table3_key,
table13_key,
                                 search: str):
        return self.get(f"select * from public.\"{table1_name}\" as one inner
join public.\"{table2_name}\" as two "
                        f"on one.\"{table1_key}\"=two.\"{table2_key}\" inner
join public.\"{table3_name}\" as three "
                        f"on three.\"{table3_key}\"=one.\"{table13_key}\""
                        f"where {search}")

    def search_data_all_tables(self, table1_name: str, table2_name: str,
table3_name: str, table4_name: str,
                               table1_key, table2_key, table3_key, table13_key,
                               table4_key, table24_key,
                               search: str):
        return self.get(f"select * from public.\"{table1_name}\" as one inner
join public.\"{table2_name}\" as two "
                        f"on one.\"{table1_key}\"=two.\"{table2_key}\" inner
join public.\"{table3_name}\" as three "
                        f"on three.\"{table3_key}\"=one.\"{table13_key}\" inner
join public.\"{table4_name}\" as four "
                        f"on four.\"{table4_key}\"=two.\"{table24_key}\""
                        f"where {search}")
```

Програмна частина model.py слугує головною частиною, оскільки має доступ до бази даних. Бібліотека Python – psycopg2 надає дану можливість.

Короткий опис функцій model.py

request : робить запит до БД, у разі успішного виконання повертає True, у разі невдачі – False.

get : усі дані що було взято з запитів SELECT, у разі невдачі – False.

get_el : повертає тільки перший запис, у разі невдачі – False.

count : повертає кількість усіх записів таблиці.

find : відповідно заданій умові шукає записи у таблиці і повертає їх кількість, у разі невдачі – False.

max, min : повертають максимальне і мінімальне значення зазначеного ключа у таблиці.

print_(cinema/movie/showtime/ticket): виводять на екран відповідні таблиці.

delete_data : видаляє відповідний запис у заданій таблиці.

update_data_(cinema/movie/showtime/ticket) : оновлює відповідний запис у заданій таблиці.

insert_data_(cinema/movie/showtime/ticket) : додає відповідний запис у задану таблицю.

(cinema/movie/showtime/ticket)_data_generator : додає рандомізований запис у задану таблицю.

search_data_(two/three/all)_tables : здійснює пошук по двум/трьом/чотирьом заданим таблицям.