

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Розрахунково-графічна робота

з дисципліни

«Бази даних та засоби управління»

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

Виконала студентка:

Михайліченко С.В.

групи: КВ-11

Перевірив: Петрашенко А. В.

Оцінка:

Київ – 2023

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

Загальне завдання роботи полягає у наступному:

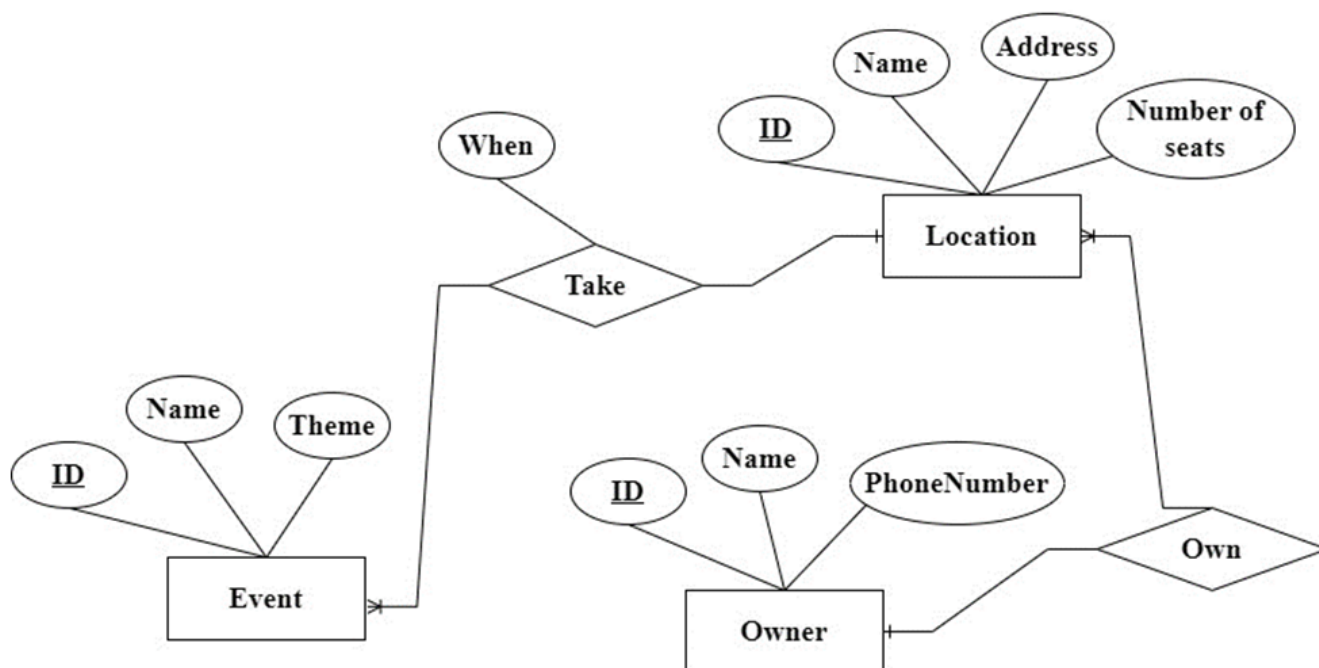
1. Реалізувати функції перегляду, внесення, редагування та видалення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами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

Репозиторій на github: <https://github.com/Sonneetta/database/tree/rgr>

Для виконання цієї розрахунково-графічної роботи була виконана мова програмування C#, pgAdmin4 та фреймворк Entity Framework(для того, щоб пов'язати програму на C# із базою даних)

Інформація про базу даних

Розробка моделі «сутність-зв'язок» предметної галузі для проектування бази даних «A platform for booking and managing venues for events». Предметна галузь – 65 «Платформа для бронювання та управління майданчиками для подій.».



Малюнок 1. ER-діаграма побудована за нотацією «Crow`s foot»

Сутності з описом призначення:

Предметна галузь «A platform for booking and managing venues for events» включає в себе 3 сутності, кожна сутність містить декілька атрибутів:

1. Event (ID, Name, Theme).
2. Location (ID, Name, Address, Number of seats).
3. Owner (ID, Name, PhoneNumber).

Сутність Event описує подію, яка відбудеться на певній локації. Кожна подія має свій ідентифікатор ID, а також має назву та тематику.

Сутність Location описує місце, де певна подія відбуватиметься. Кожна локація має свій ідентифікатор, назву, адресу та кількість можливих відвідувачів за один раз.

Сутність Owner описує власника місця, де відбувається певна подія. Кожен власник має свій ідентифікатор, ім'я та номер телефону.

Зв'язки між сутностями:

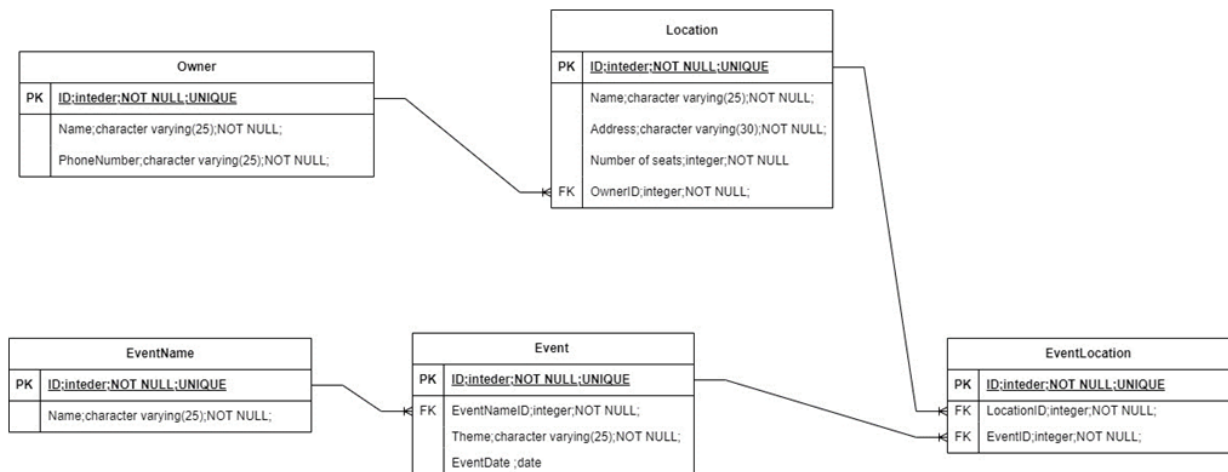
Зв'язок між Event та Location:

Кожна подія має своє місце проведення. Зв'язок 1:N – на одній локації можуть відбуватися декілька подій(наприклад в ресторані можуть замовити столики під святкування різних подій).

Зв'язок між Location та Owner:

Кожна локація має свого власника. Оскільки один власник може мати декілька локацій, зв'язок 1:N.

Схема бази даних PostgreSQL



Малюнок 2. Схема бази даних у графічному вигляді

Функціональні залежності:

1. Event (ID, Name, Theme).
 - ID→Name
 - ID→Theme
 - ID→EventID
2. Location (ID, Name, Address, Number of seats).
 - ID→Name
 - ID→Address
 - ID→Number of seats

ID → OwnerID

3. Owner (ID, Name, PhoneNumber).

ID → Name

ID → PhoneNumber

4. EventName (ID, Name).

ID → Name

5. EventLocation (ID).

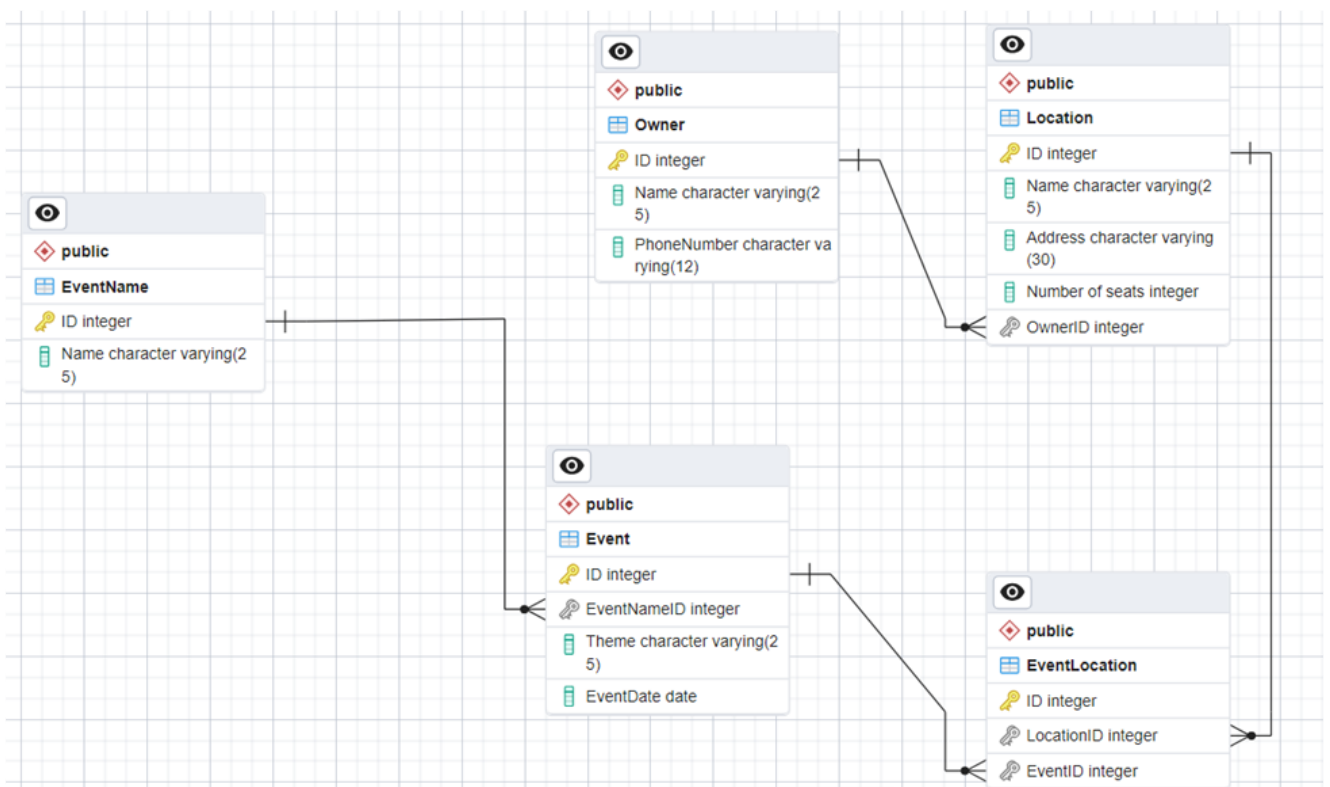
ID → LocationID

ID → EventID

Схема бази даних відповідає 1НФ, тому що значення в кожній комірці таблиці є атомарними, кожне поле таблиці є неподільним, кожен рядок є унікальним, немає повторень рядків.

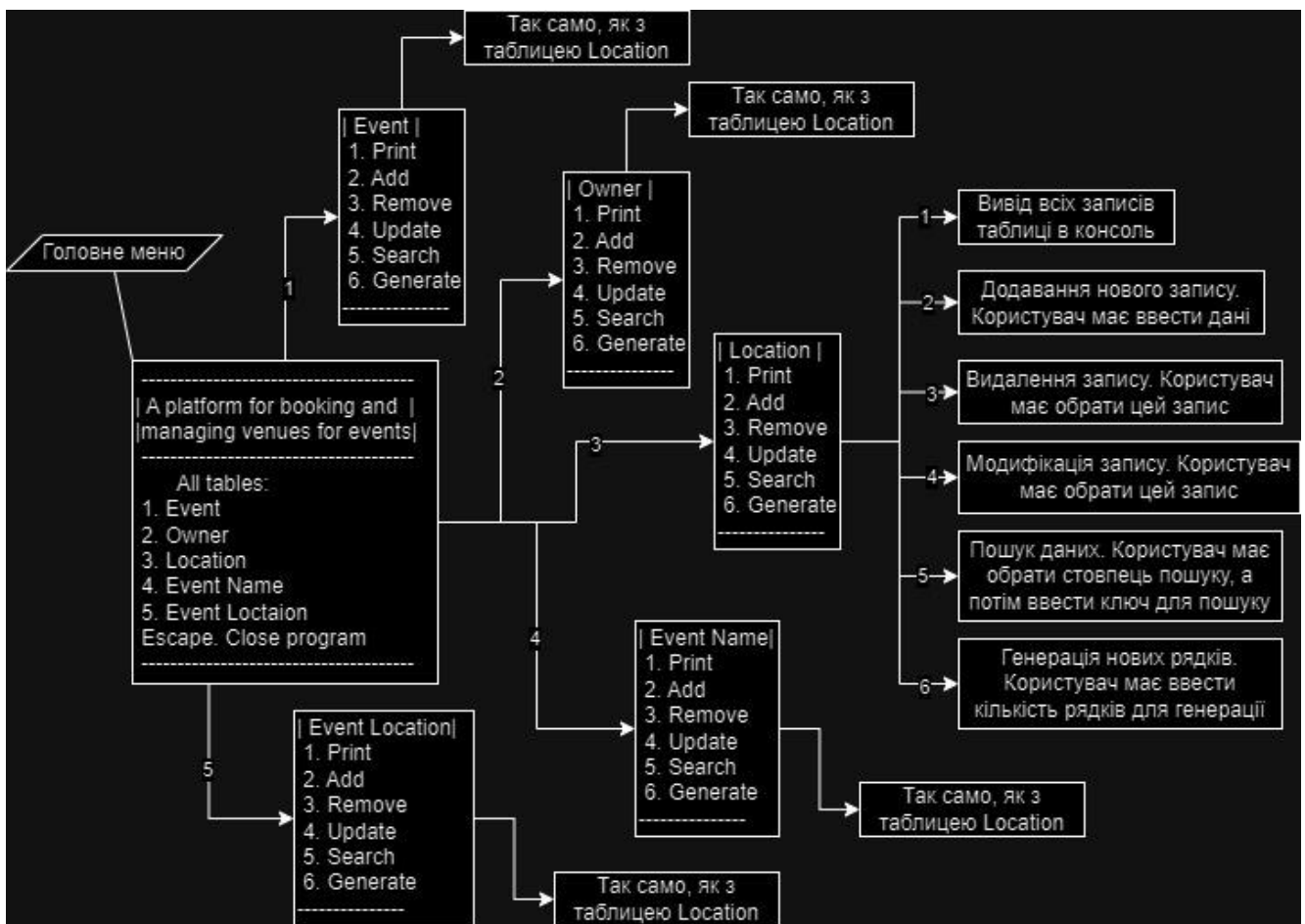
Схема бази даних відповідає 2НФ, бо вона відповідає 1НФ та кожен неключовий атрибут залежить від первинного повного ключа, отже первинний ключ одразу визначає запис та не є надмірним.

Схема бази даних відповідає 3НФ, тому що вона відповідає 2НФ та кожен неключовий атрибут не є транзитивно залежним від кожного кандидатного ключа. В таблицях нема не ключового поля, яке залежить від значення іншого не ключового поля.



Малюнок 3. Схема бази даних у pgAdmin4

Схема меню користувача з описом функціональності



Навігація по меню відбувається з клавіатури, в основному за допомогою клавіш 1-6. Всюди, де треба вибрати опцію меню, треба просто натиснути на потрібну клавішу. Не треба спочатку натискати цю клавішу, а потім натискати Enter. І навпаки, якщо треба ввести саме дані (наприклад, для пошуку, або для введення нових даних в таблицю), то треба ввести дані і натиснути Enter.

Головне меню відображає всі 5 таблиць, а також можливість завершити програму:

1. **Event** – таблиця Event
2. **Owner** – таблиця Owner
3. **Patient** – таблиця Patient
4. **Event Name** – таблиця EventName
5. **Event Location** – таблиця EventLocation
6. **Escape** – закрити програму

Кожна таблиця має підменю, яке має такі пункти:

- **Print** – перегляд всіх записів таблиці. Робиться SQL запит до відповідної таблиці, а результат цього запиту буде виведено в консоль.
- **Add** – вставка нового запису в таблицю. Користувач має внести всі необхідні дані, після чого ці дані будуть проходити валідацію. Якщо вставка цих даних не є можливою, в консоль буде видане повідомлення про помилку. Якщо ж дані є валідними і все добре, то рядок буде вставлено, а після вставки буде виведено в консоль повідомлення про успішну операцію.
- **Remove** – видалення запису з таблиці. Користувач має обрати індекс того запису, який йому треба видалити. Відбувається перевірка введенного

індексу, чи є він в таблиці, чи ні. Якщо його немає, то програма виведе в консоль повідомлення про помилку. Якщо все добре, то запис буде видалено. Після видалення в консоль буде виведено строку, що видалення пройшло успішно.

- **Update** – зміна існуючого запису таблиці. На початку буде виведено в консоль записи всіх записів таблиці. Це робиться для того, щоб користувач, як і в випадку з видаленням, міг обрати той рядок, який він хоче модифікувати. Після цього користувач вводить нові дані про цей запис. Потім відбувається валідація даних, які ввів користувач. Якщо, наприклад, користувач ввів неіснуючий індекс, або ж введені нові дані не є коректними, то в консоль буде виведено повідомлення про помилку.
- **Search** – пошук записів таблиці. В цьому пункті меню користувач має обрати те поле, по якому йому треба шукати запис. Після того, як він робить вибір, користувач має внести або ключ для пошуку(частину шукової строки, або ж повну), або потрібний діапазон значень. Залежить від ситуації, від типу обраного стовпця таблиці.
- **Generate** – заповнення таблиці випадковими даними. Користувач має ввести ту кількість рядків, яку він хоче отримати після генерації.

Завдання №1

Забезпечити можливість введення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв'язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати вилучення рядків за умови наявності даних у підлеглий таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні внесення нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.

В цій роботі реалізовані 4 основні CRUD-операції для роботи з кожною з таблиць: створення нових записів, читання даних з таблиці, модифікація даних, видалення даних. Також були реалізовані методи для пошуку потрібних даних в таблицях.

1. Вилучення даних

Вилучення даних відбувається SQL-запитом, завдяки команді SELECT. Ось приклад вилучення даних з таблиці **Event**:

```

1 -----
| Events: |
|-----|
| Id|      Event name|      Event theme|Event date|
|-----|
| 1|      Party|      Halloween|30.10.2025|
| 2|      Concert|      Musical|02.07.2025|
| 3|      Party|      Birthday|12.09.2025|
| 4|      Concert|      Pop|25.01.2025|
| 5|      Wedding|      Gothic|05.03.2026|
| 6| Charity evening|      Gallery|07.07.2026|
| 7| Charity evening|      War|05.01.2024|
| 9|      Concert|      WCY|15.08.2023|
|10|      Party|      GNN|22.11.2023|
|11|      Wedding|      VTI|09.12.2023|
|12|      Party|      GCU|09.08.2023|
|-----|

```

В кодї цей SQL-запит виглядає наступним чином:

```

@"SELECT ""Event"". ""ID"" AS Id,
""EventName"". ""Name"" AS EventName,
""Event"". ""Theme"" AS Theme,
""Event"". ""EventDate"" AS EventDate
FROM ""Event"", ""EventName""
WHERE ""Event"". ""EventNameId"" = ""EventName"". ""ID""
ORDER BY ""Event"". ""ID"" ASC"

```

Як можна помітити, відбувається звернення одразу до двох таблиць, адже в таблиці **Event** присутній зовнішній ключ.

Ще один приклад виведення даних з іншої таблиці. На цей раз це таблиця **Location**:

```

1 -----
| Location: |
|-----|
| Id|      Name|      Address|Seats|      Owner|
|-----|
| 1|      Star Lounge|      123 Galaxy Street| 200|      Ethan Turner|
| 2|      Mystic Gardens|      789 Enchanted Avenu| 150|      Olivia Bennett|
| 3|      Neon Plaza|      456 Luminescent Lane| 300|      Jackson Mitchell|
| 4|      Sapphire Lounge|      567 Celestial Boulevard| 250|      Ava Parker|
| 5|      Velder Sky Bar|      890 Stellar Drive| 180|      Olivia Bennett|
| 6|      CIC MSA|      MXI QBI 792| 79|      Jackson Mitchell|
| 7|      WIL XRE|      KOX PXO 790| 826|      Ethan Turner|
|-----|

```

В кодї цей SQL-запит виглядає наступним чином:

```

@"SELECT
""Location"". ""ID"" as Id,
""Location"". ""Name"" as Name,
""Location"". ""Address"" as Address,
""Location"". ""NumberOfSeats"" as NumberOfSeats,
""Owner"". ""Name"" as Owner
FROM ""Location"", ""Owner""
WHERE ""Location"". ""OwnerId"" = ""Owner"". ""ID""
ORDER BY ""Location"". ""ID"" ASC"

```


2. Видалення даних

Вилучення даних відбувається SQL-запитом, завдяки команді DELETE. Видалення відбувається за обраним ID того запису, який користувач хоче видалити. Ось приклад видалення рядку з таблиці **EventLocation**:

```
3 -----
| Event Location: |
|-----|
| Id|          Event name|          Location name|
|-----|
| 5|          Wedding|      Sapphire Lounge|
| 4|    Charity evening|      Sapphire Lounge|
| 3|          Party|          Neon Plaza|
| 2|          Concert|          Star Lounge|
| 1|          Party|      Mystic Gardens|
| 6|          Party|          Star Lounge|
| 7|          Concert|      Volver Sky Bar|
| 8|          JBG LGQ|          WIL XRE|
|-----|
Choose the ID of the row which you want to remove.
8
You have removed this row from table.
```

SQL-запит для цього виглядає наступним чином:

```
$"DELETE FROM \"EventLocation\" WHERE \"ID\" = {index};
```

Можна переконавшись, що рядок був дійсно видалений. Для цього треба вивести всі дані таблиці:

Id	Event name	Location name

5	Wedding	Sapphire Lounge
4	Charity evening	Sapphire Lounge
3	Party	Neon Plaza
2	Concert	Star Lounge
1	Party	Mystic Gardens
6	Party	Star Lounge
7	Concert	Volver Sky Bar

Як можна помітити, останнього рядку в таблиці вже немає.

Бувають такі ситуації, коли видалення даних неможливо. Наприклад, коли поле таблиці пов'язано зовнішнім ключем із іншим полем іншої таблиці. В таких випадках видалення не є можливим. Спробуємо видалити запис з таблиці **EventName**:

```
3 -----
| Event Name: |
|-----|
| Id|          Event name|
|-----|
| 1|          Party|
| 2|          Concert|
| 3|          Wedding|
| 4|    Charity evening|
| 5|          Ball|
| 6|          XCR TNY|
| 7|          JBG LGQ|
| 8|          VCY PXF|
|-----|
Choose the ID of Event Name which you want to remove.
2
You can not remove this row. It is connected with other row
Press any key to continue
```

Як можна помітити, видалення не вдалось, оскільки цей рядок пов'язаний з іншими рядками іншої таблиці, а саме в цих випадках:

```
1 -----
| Events: |
|-----|
| Id|      Event name|      Event theme|Event date|
|-----|
| 1|      Party|      Halloween|30.10.2025|
| 2|      Concert|      Musical|02.07.2025|
| 3|      Party|      Birthday|12.09.2025|
| 4|      Concert|      Pop|25.01.2025|
| 5|      Wedding|      Gothic|05.03.2026|
| 6|      Charity evening|      Gallery|07.07.2026|
| 7|      Charity evening|      War|05.01.2024|
| 9|      Concert|      WCY|15.08.2023|
|10|      Party|      GNN|22.11.2023|
|11|      Wedding|      VTI|09.12.2023|
|12|      Party|      GCU|09.08.2023|
|-----|
```

Також передбачається перевірка, чи є введене користувачем значення ID реальним:

```
3 -----
| Event Name: |
|-----|
| Id|      Event name|
|-----|
| 1|      Party|
| 2|      Concert|
| 3|      Wedding|
| 4|      Charity evening|
| 5|      Ball|
| 6|      XCR TNY|
| 7|      JBG LGQ|
| 8|      VCY PXF|
|-----|
Choose the ID of Event Name which you want to remove.
34
You can not remove this row. Invalid ID
```

Як можна побачити, ID зі значенням 34 не знаходиться в даній таблиці. Тому програма видала повідомлення про помилку. Попри це, програма продовжує працювати.

Приклади запитів для видалення:

```
$"DELETE FROM \"Location\" WHERE \"ID\" = {index};"
```

```
$"DELETE FROM \"Owner\" WHERE \"ID\" = {index};"
```

3. Вставка даних

Вставка даних відбувається SQL-запитом, завдяки команді INSERT. Ось приклад вставки даних в таблицю **Owner**:

```
2
Input data about new owner:
Name:
Mike Loyson
Phone number:
787-1121
You have added a new row into table.
Press any key to continue
```

SQL-запит виглядає наступним чином:

```
$"INSERT INTO "Owner" VALUES ('{owner.Id}',
'{owner.Name}', '{owner.Phone}')
```

Як і було описано в попередніх пунктах, спочатку користувач має внести дані, які будуть внесені в таблицю. Також ці дані проходять валідацію. Якщо, наприклад,

ввести в таблицю **Location** ім'я такого власника, якого немає в таблиці власників **Owner**, то буде виведено повідомлення про помилку:

Всі існуючі власники:

```
1 -----
| Owner: |
|-----|
| Id|          Name|          Phone|
|-----|
| 1|      Ethan Turner|      876-5432|
| 2|    Olivia Bennett|      234-5678|
| 3|   Jackson Mitchell|      987-6543|
| 4|      Ava Parker|      123-4567|
| 5|    Morgan Brown|      897-1311|
| 6|        OGA XBK|      971-3075|
| 7|        GTI JNN|      750-3103|
| 8|      Mike Loyson|      787-1121|
|-----|
```

Спроба використати неіснуючого власника в таблицю **Location** :

```
2
Input data about new location:
Name:
Flower Park
Address:
Kovalskiy Lane 5
Number of seats:
200
Owner:
Kkkskskk skkalw
Error. You have entered unexisting owner's name
```

Як можна побачити, дані не пройшли валідацію. Але якщо надати коректні дані, то вставка відбудеться:

```
Input data about new location:
Name:
Park of flowers
Address:
Kovalskiy lane
Number of seats:
235
Owner:
Ethan Turner
You have added a new row into table.

Press any key to continue
```

Перевіримо, чи відбулася вставка в таблицю:

```
1 -----
| Location: |
|-----|
| Id|          Name|          Address|Seats|          Owner|
|-----|
| 1|      Star Lounge|      123 Galaxy Street| 200|      Ethan Turner|
| 2|    Mystic Gardens|      789 Enchanted Avenu| 150|    Olivia Bennett|
| 3|      Neon Plaza|      456 Luminescent Lane| 300|   Jackson Mitchell|
| 4|    Sapphire Lounge|      567 Celestial Boulevard| 250|        Ava Parker|
| 5|    Volver Sky Bar|      890 Stellar Drive| 180|    Olivia Bennett|
| 6|        CIC MSA|      MXI QBI 792| 79|   Jackson Mitchell|
| 7|        WIL XRE|      KOX PXO 790| 826|      Ethan Turner|
| 8|      Park of flowers|      Kovalskiy lane| 235|      Ethan Turner|
|-----|
```

Як можна помітити, вставка відбулася успішно. Для здійснення цієї операції було використано наступний SQL-запит:

```
$"INSERT INTO "Location" VALUES (
    '{location.Id}', '{location.Name}', '{location.Address}',
    {location.NumberOfSeats}, {ownerId})";
```

4. Модифікація даних

Модифікація даних відбувається SQL-запитом, завдяки команді UPDATE. Ось приклад модифікації запису таблиці **Location**:

```

| Id | Name | Address | Seats | Owner |
|----|-----|-----|-----|-----|
| 1 | Star Lounge | 123 Galaxy Street | 200 | Ethan Turner |
| 2 | Mystic Gardens | 789 Enchanted Avenu | 150 | Olivia Bennett |
| 3 | Neon Plaza | 456 Luminescent Lane | 300 | Jackson Mitchell |
| 4 | Sapphire Lounge | 567 Celestial Boulevard | 250 | Ava Parker |
| 5 | Volver Sky Bar | 890 Stellar Drive | 180 | Olivia Bennett |
| 6 | CIC MSA | MXI QBI 792 | 79 | Jackson Mitchell |
| 7 | WIL XRE | KOX PXO 790 | 826 | Ethan Turner |
| 8 | Park of flowers | Kovalskiy lane | 235 | Ethan Turner |
|----|-----|-----|-----|-----|
Choose the ID of location which you want to update.
7
Input new data about location:
Name:
New location
Address:
14 Some Street
Number of seats:
300
Owner:
Ava Parker
You have updated this row.
Press any key to continue

```

Запит для модифікації:

```
$"UPDATE "Location" SET "Name"='{location.Name}',
"Address"='{location.Address}', "NumberOfSeats"={location.NumberOfSeats},
"OwnerId"={ownerId} WHERE "ID" = {id}"
```

Результат:

Location:				
Id	Name	Address	Seats	Owner
1	Star Lounge	123 Galaxy Street	200	Ethan Turner
2	Mystic Gardens	789 Enchanted Avenu	150	Olivia Bennett
3	Neon Plaza	456 Luminescent Lane	300	Jackson Mitchell
4	Sapphire Lounge	567 Celestial Boulevard	250	Ava Parker
5	Volver Sky Bar	890 Stellar Drive	180	Olivia Bennett
6	CIC MSA	MXI QBI 792	79	Jackson Mitchell
7	New location	14 Some Street	300	Ava Parker
8	Park of flowers	Kovalskiy lane	235	Ethan Turner

Якщо ввести некоректні дані під час введення нових даних, програма повідомляє про помилку:

```

Location: |
-----
| Id|          Name|          Address|Seats|          Owner|
-----
| 1|    Star Lounge|    123 Galaxy Street| 200|    Ethan Turner|
| 2|    Mystic Gardens|    789 Enchanted Avenu| 150|    Olivia Bennett|
| 3|    Neon Plaza|    456 Luminescent Lane| 300|    Jackson Mitchell|
| 4|    Sapphire Lounge|    567 Celestial Boulevard| 250|    Ava Parker|
| 5|    Volver Sky Bar|    890 Stellar Drive| 180|    Olivia Bennett|
| 6|    CIC MSA|    MXI QBI 792| 79|    Jackson Mitchell|
| 7|    New location|    14 Some Street| 300|    Ava Parker|
| 8|    Park of flowers|    Kovalskiy lane| 235|    Ethan Turner|
-----
Choose the ID of location which you want to update.
1
Input new data about location:
Name:
Star Lounge
Address:
123 New Street
Number of seats:
300
Owner:
Other Owner
Error. You have entered unexisting owner.
Press any key to continue

```

Як можна помітити, дані не пройшли валідацію. Отже, рядок не було змінено.

Завдання №2

*Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не мовою програмування, а відповідним SQL-запитом!** Кількість даних для генерування має вводити користувач з клавіатури. Для тесту взяти 100 000 записів для однієї-двох таблиць. Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (foreign key).*

Приклад генерації 100 стовпців в таблицю Event.

SQL запит:

```

$"INSERT INTO "Event" VALUES ('{id}', '{nameid}',
concat(chr(trunc(65+random()*25)::int),
chr(trunc(65+random()*25)::int), chr(trunc(65+random()*25)::int),
chr(trunc(65+random()*25)::int)),chr(trunc(65+random()*25)::int),
(current_date - random() * interval '365 days')::date)";

```

Результат генерації:

```

1
| Event |
1. Print
2. Add
3. Remove
4. Update
5. Search
6. Generate
-----
6 How many rows do you want to generate?:
100
New rows were generated.
Press any key to continue

```

```
1 -----
| Events: |
|-----|
| Id|      Event name|      Event theme|Event date|
|-----|
| 1|      Party|      Halloween|30.10.2025|
| 2|      Concert|      Musical|02.07.2025|
| 3|      Party|      Birthday|12.09.2025|
| 4|      Concert|      Pop|25.01.2025|
| 5|      Wedding|      Gothic|05.03.2026|
| 6|      Charity evening|      Gallery|07.07.2026|
| 7|      Charity evening|      War|05.01.2024|
| 9|      Concert|      WCY|15.08.2023|
|10|      Party|      GNN|22.11.2023|
|11|      Wedding|      VTI|09.12.2023|
|12|      Party|      GCU|09.08.2023|
|13|      JBG LGQ|      NPJN|04.02.2023|
|14|      Concert|      LNSX|03.07.2023|
|15|      Party|      DOVQ|05.08.2023|
|16|      JBG LGQ|      KLJX|02.04.2023|
|17|      JBG LGQ|      WLBH|26.03.2023|
|18|      Wedding|      SBSB|16.10.2023|
|19|      JBG LGQ|      TFAG|17.08.2023|
|20|      Ball|      AWMG|29.12.2022|
|21|      Charity evening|      FBDY|13.09.2023|
|22|      Ball|      YLUS|26.07.2023|
|23|      XCR TNY|      SMYO|20.05.2023|
|24|      JBG LGQ|      SOJE|29.03.2023|
|25|      VCY PXF|      OBXI|11.01.2023|
|26|      XCR TNY|      KBEN|29.07.2023|
|27|      Concert|      THJD|06.08.2023|
```

```
.....

| 84|      Party|      UIFE|04.04.2023|
| 85|      XCR TNY|      SFUU|11.02.2023|
| 86|      JBG LGQ|      XFNL|25.07.2023|
| 87|      Charity evening|      ROSK|01.12.2023|
| 88|      Ball|      PEEV|09.12.2023|
| 89|      VCY PXF|      QTQG|01.07.2023|
| 90|      Wedding|      ACSN|09.07.2023|
| 91|      JBG LGQ|      TDUR|07.10.2023|
| 92|      JBG LGQ|      WJRU|23.02.2023|
| 93|      XCR TNY|      PFGW|12.11.2023|
| 94|      Ball|      DPAJ|19.08.2023|
| 95|      Ball|      IGHA|15.12.2023|
| 96|      Wedding|      IJGK|28.10.2023|
| 97|      Concert|      PTKJ|05.01.2023|
| 98|      Wedding|      HOOV|07.05.2023|
| 99|      Ball|      RFJV|30.06.2023|
|100|      Wedding|      DMNV|14.06.2023|
|101|      Ball|      EHUV|22.11.2023|
|102|      Concert|      JGOX|23.02.2023|
|103|      JBG LGQ|      HOHT|06.11.2023|
|104|      Charity evening|      JTJU|17.04.2023|
|105|      XCR TNY|      GXVD|02.05.2023|
|106|      JBG LGQ|      VVFJ|27.11.2023|
|107|      VCY PXF|      MGBK|06.07.2023|
|108|      XCR TNY|      JMTE|14.02.2023|
|109|      Ball|      RUHM|18.10.2023|
|110|      JBG LGQ|      IKKR|25.02.2023|
|111|      Party|      FYCN|05.10.2023|
|112|      Party|      YROS|28.04.2023|
```

Приклад генерації 100 стовпців в таблицю Owner.

SQL запит:

```
$"INSERT INTO "Owner" VALUES ('{Id}',
```

```
concat(chr(trunc(65+random()*25)::int), chr(trunc(65+random()*25)::int),
```

```
chr(trunc(65+random()*25)::int), ' ', chr(trunc(65+random()*25)::int),
```

```
chr(trunc(65+random()*25)::int), chr(trunc(65+random()*25)::int)),
```

```
CONCAT(ROUND(random() * 999), '-', TO_CHAR(ROUND(random() *
9999), 'FM0000')) )"

```

Результат генерації:

```
6 How many rows do you want to generate?:
100
New rows were generated.

Press any key to continue
```

Id	Name	Phone
1	Ethan Turner	876-5432
2	Olivia Bennett	234-5678
3	Jackson Mitchell	987-6543
4	Ava Parker	123-4567
5	Morgan Brown	897-1311
6	OGA XBK	971-3075
7	GTI JNN	750-3103
8	Mike Loyson	787-1121
9	UNI PMV	325-5584
10	JFM KBK	780-4736
11	LQR ULL	782-3224
12	XXJ JEQ	965-0858
13	VFU KGP	774-0980
14	QLL FVD	597-2077
15	LAQ EBN	462-3703
16	XMW PQB	109-1009
17	UKT VAS	435-8665
18	KDS LUS	580-3002
19	XSB BIL	468-9580
20	LFB OFS	439-7495
21	SXA QYN	194-0029
22	GDV HNS	298-9342
23	KFQ WGS	984-4298
24	VMK EDW	884-1985
25	WFE PCP	67-3722
26	STY IAW	278-6038
27	NWQ YOM	998-2954
28	TEX FYB	630-5167

.....

80	LMB BJE	609-5974
81	TBU OGV	869-9654
82	HPA UDK	483-3297
83	YSF YTG	656-9131
84	NJU MJH	653-6093
85	VMW JEQ	757-8764
86	LVJ UQC	912-6491
87	EKO JOS	404-4452
88	RXG PAO	924-8344
89	DRV PGB	129-5501
90	CSM BUL	719-7203
91	UNS DPH	943-0451
92	CPW MTP	869-5137
93	KXL HPF	640-3695
94	PUG XNJ	869-3761
95	RVK FUM	760-1715
96	PHX VVB	154-4370
97	KIR SXB	524-8108
98	ONU NWA	931-8031
99	XLI ETI	870-9309
100	AMO FFF	238-4555
101	FMK ESW	380-5672
102	NSJ GNQ	870-6843
103	YRK EQW	851-4430
104	RQE POR	235-2991
105	RKE NQC	415-1037
106	JDH IUX	766-8053
107	RYN BDM	124-0832
108	BTX XBL	594-9135

Завдання №3

Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують (WHERE) та групують (GROUP BY) рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість введення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

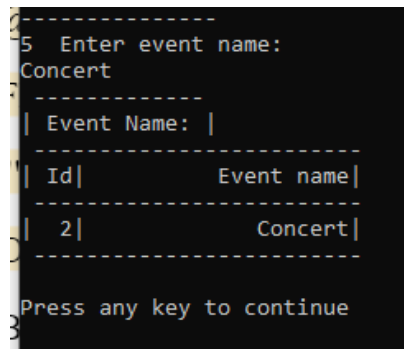
Пошук в таблиці **EventName**:

Для пошуку в таблиці **EventName** було використано SQL-запит:

```
@ "SELECT *  
FROM ""EventName"" WHERE  
""EventName"". ""Name"" LIKE '%||{0}||%'  
ORDER BY ""EventName"". ""ID"" ASC", nm);
```

В цьому випадку замість '%||{0}||%' підставляється введене користувачем значення nm.

Отриманий результат:



```
-----  
5 Enter event name:  
Concert  
-----  
| Event Name: |  
-----  
| Id|          Event name|  
-----  
| 2|          Concert|  
-----  
Press any key to continue
```

Пошук в таблиці **Owner**:

Для пошуку в таблиці **Owner** було використано SQL-запит:

```
( @ "SELECT ""Owner"". ""ID"" as Id,  
""Owner"". ""Name"" as Name,  
""Owner"". ""PhoneNumber"" as Phone  
FROM ""Owner"" WHERE  
""Owner"". ""PhoneNumber"" LIKE '%||{0}||%'  
ORDER BY ""Owner"". ""ID"" ASC", ph);
```


Результат пошуку:

```
2
| Owner |
1. Print
2. Add
3. Remove
4. Update
5. Search
6. Generate
-----
5 Choose the key of search:
1. Owner name
2. Phone number
2 Enter phone number:
12
-----
| Owner: |
-----
-----
| Id | Name | Phone |
-----
| 4 | Ava Parker | 123-4567 |
| 8 | Mike Loyson | 787-1121 |
| 32 | WGR VEG | 120-2413 |
| 46 | IJX EJY | 866-1240 |
| 86 | LVJ UQC | 912-6491 |
| 89 | DRV PGB | 129-5501 |
| 107 | RYN BDM | 124-0832 |
-----
Press any key to continue
```

Пошук в таблиці **Location**:

Для пошуку в таблиці **Location** було використано SQL-запит:

```
( @"SELECT
```

```
    ""Location"". ""ID"" as Id,
```

```
    ""Location"". ""Name"" as Name,
```

```
    ""Location"". ""Address"" as Address,
```

```
    ""Location"". ""NumberOfSeats"" as NumberOfSeats,
```

```
    ""Owner"". ""Name"" as Owner
```

```
FROM ""Location"", ""Owner""
```

```
WHERE ""Location"". ""OwnerId"" = ""Owner"". ""ID""
```

```
AND ""Location"". ""NumberOfSeats"" >= {0}
```

```
AND ""Location"". ""NumberOfSeats"" <= {1}
```

```
ORDER BY ""Location"". ""ID"" ASC", minNos, maxNos);
```

Результати пошуку:

```
-----
5 Choose the key of search:
1. Location name
2. Location address
3. Number of seats
4. Owner name
3 Enter a minimal number of seats:
300
Enter a minimal maximum of seats:
500
-----
| Location: |
-----

-----
| Id | Name | Address | Seats | Owner |
-----
| 3 | Neon Plaza | 456 Luminescent Lane | 300 | Jackson Mitchell |
| 7 | New location | 14 Some Street | 300 | Ava Parker |
-----
```

Як можна побачити, програма вивела лише ті записи, які підпадають під потрібні умови.

Завдання №4

Програмний код організувати згідно шаблону Model-View-Controller(MVC). Приклад організації коду згідно шаблону доступний за [посиланням](#) та його опис [за даним посиланням](#). При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

Дані взяті з методичних вказівок:

Три компоненти шаблону MVC роз'єднані, і вони відповідають за різні речі:

Модель керує даними та визначає правила та поведінку. Він представляє бізнес-логіку програми. Дані можуть зберігатися в самій моделі або в базі даних (лише модель має доступ до бази даних).

View представляє дані користувачеві. Представлення може бути будь-яким видом вихідного представлення: HTML-сторінкою, діаграмою, таблицею або навіть простим текстовим виводом. Представлення ніколи не повинно викликати власні методи; це повинен робити тільки контролер.

Контролер приймає дані користувача та делегує представлення даних представленню, а обробку даних — моделі.

Оскільки модель, представлення та контролер роз'єднані, кожен із трьох можна розширювати, змінювати та замінювати без необхідності переписувати два інших компоненти.

Клас **моделі** та опис його методів:

```

public class ModelClass
{
    private ContextClass context;
    1 reference
    public ModelClass() [...]
    5 references
    public List<TEvent> GetAllEvent() [...] // Отримання всіх івентів
    1 reference
    public int AddEvent(TEvent event_) [...] // Додавання нового івенту
    1 reference
    public int DeleteEvent(int index) [...] // Видалення івенту
    1 reference
    public int UpdateEvent(int id, TEvent event_) [...] // Оновлення івенту
    1 reference
    public List<TEvent> SearchEventByName(string nm) [...] // Пошук івенту за іменем
    1 reference
    public List<TEvent> SearchEventByTheme(string tm) [...] // Пошук івенту за темою
    1 reference
    public List<TEvent> SearchEventByDate(DateOnly dt) [...] // Пошук івенту за датою
    1 reference
    public void GenerateEvents(int n) [...] //Генерація нових івентів
    ///////////////////////////////////////////////////
    3 references
    public List<TOwner> GetAllOwner() [...] // Отримання всіх власників
    1 reference
    public int AddOwner(TOwner owner) [...] // Додавання нового власника
    1 reference
    public int DeleteOwner(int index) [...] // Видалення власника
    1 reference
    public int UpdateOwner(int id, TOwner owner) [...] // Оновлення власника
    1 reference
    public List<TOwner> SearchOwnerByName(string nm) [...] // Пошук власника за іменем
    1 reference
    public List<TOwner> SearchOwnerByPhone(string ph) [...] // Пошук власника за телефоном
    1 reference
    public void GenerateOwner(int n) [...] // Генерація нових власників
    ///////////////////////////////////////////////////
    5 references
    public List<TLocation> GetAllLocation() [...] // Отримання всіх локацій
    1 reference
    public int AddLocation(TLocation location) [...] // Додавання нової локації
    1 reference
    public int DeleteLocation(int index) [...] // Видалення локації
    1 reference
    public int UpdateLocation(int id, TLocation location) [...] // Оновлення локації
    1 reference
    public List<TLocation> SearchLocationByName(string nm) [...] // Пошук локації за назвою
    1 reference
    public List<TLocation> SearchLocationByAddress(string address) [...] // Пошук локації за адресою
    1 reference
    public List<TLocation> SearchLocationByNumberOfSeats(int minNos, int maxNos) [...] // Пошук локації за кількістю місць
    1 reference
    public List<TLocation> SearchLocationByOwner(string owner) [...] // Пошук локації за власником
    1 reference
    public void GenerateLocation(int n) [...] // Генерація нових локацій
    ///////////////////////////////////////////////////
    4 references
    public List<TEventName> GetAllEventName() [...] // Отримання назв івентів
    1 reference
    public int AddEventName(TEventName eventName) [...] // Додавання нової назви івенту
    1 reference
    public int DeleteEventName(int index) [...] // Видалення назви івенту
    1 reference
    public int UpdateEventName(int id, TEventName eventName) [...] // Оновлення назви івенту
    1 reference
    public List<TEventName> SearchEventName(string nm) [...] // Пошук назви івенту
    1 reference
    public void GenerateEventName(int n) [...] // Генерація нових назв івентів
    ///////////////////////////////////////////////////
    2 references
    public List<TEventLocation> GetAllEventLocation() [...] // Отримання всіх пар івентів та локацій
    1 reference
    public int AddEventLocation(int eventId, int locationId) [...] // Додавання нової пари
    1 reference
    public int DeleteEventLocation(int index) [...] // Видалення пари
    1 reference
    public int UpdateEventLocation(int eventId, int locationId, int id) [...] // Оновлення пари
    1 reference
    public List<TEventLocation> SearchEventLocationByEvent(string name) [...] // Пошук пари за івентом
    1 reference
    public List<TEventLocation> SearchEventLocationByLocation(string name) [...] // пошук пари за локацією
    1 reference
    public void GenerateEventLocation(int n) [...] // Генерація нових пар івентів та локацій
}

```