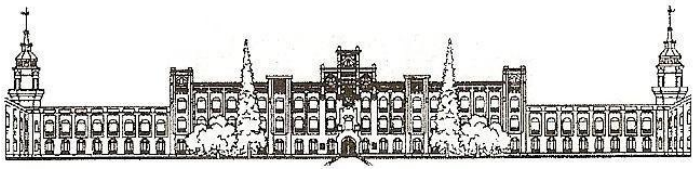


Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

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»

Виконав студент групи:

КВ-11 Шевчук Я.О.

Перевірив: Петрашенко А. В.

Оцінка:

Київ – 2023

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

Загальне завдання роботи полягає у наступному:

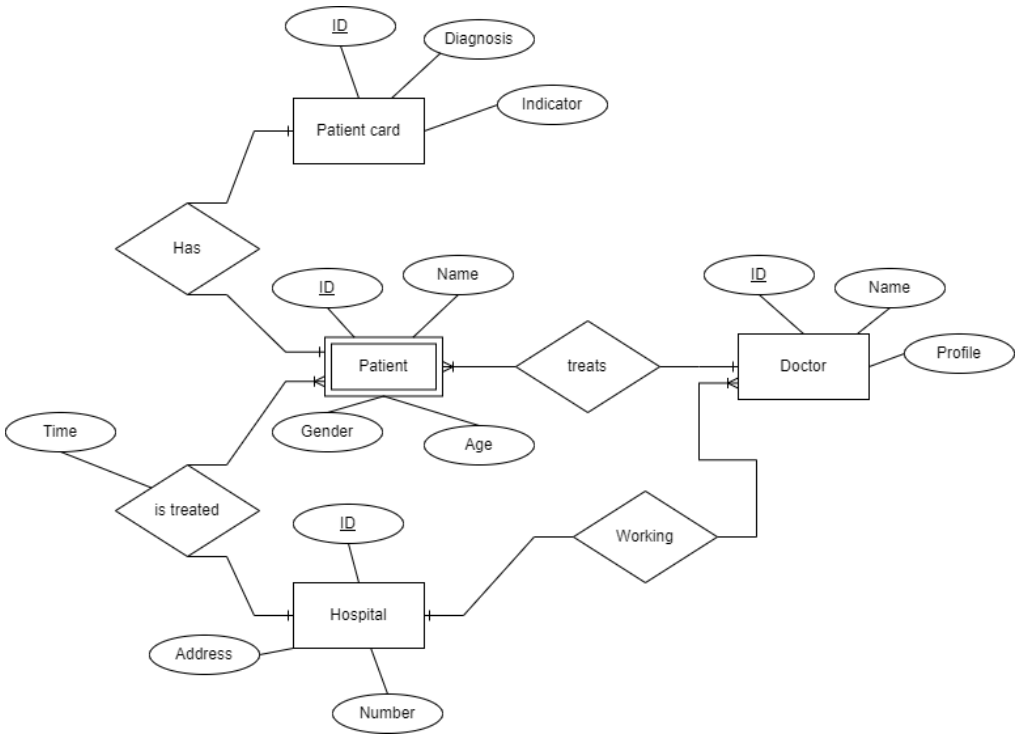
1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами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

Репозиторій на github:
https://github.com/YaroslavaKV-11/database/tree/RGR

Використані інструменти розробки: **pgAdmin4**, мова програмування **C#**, фреймворк **Entity Framework**

Інформація про базу даних

Розробка моделі «сутність-зв’язок» предметної галузі для проектування бази даних «Patient health monitoring system». Предметна галузь - «Система відстеження стану здоров’я пацієнтів».



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

Сутності з описом призначення:

Предметна галузь «Patient health monitoring system» включає в себе 4 сутності, кожна сутність містить декілька атрибутів:

1. Patient (Id, name, gender, age).
2. Patient card (Id, diagnosis, indicator).
3. Hospital (Id, number, address).
4. Doctor (id, name, profile).

Сутність Patient описує пацієнта, який відвідує лікарню. Кожен пацієнт має свій ідентифікатор Id, а також має ім’я, стать та вік.

Сутність Patient card описує стан здоров’я пацієнта. Кожна карта пацієнта має свій ідентифікатор, діагноз пацієнта та показник про його здоров’я(у відсотках).

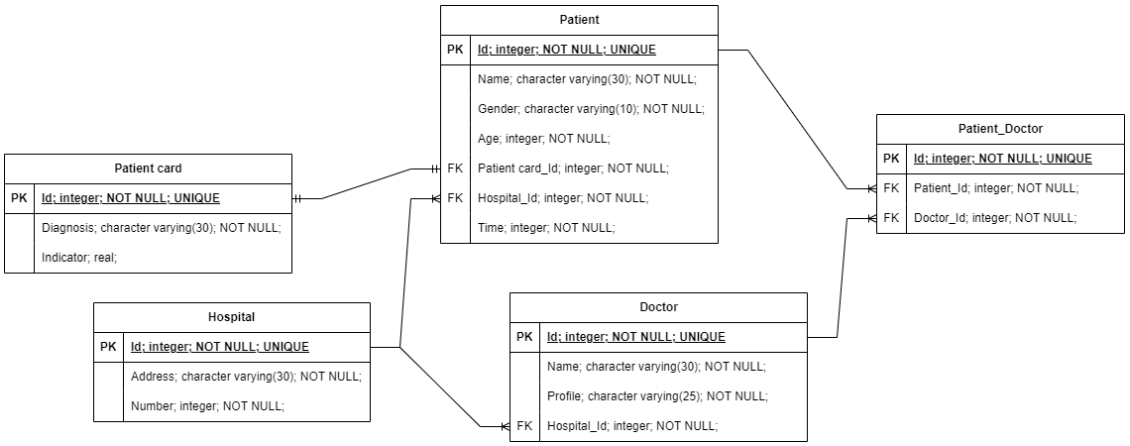
Сутність Hospital описує лікарню, яку відвідує пацієнт. Кожна лікарня має свій ідентифікатор, адресу і кількість пацієнтів, які її відвідують.

Сутність Doctor описує лікаря до якого ходить пацієнт. Кожен лікар має свій ідентифікатор, ім’я та спеціалізацію(профіль).

Зв’язки між сутностями:

Зв’язок між Patient та Patient card:
Кожен пацієнт має свою картку пацієнта, де міститься інформація про його стан здоров’я. Оскільки, пацієнт має тільки одну картку і одна карта відповідає лише одному пацієнту, то зв’язок 1:1.
Зв’язок між Patient та Hospital:
Кожен пацієнт для лікування повинен відвідувати лікарню до якої приписаний. Оскільки, в одній лікарні лікується багато пацієнтів, а один пацієнт відвідує одну лікарню, то зв’язок 1:N.
Зв’язок між Patient та Doctor:
Кожен пацієнт лікується у свого сімейного лікаря. Оскільки в сімейного лікаря своїх пацієнтів багато, але в одного пацієнта лише один сімейний лікар, то зв’язок 1:N.
Зв’язок між Doctor та Hospital:
Кожен лікар працює у лікарні. Оскільки, в лікарні багато лікарів, але один лікар працює лише в одній лікарні, то зв’язок 1:N.

Схема бази даних PostgreSQL



Малюнок 2. Схема бази даних у графічному вигляді

Функціональні залежності:

Patient (*Id*, name, gender, age).

Id → name
Id → gender

Id → age

Patient card (*Id*, diagnosis, indicator).

Id → diagnosis

Id → indicator

Hospital (*Id*, number, address).

Id → number

Id → address

Doctor (*Id*, name, profile).

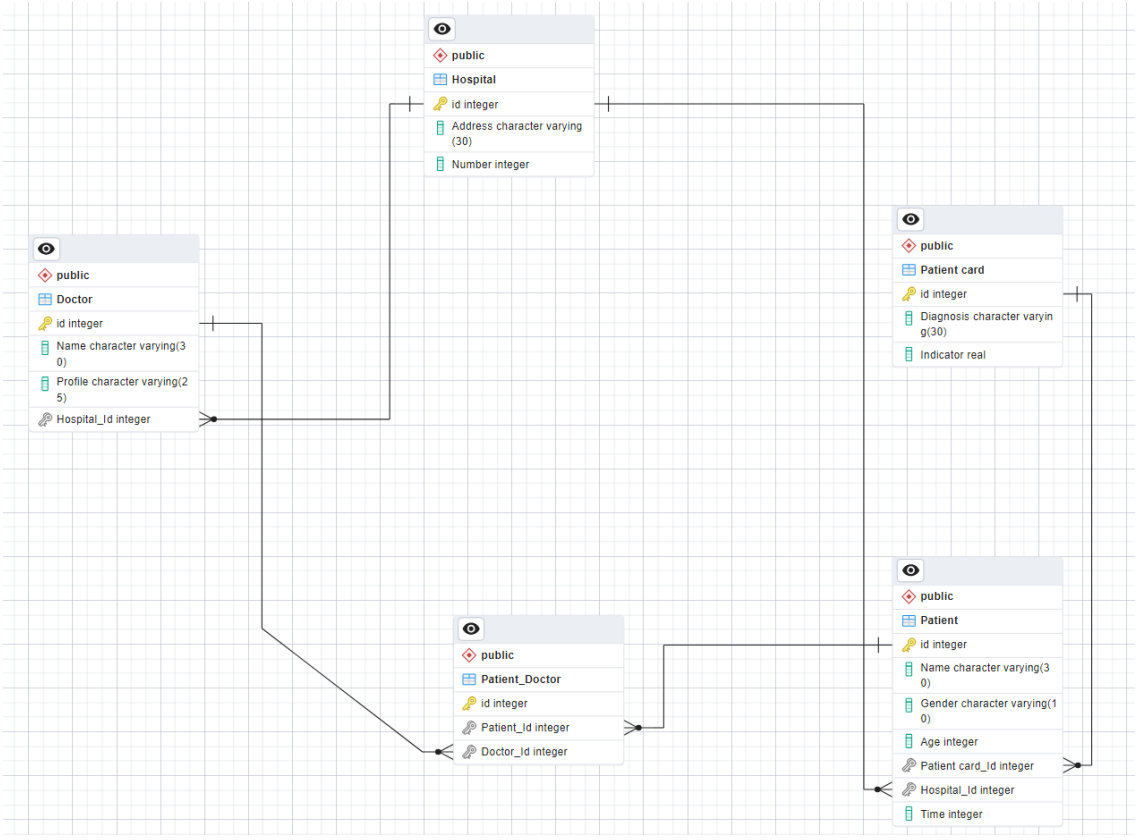
Id → name

Id → profile

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.

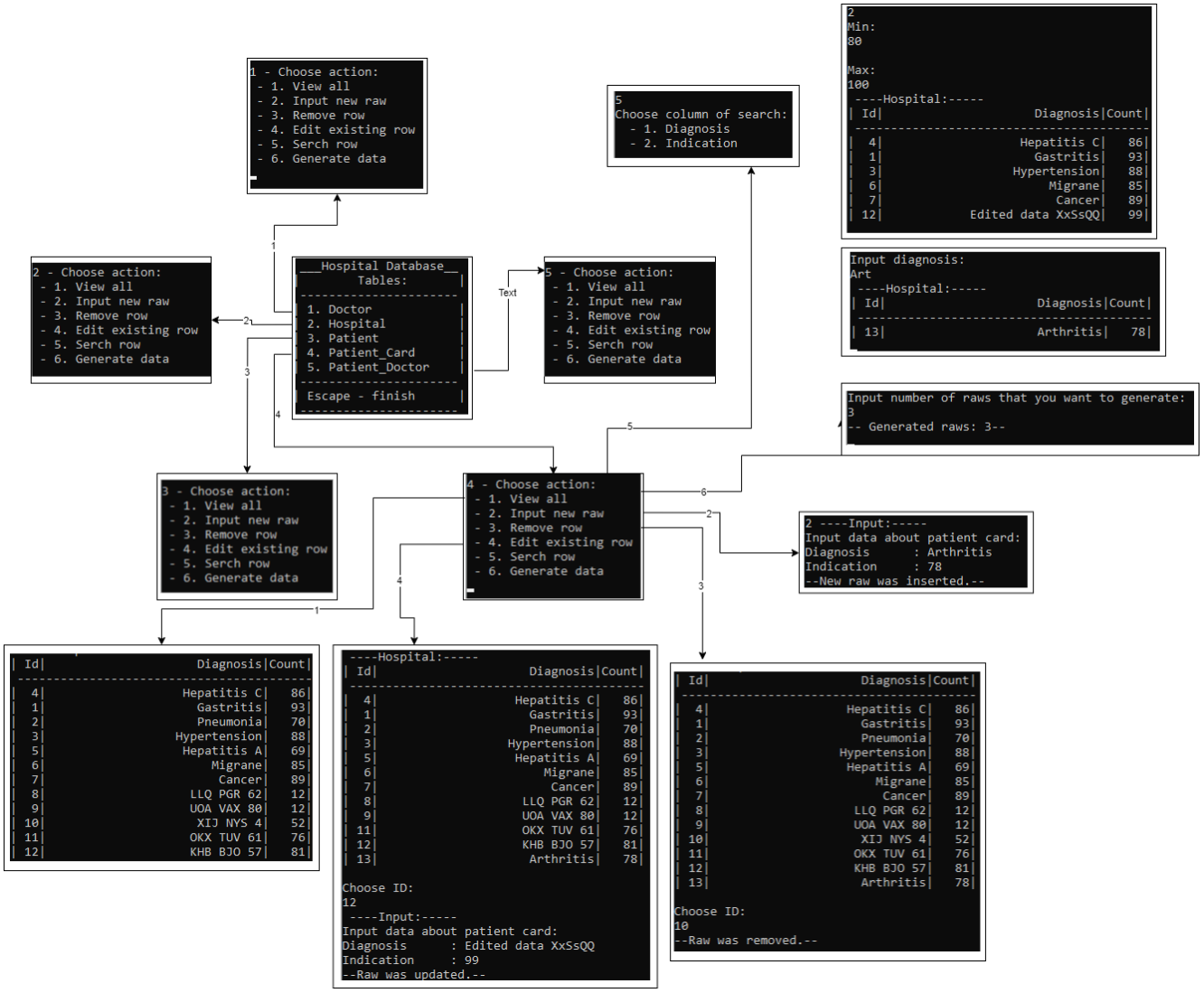
Схема бази даних відповідає 2НФ, бо вона відповідає 1НФ та кожен неключовий атрибут залежить від первинного повного ключа, отже первинний ключ одразу визначає запис та не є надмірним.

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

Схемf меню користувача з описом функціональності кожного пункту



Головне меню - Hospital Database - має 6 пунктів:

1. **Doctor** – перехід до таблиці Doctor
2. **Hospital** – перехід до таблиці Hospital
3. **Patient** – перехід до таблиці Patient
4. **Patient_Card** – перехід до таблиці Patient_Card
5. **Patient_Doctor** – перехід до таблиці Patient_Doctor
6. **Escape** – вихід з програми

Кожний підпункт цього меню(або кожна таблиця) має підменю, яке складається з наступних 6 пунктів:

1. **View all** – перегляд всіх записів таблиці. Робиться SQL запит до відповідної таблички. Результат цього запиту виводиться в консоль.
2. **Input new row** – вставка нового запису в таблицю. Спочатку користувач вводить з клавіатури всі необхідні дані про новий запис, після чого цей запис додається до відповідної таблиці. Перед додаванням відбувається програмна валідація введених даних. Якщо вставка цих даних не є можливою, в консоль буде видане повідомлення про це.
3. **Remove raw** – видалення існуючого запису з таблиці. Для початку

користувачу виведуться всі записи таблиці. Потім він має обрати індекс того запису, який він хоче видалити. Після цього відбувається видалення запису таблиці через команду Delete. Відбувається також перевірка введеного індексу. Якщо такого індексу в таблиці нема, то програма виведе в консоль повідомлення про це.

4. **Edit existing row** – зміна існуючого запису таблиці. Спочатку користувачу виведуться всі записи таблиці. Потім він має обрати індекс того запису, який він хоче модифікувати. Потім відбувається зміна обраного запису. Також відбувається перевірка введеного індексу, а разом з нею і валідація введених нових даних.
5. **Search row** – пошук записів таблиці. Після обрання цього підпункту меню в консоль виводиться ще одне підменю, в якому користувач має обрати колонку, по якій має бути пошук: по стовпцю Name, по стовпцю Age, по стовпцю Address і так далі.
6. **Generate data** – заповнення таблиці рандомно згенерованими даними. Спочатку користувач вводить кількість записів, яку має генерувати програма. Після цього відбувається заповнення таблиці.

Завдання №1

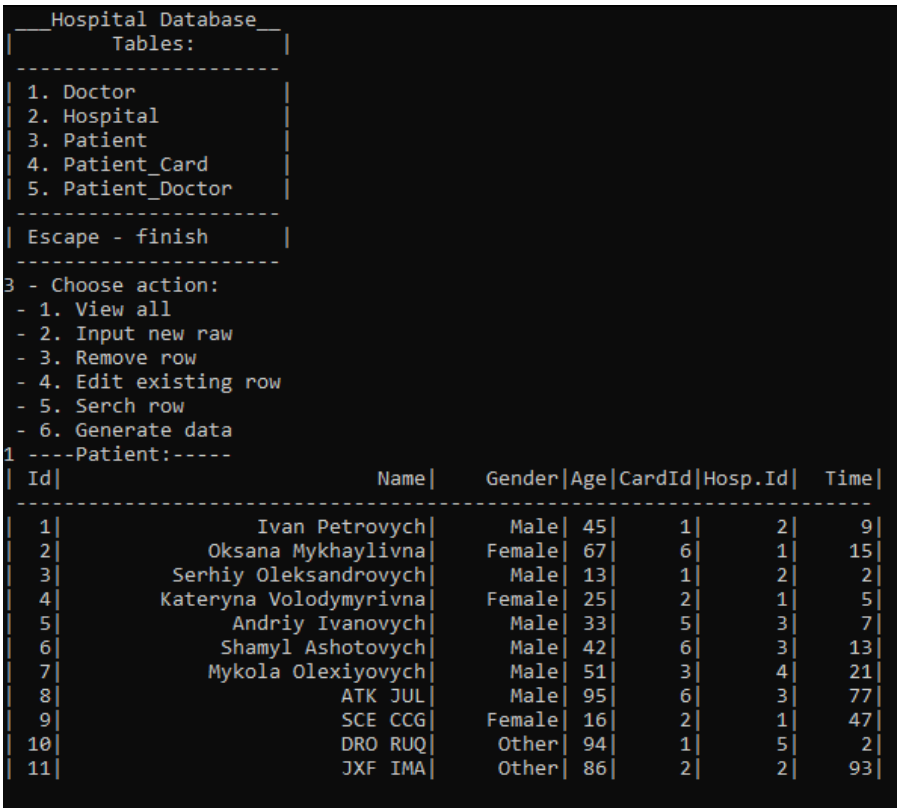
Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера 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 підлеглій 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.

1. Вилучення даних

SQL запит, який був використаний для вилучення даних з таблиці **Patient**:

```
SELECT * FROM "Patient";
```

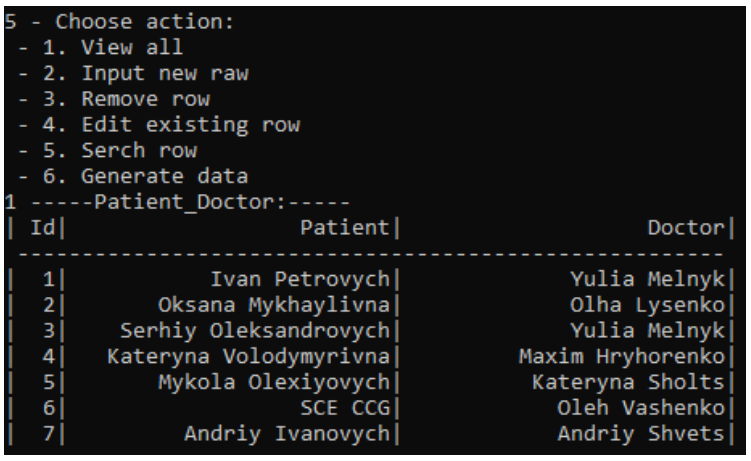
Результат запиту:



Цей запит є найпростішим з усіх, що може бути. Тому слід показати запит для вилучення даних з таблиці **Patient_Doctor**:

```
SELECT "Patient_Doctor"."Id",
        "Patient"."Name" AS Patient,
        "Doctor"."Name" AS Doctor
        FROM "Patient", "Doctor", "Patient_Doctor"
        WHERE "Patient_Doctor"."Patient_Id" = "Patient"."Id"
        AND "Patient_Doctor"."Doctor_Id" = "Doctor"."Id"
```

Результат запиту:



В даному запиті поєднуються дані з 3 таблиць: з таблиці **Patient_Doctor** (береться поле Id), **Patient** (береться поле Name), та **Doctor** (також береться поле Name).

2. Видалення даних

Видалення даних буде розглянуто на прикладі таблиці **Doctor**. Дані до

видалення:

```
1 ----Doctor:-----
| Id|                      Name|                  Profile|№ H.Id|
-----------------------------------------------------------------
|  1|             Yulia Melnyk|       Gastroenterologist|№    2|
|  2|          Maxim Hryhorenko|             Pediatrician|№    1|
|  3|            Serhiy Tkachuk|             Cardiologist|№    3|
|  4|              Olha Lysenko|             Neurosurgeon|№    1|
|  7|             Andriy Shvets|              Dietologist|№    3|
|  6|           Kateryna Sholts|                  Dentist|№    2|
|  9|             Oleh Vashenko|           Dermatologist|№    3|
| 10|              Makar Chuhov|                  Dentist|№    2|
```

Видалення буде відбуватись через команду:

DELETE from "Doctor" WHERE "Id" = {id};

Де {id} – вказане користувачем значення Id, запис которого треба видалити. Таким же чином виконується видалення з усіх інших таблиць. Якщо спробувати видалити запис, який має зв’язок з іншою таблицею, то буде видано повідомлення про помилку. Наприклад, лікар Yulia Melnyk має відношення до записів іншої таблиці(див. скріншот з 1 пункту, де наявні всі дані таблиці **Patient_Doctor**). Отже, видалення цього запису не є можливим:

```
1 - Choose action:
 - 1. View all
 - 2. Input new raw
 - 3. Remove row
 - 4. Edit existing row
 - 5. Serch row
 - 6. Generate data
3 ----Doctor:-----
| Id|                      Name|                  Profile|№ H.Id|
-----------------------------------------------------------------
|  1|             Yulia Melnyk|       Gastroenterologist|№    2|
|  2|          Maxim Hryhorenko|             Pediatrician|№    1|
|  3|            Serhiy Tkachuk|             Cardiologist|№    3|
|  4|              Olha Lysenko|             Neurosurgeon|№    1|
|  7|             Andriy Shvets|              Dietologist|№    3|
|  6|           Kateryna Sholts|                  Dentist|№    2|
|  9|             Oleh Vashenko|           Dermatologist|№    3|
| 10|              Makar Chuhov|                  Dentist|№    2|

Choose ID:
1
--Error. Probably there is no such doctor, or maybe this row is connected with others--
```

Також можна спробувати видалити запис, ввевши Id, якого немає в таблиці. Наприклад 15. Також буде видане повідомлення про помилку:

```
3 ----Doctor:-----
| Id|                      Name|                  Profile|№ H.Id|
-----------------------------------------------------------------
|  1|             Yulia Melnyk|       Gastroenterologist|№    2|
|  2|          Maxim Hryhorenko|             Pediatrician|№    1|
|  3|            Serhiy Tkachuk|             Cardiologist|№    3|
|  4|              Olha Lysenko|             Neurosurgeon|№    1|
|  7|             Andriy Shvets|              Dietologist|№    3|
|  6|           Kateryna Sholts|                  Dentist|№    2|
|  9|             Oleh Vashenko|           Dermatologist|№    3|
| 10|              Makar Chuhov|                  Dentist|№    2|

Choose ID:
15
--Error. Probably there is no such doctor, or maybe this row is connected with others--
```

Такі ж самі запобігання виникнення помилок передбачено в усіх інших таблицях. Алгоритм той же самий.

3. Вставка даних

Приклад вставки даних буде розглянуто на прикладі вставки даних в таблицю **Patient,** оскільки вона має найбільшу кількість полів та 2 зовнішні ключі.

Дані таблиці до вставки:

Id	Name	Gender	Age	CardId	Hosp.Id	Time
1	Ivan Petrovych	Male	45	1	2	9
2	Oksana Mykhaylivna	Female	67	6	1	15
3	Serhiy Oleksandrovych	Male	13	1	2	2
4	Kateryna Volodymyrivna	Female	25	2	1	5
5	Andriy Ivanovych	Male	33	5	3	7
6	Shamyl Ashotovych	Male	42	6	3	13
7	Mykola Olexiyovych	Male	51	3	4	21
8	ATK JUL	Male	95	6	3	77
9	SCE CCG	Female	16	2	1	47
10	DRO RUQ	Other	94	1	5	2
11	JXF IMA	Other	86	2	2	93

Спочатку відбувається програмна валідація, в ході якої перевіряється можливість вставки введених користувачем даних. Після чого відбувається вставка цим запитом:

```
INSERT INTO \"Patient\" VALUES ('{pat.Id}', '{pat.Name}', '{pat.Gender}','{pat.Age}'
,'{pat.Card_Id}',{pat.Hospital_Id}, {pat.Time})
```

Де '{pat.Id}' , {pat.Name} і т.д. – вставка даних, які ввів користувач. Поле Id для цього запису буде згенеровано автоматично, буде збільшено на одиницю найбільший індекс існуючого запису таблиці.

Введення даних користувачем:

```
Input data about patient:
Name        : New Name
Gender      : Female
Age         : 33
Card Id     : 4
Hospital Id : 2
Time        : 20
--New raw was inserted.--
```

Результат запиту:

```
1 ----Patient:-----
| Id|                          Name|    Gender|Age|CardId|Hosp.Id|  Time|
 ---------------------------------------------------------------------------
|  1|                Ivan Petrovych|      Male| 45|     1|      2|     9|
|  2|            Oksana Mykhaylivna|    Female| 67|     6|      1|    15|
|  3|         Serhiy Oleksandrovych|      Male| 13|     1|      2|     2|
|  4|       Kateryna Volodymyrivna|    Female| 25|     2|      1|     5|
|  5|             Andriy Ivanovych|      Male| 33|     5|      3|     7|
|  6|            Shamyl Ashotovych|      Male| 42|     6|      3|    13|
|  7|           Mykola Olexiyovych|      Male| 51|     3|      4|    21|
|  8|                      ATK JUL|      Male| 95|     6|      3|    77|
|  9|                      SCE CCG|    Female| 16|     2|      1|    47|
| 10|                      DRO RUQ|     Other| 94|     1|      5|     2|
| 11|                      JXF IMA|     Other| 86|     2|      2|    93|
| 12|                     New Name|    Female| 33|     4|      2|    20|
```

Якщо дані будуть введені некоректно, то будуть надані відповідні повідомлення. Наприклад, введемо неіснуючий Id лікарні.

Всі записи таблиці Hospital:

```
| Id|                       Address|Count|
 ----------------------------------------
|  1|    15 Shevchenko Street, Kyiv|  500|
|  3|  42 Kharkivska Street, Dnipro|  890|
|  2|       87 Lvivska Street, Lviv| 1500|
|  4|   15 Volodymyra Hryshka, Lviv|  325|
|  5|    45 Kolohovska Street, Kyiv|  333|
```

Введемо Id, якого нема. Наприклад 8:

```
2 ----Patient:-----
Input data about patient:
Name        : xxxx xxx
Gender      : Male
Age         : 43
Card Id     : 2
Hospital Id : 8
Time        : 22
--There is no such hospital Id.--
```

Як можна побачити, повідомлення видано: --There is no such hospital Id.--

Такі ж самі алгоритм валідації передбачені для всіх інших зовнішніх полів цієї таблиці та інших таблиць.

4. Модифікація даних

Модифікація даних буде розглянута на прикладі таблиці **Hospital.** Наявні записи таблиці:

```
1 ----Hospital:-----
| Id|                       Address|Count|
 ----------------------------------------
|  1|    15 Shevchenko Street, Kyiv|  500|
|  3|  42 Kharkivska Street, Dnipro|  890|
|  2|       87 Lvivska Street, Lviv| 1500|
|  4|   15 Volodymyra Hryshka, Lviv|  325|
|  5|    45 Kolohovska Street, Kyiv|  333|
```

Під час модифікації існуючого запису таблиці користувач має обрати Id запису,

який потрібно модифікувати. Потім користувач має ввести нові дані запису. Після цього відбувається команда:

```
UPDATE "Hospital" SET "Address"='{hsp.Address}',
                "Number"='{hsp.Number}' WHERE "Id" = {id_};
```

Знову ж таки, де {hsp.Address} та hsp.Number – дані, введені користувачем.

Введення даних:

```
2 - Choose action:
 - 1. View all
 - 2. Input new raw
 - 3. Remove row
 - 4. Edit existing row
 - 5. Serch row
 - 6. Generate data
4--... --
 ----Hospital:-----
| Id|                        Address|Count|
 ----------------------------------------
|  1|    15 Shevchenko Street, Kyiv|  500|
|  3|  42 Kharkivska Street, Dnipro|  890|
|  2|       87 Lvivska Street, Lviv| 1500|
|  4|   15 Volodymyra Hryshka, Lviv|  325|
|  5|    45 Kolohovska Street, Kyiv|  333|

Choose ID:
4
 ----Input:-----
Input data about hospital:
Address            : 22 Koshova Street, Vinnytsya
Number of patients : 350
--Raw was updated.--
```

Результат запиту:

```
1 ----Hospital:-----
| Id|                        Address|Count|
 ----------------------------------------
|  1|    15 Shevchenko Street, Kyiv|  500|
|  3|  42 Kharkivska Street, Dnipro|  890|
|  2|       87 Lvivska Street, Lviv| 1500|
|  5|    45 Kolohovska Street, Kyiv|  333|
|  4|  22 Koshova Street, Vinnytsya|  350|
```

Під час модифікації також відбувається перевірка введених користувачем даних. Якщо дані не є валідними, буде видано відповідне повідомлення:

```
 ----Hospital:-----
| Id|                        Address|Count|
 ----------------------------------------
|  1|    15 Shevchenko Street, Kyiv|  500|
|  3|  42 Kharkivska Street, Dnipro|  890|
|  2|       87 Lvivska Street, Lviv| 1500|
|  5|    45 Kolohovska Street, Kyiv|  333|
|  4|  22 Koshova Street, Vinnytsya|  350|

Choose ID:
12
 ----Input:-----
Input data about hospital:
Address            : xxx
Number of patients : 1
--There is no such hospital that you want to update.--
```

Перевірка введених даних при модифікації запису таблиці Patient.

SQL запит:

```
UPDATE "Patient" SET "Name"='{pat.Name}',  "Gender"='{pat.Gender}', "Age" = {pat.Age}, "Card_Id" = {pat.Card_Id}, "Hospital_Id" = {pat.Hospital_Id}, "Time" = {pat.Time} WHERE "Id" = {id};
```

```
| Id|                          Name|    Gender|Age|CardId|Hosp.Id|  Time|
 ---------------------------------------------------------------------------
|  1|                Ivan Petrovych|      Male| 45|     1|      2|     9|
|  2|            Oksana Mykhaylivna|    Female| 67|     6|      1|    15|
|  3|          Serhiy Oleksandrovych|      Male| 13|     1|      2|     2|
|  4|        Kateryna Volodymyrivna|    Female| 25|     2|      1|     5|
|  5|              Andriy Ivanovych|      Male| 33|     5|      3|     7|
|  6|             Shamyl Ashotovych|      Male| 42|     6|      3|    13|
|  7|            Mykola Olexiyovych|      Male| 51|     3|      4|    21|
|  8|                       ATK JUL|      Male| 95|     6|      3|    77|
|  9|                       SCE CCG|    Female| 16|     2|      1|    47|
| 10|                       DRO RUQ|     Other| 94|     1|      5|     2|
| 11|                       JXF IMA|     Other| 86|     2|      2|    93|
| 12|                      New Name|    Female| 33|     4|      2|    20|

Choose ID:
5
 ----Patient:-----
Input data about patient:
Name        : Mykola Oleksiyovych
Gender      : Male
Age         : 54
Card Id     : 123
Hospital Id : 3
Time        : 12
--There is no such card--
```

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 **Patient**. SQL зпит:

```
INSERT INTO "Patient" 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) ),

'{gender}', trunc(random()*100)::int, {card_id}, {hosp_id}, trunc(random()*100)::int);
```

Зовнішні ключі довелось згенерувати програмно, щоб рандомно було обрано вже існуючий запис іншої таблиці. ({card_id}, {hosp_id}).

```
3 - Choose action:
 - 1. View all
 - 2. Input new raw
 - 3. Remove row
 - 4. Edit existing row
 - 5. Serch row
 - 6. Generate data
6
Input number of raws that you want to generate:
100
-- Generated raws: 100--
```

Результат генерації 100 записів:

```
3 - Choose action:
 - 1. View all
 - 2. Input new raw
 - 3. Remove row
 - 4. Edit existing row
 - 5. Serch row
 - 6. Generate data
1 ----Patient:-----
| Id|                        Name|    Gender|Age|CardId|Hosp.Id|  Time|
 ------------------------------------------------------------------------
|  1|                Ivan Petrovych|      Male| 45|     1|      2|     9|
|  2|            Oksana Mykhaylivna|    Female| 67|     6|      1|    15|
|  3|          Serhiy Oleksandrovych|      Male| 13|     1|      2|     2|
|  4|         Kateryna Volodymyrivna|    Female| 25|     2|      1|     5|
|  5|              Andriy Ivanovych|      Male| 33|     5|      3|     7|
|  6|            Shamyl Ashotovych|      Male| 42|     6|      3|    13|
| 99|                       GVL PRN|     Other| 34|     1|      1|    61|
|100|                       OWW IBX|     Other| 49|     7|      1|    58|
|101|                       RMU KBT|     Other| 70|    16|      2|    93|
|  7|           Mykola Olexiyovych|      Male| 51|     3|      4|    21|
|  8|                       ATK JUL|      Male| 95|     6|      3|    77|
|  9|                       SCE CCG|    Female| 16|     2|      1|    47|
| 10|                       DRO RUQ|     Other| 94|     1|      5|     2|
| 11|                       JXF IMA|     Other| 86|     2|      2|    93|
| 12|                      New Name|    Female| 33|     4|      2|    20|
| 13|                       ELG HJL|     Other| 65|    15|      5|    30|
| 14|                       QNM ISO|     Other| 83|    12|      1|    34|
| 15|                       BFA CBR|     Other| 77|    16|      4|     3|
| 16|                       GRR CSO|    Female| 11|     2|      2|    63|
| 17|                       EVI SEE|    Female| 72|    16|      2|    96|
| 18|                       QFV AED|      Male| 32|     4|      5|    43|
| 19|                       LLF VDN|    Female| 65|     5|      4|    78|
| 20|                       WJP SCY|    Female| 30|    14|      4|    73|
| 21|                       BEN AVN|    Female| 74|     5|      1|    39|
```

.....

```
| 93|                       YSL YYO|     Other| 60|     7|      3|    94|
| 94|                       TCB DQT|     Other|  3|     6|      2|    17|
| 95|                       HAR WIL|     Other| 75|     6|      2|    33|
| 96|                       CHW PJS|      Male| 80|    13|      3|    90|
| 97|                       DRO FSR|    Female| 50|     5|      3|    26|
| 98|                       SXQ FBM|      Male| 84|     3|      1|    27|
|102|                       QKG FIK|     Other| 14|     4|      2|    59|
|103|                       UAT HLM|     Other| 19|    15|      5|     2|
|104|                       DVY HRL|    Female| 77|     3|      4|    43|
|105|                       DDO DLI|     Other| 35|     2|      4|    91|
|106|                       GAP VTY|      Male| 92|     7|      3|    42|
|107|                       ESX CEU|    Female| 41|     6|      3|    93|
|108|                       XKB ACC|    Female| 58|     4|      2|    31|
|109|                       SVH DNN|     Other| 91|    15|      5|    26|
|110|                       HWY HRA|    Female| 59|     3|      3|    89|
|111|                       PGQ MSQ|     Other| 94|    15|      3|    93|
|112|                       OVG CRE|    Female| 37|     5|      4|    83|
```

Приклад генерування даних для таблиці **Hospital**:

Запит:

```
INSERT INTO "Hospital" 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), ' ',
trunc(random()*100)::int), trunc(random()*2000)::int)
```

Результат генерації 100 записів:

```
6
Input number of raws that you want to generate:
100
-- Generated raws: 100--
```

```
1 ----Hospital:-----
| Id|                      Address|Count|
 ----------------------------------------
|  1|    15 Shevchenko Street, Kyiv|  500|
|  3|  42 Kharkivska Street, Dnipro|  890|
|  2|       87 Lvivska Street, Lviv| 1500|
|  5|    45 Kolohovska Street, Kyiv|  333|
|  4|  22 Koshova Street, Vinnytsya|  350|
|  6|                   IDH PDC 2|  275|
|  7|                  BRT SRD 34|  497|
|  8|                   VYV VHU 5|  830|
|  9|                  VWP RAO 18| 1084|
| 10|                  SCU QNH 33| 1069|
| 11|                  MMX YTG 44|  903|
| 12|                  MVO LNB 11| 1150|
| 13|                  TPW UKA 49| 1011|
| 14|                   MAO DMP 9|  627|
| 15|                  JYH OIX 70|  380|
| 16|                  IGL BIV 57| 1680|
| 17|                  WHG DHX 79| 1423|
| 18|                  GMY IRM 20|   56|
| 19|                  OWU OCR 53|  647|
| 20|                  EWG ESQ 22|  784|
| 21|                  YGR WTO 45|  229|
| 22|                  GLU KGW 96|  423|
| 23|                  OIH XGW 37|  808|
| 24|                  CWW YGC 11|  107|
| 25|                   WOG QDO 5| 1839|
```

.....

```
| 78|                  IYM LJV 78| 1778|
| 79|                   RTR DOY 3| 1282|
| 80|                  GHF PKG 99|  782|
| 81|                  IVY IIE 41| 1379|
| 82|                  IJN AVY 40| 1864|
| 83|                   RMH FJH 0| 1238|
| 84|                  WTT UYI 28| 1292|
| 85|                  HUN RVI 64|  870|
| 86|                  LGO HSC 62| 1133|
| 87|                  LKU AES 38| 1991|
| 88|                  YRJ WFU 33| 1949|
| 89|                  HTS RAT 26| 1356|
| 90|                  KIR MKO 87| 1081|
| 91|                  TQP XXO 69|  396|
| 92|                   INC UAQ 9|  724|
| 93|                  TBB YXH 98|  185|
| 94|                  GIP BME 60| 1981|
| 95|                  LRJ OGA 33|  542|
| 96|                  DMI WGN 32| 1666|
| 97|                  KXI TUI 56|  667|
| 98|                   VBO NNM 7| 1727|
| 99|                  NBH DQV 56|  207|
|100|                   EBR DDW 9|  117|
|101|                  HNP JLM 25| 1393|
|102|                  CEA VUS 53|  469|
|103|                  MEV CWF 97|  900|
|104|                  NNT VRC 53| 1691|
|105|                  VLI RCC 78|  386|
```

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 уведення 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 по таблиці **Patient**:

Пошук за Іменем:

Запит:

```
(  @"SELECT *
    FROM ""Patient""
    WHERE ""Patient"".""Name"" LIKE '%'||{0}||'%'", name);
```

Де name – введене користувачем значення імені.

Результат запиту:

```
Choose column of search:
  - 1. Name
  - 2. Gender
  - 3. Age
  - 4. Card Id
  - 5. Hospital Id
  - 6. Time of being in hospital
1
Input name:
P
 ----Patient:-----
| Id|                          Name|    Gender|Age|CardId|Hosp.Id|  Time|
 ------------------------------------------------------------------------
|  1|                Ivan Petrovych|      Male| 45|     1|      2|     9|
| 99|                       GVL PRN|     Other| 34|     1|      1|    61|
| 20|                       WJP SCY|    Female| 30|    14|      4|    73|
| 25|                       VBT PFL|      Male| 35|    14|      1|    81|
| 26|                       HXK GPD|      Male| 29|    14|      5|    27|
| 32|                       CTB PBL|     Other| 33|     9|      2|    99|
| 37|                       PPT DKA|    Female| 31|    13|      4|    34|
| 41|                       JHP MKD|    Female| 22|     3|      1|    98|
| 42|                       VKS CDP|    Female| 65|    12|      4|    59|
| 54|                       PYT WCO|    Female| 42|     1|      3|    85|
| 58|                       AUY PUG|      Male|  1|     8|      2|    23|
| 61|                       BPO LKH|      Male| 66|    16|      3|     5|
| 65|                       BIL EBP|    Female| 91|    14|      2|    17|
| 73|                       YWC VPB|     Other|  9|     9|      1|    38|
| 78|                       IAP SBP|     Other| 22|     1|      2|    54|
| 96|                       CHW PJS|      Male| 80|    13|      3|    90|
|106|                       GAP VTY|      Male| 92|     7|      3|    42|
|111|                       PGQ MSQ|     Other| 94|    15|      3|    93|
```

Як можна побачити, запит повернув всі записи, в полі Name яких є літера ‘P’.

Пошук за віком. SQL запит:

```
Min:
40

Max:
60
 ----Patient:-----
| Id|                          Name|    Gender|Age|CardId|Hosp.Id|  Time|
 ------------------------------------------------------------------------
|  1|                Ivan Petrovych|      Male| 45|     1|      2|     9|
|  6|             Shamyl Ashotovych|      Male| 42|     6|      3|    13|
|100|                       OWW IBX|     Other| 49|     7|      1|    58|
|  7|             Mykola Olexiyovych|      Male| 51|     3|      4|    21|
| 44|                       BYQ CBM|    Female| 43|     1|      1|    16|
| 46|                       DUF NDK|    Female| 57|    16|      3|    72|
| 48|                       EGM IHB|     Other| 46|    13|      1|     2|
| 49|                       HVK NSW|      Male| 43|     7|      4|    29|
| 51|                       OIM TEF|    Female| 50|     9|      1|    25|
| 54|                       PYT WCO|    Female| 42|     1|      3|    85|
| 56|                       HHG BFM|    Female| 48|    14|      5|    71|
| 59|                       XMC XFT|    Female| 53|    15|      3|    93|
| 66|                       BCA GMO|      Male| 47|    14|      1|    29|
| 68|                       NNW GKV|    Female| 50|    14|      4|    93|
| 69|                       IUM HYL|     Other| 51|    14|      5|    39|
| 71|                       TCR OXL|    Female| 60|    12|      2|    11|
| 72|                       QLJ MQB|      Male| 41|     2|      5|    94|
| 77|                       HMS TMR|      Male| 53|    13|      5|     1|
| 81|                       YHD IEJ|     Other| 46|     3|      2|    63|
| 86|                       FCG TUG|      Male| 59|    11|      5|    10|
| 88|                       TSJ DMK|      Male| 45|     9|      1|    42|
| 89|                       OLG EBJ|      Male| 50|    15|      4|    84|
| 90|                       QOC LTR|    Female| 60|     8|      2|    60|
| 92|                       GUV RVK|      Male| 53|     7|      1|    97|
| 93|                       YSL YYO|     Other| 60|     7|      3|    94|
| 97|                       DRO FSR|    Female| 50|     5|      3|    26|
|107|                       ESX CEU|    Female| 41|     6|      3|    93|
|108|                       XKB ACC|    Female| 58|     4|      2|    31|
|110|                       HWY HRA|    Female| 59|     3|      3|    89|
```

SQL запит:

(“SELECT *

FROM "Patient"

WHERE "Patient"."Age" >= {0}

AND "Patient"."Age"" <= {1}", min, max);

Замість {0} та {1} підставляються введені користувачем значення.

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).

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:

```
class ModelClass
{
    84 references
    HospitalContext context { get; set; }  //Об'єкт класу HospitalContext, DbContext (для доступу до бази даних)
    1 reference
    public ModelClass()...

    /////// Методи для таблиці Doctor
    4 references
    public List<TDoctor> AllDoctors()...// Повертає всі записи таблиці Doctor
    1 reference
    public int InputDoctor(TDoctor doc)...// Метод вставки нового рядку в таблицю Doctor
    1 reference
    public int DeleteDoctor(int id) ...//Метод видалення рядка з таблиці Doctor
    1 reference
    public int UpdateDoctor(TDoctor doc, int id_)...// Метод модифікації даних запису таблиці Doctor
    1 reference
    public List<TDoctor> SearchDoctorName(string nm)...// Метод пошуку за іменем доктора
    1 reference
    public List<TDoctor> SearchDoctorProfile(string pf)...// Метод пошуку за профілем або спеціальністю доктора
    1 reference
    public List<TDoctor> SearchDoctorHospital(int hp)...// Метод пошуку за номером лікарні, в якій працює доктор
    1 reference
    public void GenerateDoctors(int n)... // Метод генерації нових даних в таблицю Doctor
```

```
    /////// Методи для таблиці Hospital
    3 references
    public List<THospital> AllHospitals()...//Повертає всі записи таблиці Hospital
    1 reference
    public int InputHospital(THospital hsp)...// Метод вставки нового рядку в таблицю Hospital
    1 reference
    public int DeleteHospital(int id)...//Метод видалення рядка з таблиці Hospital
    1 reference
    public int UpdateHospital(THospital hsp, int id_)...// Метод модифікації даних запису таблиці Hospital
    1 reference
    public List<THospital> SearchAddress(string ad)...// Метод пошуку за адресою лікарні
    1 reference
    public List<THospital> SearchNumber(int min, int max)...// Метод пошуку за кількістю місць в лікарні
    1 reference
    public void GenerateHospitals(int n)...// Метод генерації нових даних в таблицю Hospital
```

```
    /////// Методи для таблиці Patient
    4 references
    public List<TPatient> AllPatients()...//Повертає всі записи таблиці Patient
    1 reference
    public int InputPatient(TPatient pat)...// Метод вставки нового рядку в таблицю Patient
    1 reference
    public int DeletePatient(int id)...//Метод видалення рядка з таблиці Patient
    1 reference
    public int UpdatePatient(int id, TPatient pat)...// Метод модифікації даних запису таблиці Patient
    1 reference
    public List<TPatient> SearchPatientName (string name)...// Метод пошуку за іменем пацієнта
    1 reference
    public List<TPatient> SearchPatientGender(string gender)...// Метод пошуку за статтю пацієнта
    1 reference
    public List<TPatient> SearchAge(int min, int max) ...// Метод пошуку за віком пацієнта
    1 reference
    public List<TPatient> SearchPatientCard(int id)...// Метод пошуку за номером картки
    1 reference
    public List<TPatient> SearchPatientHospital(int id)...// Метод пошуку за номером лікарні пацієнта
    1 reference
    public List<TPatient> SearchPatientTime(int min, int max)...// Метод пошуку за кількістю годин, проведених у лікарні
    1 reference
    public void GeneratePatients(int n)...// Метод генерації нових даних в таблицю Patient
```

```
    /////// Методи для таблиці PatientCard
    3 references
    public List<TPatientCard> AllPatientCards()...//Повертає всі записи таблиці PatientCard
    1 reference
    public int InputPatientCard(TPatientCard card)...// Метод вставки нового рядку в таблицю PatientCard
    1 reference
    public int DeletePatientCard(int id)...//Метод видалення рядка з таблиці PatientCard
    1 reference
    public int UpdatePatientCard(TPatientCard hsp, int id_)...// Метод модифікації даних запису таблиці PatientCard
    1 reference
    public List<TPatientCard> SearchDiagnosis(string ad)...// Метод пошуку за діагнозом
    1 reference
    public List<TPatientCard> SearchIndication(int min, int max)...// Метод пошуку за індікатором
    1 reference
    public void GeneratePatientCard(int n)...// Метод генерації нових даних в таблицю PatientCard
```