

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

Розрахунково-графічна робота
з дисципліни «Бази Даних»

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

Виконав студент групи: КВ-32

Косарук Захар

Варіант: Медична система для збереження даних пацієнтів

Репозиторій на GitHub: <https://github.com/zZaKko96/Medicine-Database.git>

Telegram: <https://t.me/zZaKko>

Київ 2025

Медична система для збереження даних пацієнтів

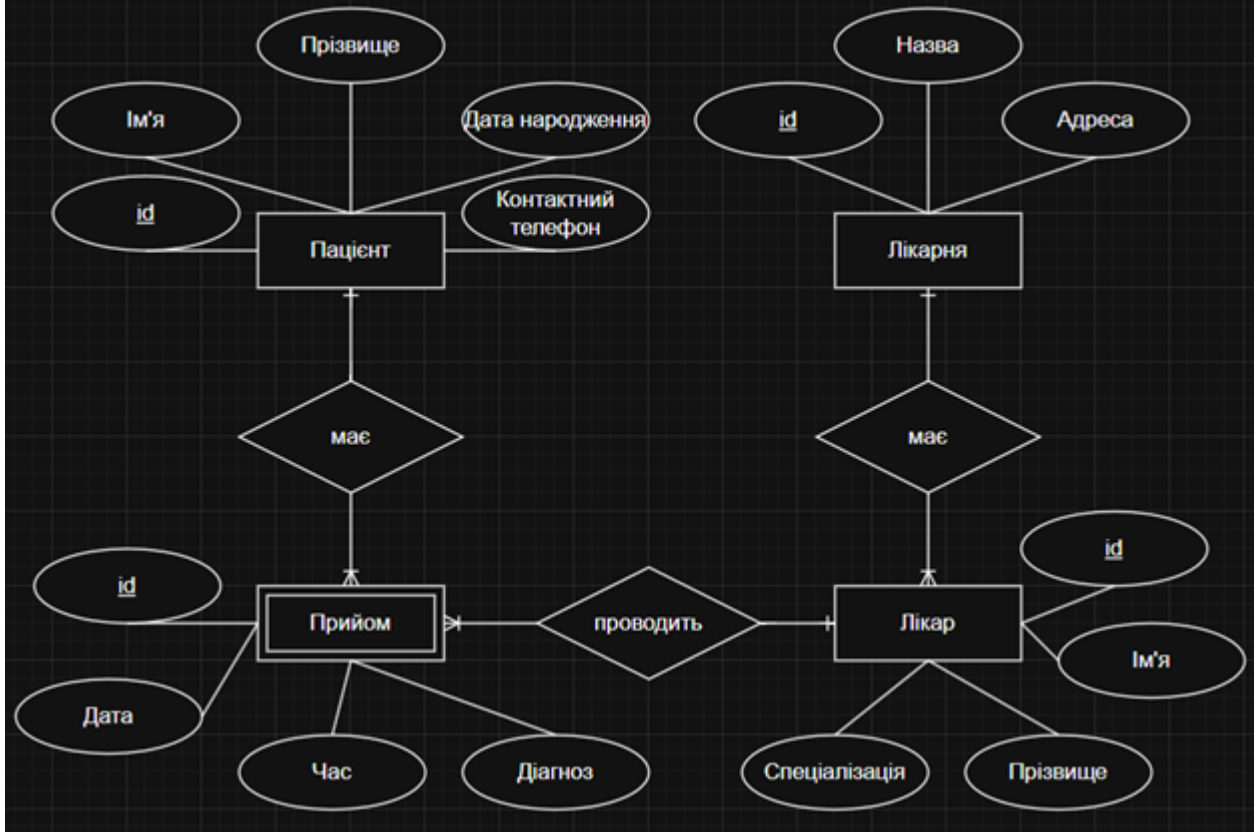


Рисунок 1. Діаграма сутність-зв'язок.

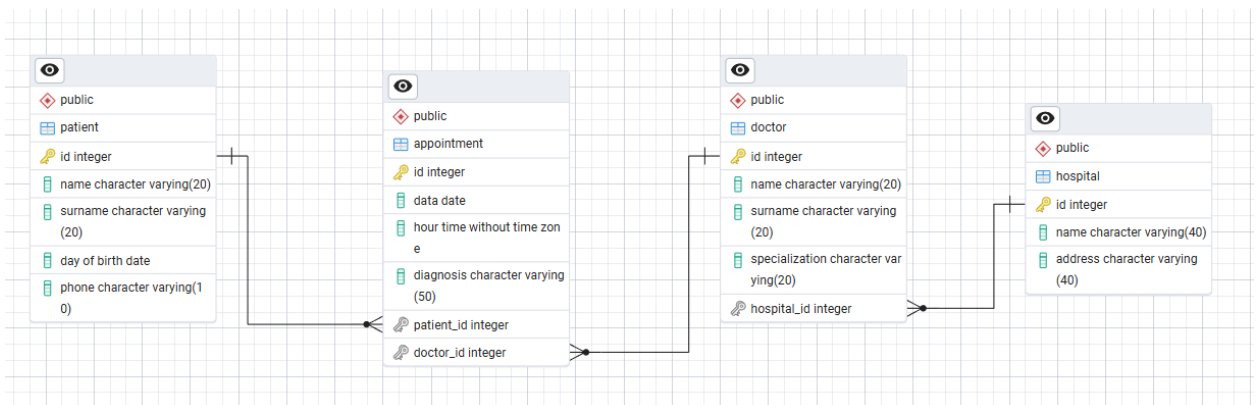


Рисунок 2. Структура бази даних.

Опис бази даних

Пацієнт – сутність для запису даних пацієнтів (ім'я, прізвище, дата народження, контактний телефон)

Лікар – сутність для запису даних лікарів (ім'я, прізвище, спеціалізація, лікарня)

Прийом – сутність для запису існуючих прийомів між пацієнтом і лікарем (дата, час, діагноз, id пацієнта (який записаний на прийом), id лікаря (який веде прийом))

Лікарня – сутність для запису даних про лікарні (назва, адреса)

Схема меню користувача

ГОЛОВНЕ МЕНЮ

```
|
+-- 1. Додати
| |
| +-- 1. Додати пацієнта
| +-- 2. Додати лікаря
| +-- ...
| \-- 0. Вихід
|
+-- 2. Видалити
| |
| +-- 1. Видалити пацієнта
| \-- 0. Вихід
|
+-- 3. Переглянути
| |
| +-- 1. Переглянути таблицю Пацієнт
| \-- 0. Вихід
|
+-- 4. Редагувати
| |
| +-- 1. Редагувати таблицю Пацієнт
| \-- 0. Вихід
|
+-- 5. Запустити генерацію
+-- 6. Очистити базу даних
+-- 7. Пошук 1
+-- 8. Пошук 2
+-- 9. Пошук 3
\-- 0. Вихід
```

Головне меню

- **1. Додати**
 - Переводить користувача у підменю для створення нових записів у базі даних (пацієнтів, лікарів, лікарень, прийомів).
- **2. Видалити**
 - Переводить у підменю для видалення існуючих записів. Система включає перевірку обмежень цілісності (наприклад, неможливо видалити лікарню, якщо за нею закріплені лікарі).

- **3. Переглянути**
 - Переводить у підменю для перегляду вмісту таблиць бази даних.
- **4. Редагувати**
 - Переводить у підменю для оновлення даних в існуючих записах.
- **5. Запустити генерацію 'рандомних' даних**
 - Викликає `GenerateDataAsync`. Запитує у користувача кількість записів та виконує один складний SQL-запит для заповнення всіх таблиць реалістичними, випадковими даними, коректно обробляючи зовнішні ключі.
- **6. Очистити усю базу даних**
 - Викликає `ClearDataAsync`. Після підтвердження користувачем (ТАК), виконує `TRUNCATE ... RESTART IDENTITY CASCADE` для повного очищення всіх таблиць та скидання лічильників `id` до 1.
- **7. Пошук пацієнтів за прізвищем лікаря та датою**
 - Викликає `SearchPatientsAsync`. Запитує прізвище лікаря та діапазон дат. Виконує SQL-запит з `JOIN`, `WHERE` та `GROUP BY`, показуючи, скільки прийомів мав кожен пацієнт у цього лікаря. Виводить час виконання запиту в мс.
- **8. Вивести статистику по лікарях**
 - Викликає `SearchDoctorStatisticsAsync`. Запитує спеціалізацію та діапазон дат. Виконує SQL-запит з `LEFT JOIN`, `WHERE`, `GROUP BY` та `COUNT`, показуючи, скільки прийомів провів кожен лікар. Виводить час виконання запиту в мс.
- **9. Вивести статистику по лікарнях**
 - Викликає `SearchHospitalStatisticsAsync`. Запитує адресу та діагноз. Виконує SQL-запит з `JOIN`, `WHERE`, `GROUP BY` та `COUNT(DISTINCT)`, показуючи, скільки унікальних пацієнтів з цим діагнозом прийняла кожна лікарня. Виводить час виконання запиту в мс.
- **0. Вихід**
 - **Опис:** Завершує головний цикл у `Controller.cs` та коректно закриває програму.

Підменю

Меню "Додати" (Пункт 1)

- **1. Додати пацієнта:** Створює новий запис у таблиці patient.
- **2. Додати лікаря:** Створює новий запис у doctor. Перевіряє існування hospital_id.
- **3. Додати прийом:** Створює новий запис у appointment. Перевіряє існування patient_id та doctor_id.
- **4. Додати лікарню:** Створює новий запис у таблиці hospital.
- **0. Вихід:** Повертає до головного меню.

Меню "Видалити" (Пункт 2)

- **1. Видалити пацієнта:** Видаляє пацієнта за id. Блокує видалення, якщо у пацієнта є прийоми.
- **2. Видалити лікаря:** Видаляє лікаря за id. Блокує видалення, якщо у лікаря є прийоми.
- **3. Видалити прийом:** Видаляє прийом за id.
- **4. Видалити лікарню:** Видаляє лікарню за id. Блокує видалення, якщо за лікарнею закріплені лікарі.
- **0. Вихід:** Повертає до головного меню.

Меню "Переглянути" (Пункт 3)

- **1. Переглянути таблицю Пацієнт:** Виконує SELECT * FROM patient та виводить список.
- **2. Переглянути таблицю Лікар:** Виконує SELECT * FROM doctor та виводить список.
- **3. Переглянути таблицю Прийом:** Виконує SELECT * FROM appointment та виводить список.
- **4. Переглянути таблицю Лікарня:** Виконує SELECT * FROM hospital та виводить список.
- **0. Вихід:** Повертає до головного меню.

Меню "Редагувати" (Пункт 4)

- **1. Редагувати таблицю Пацієнт:** Запитує id, завантажує поточні дані, дозволяє користувачу ввести нові значення та виконує UPDATE.
- **2. Редагувати таблицю Лікар:** Аналогічно до пацієнта, оновлює дані лікаря.

- **3. Редагувати таблицю Прийом:** Аналогічно, оновлює дані прийому.
- **4. Редагувати таблицю Лікарня:** Аналогічно, оновлює дані лікарні.
- **0. Вихід:** Повертає до головного меню.

Мова програмування та використані бібліотеки

- Мова програмування: C#
- Платформа: .NET 8
- Середовище розробки: Visual Studio 2022
- Система управління базами даних (СУБД): PostgreSQL
- Бібліотека для взаємодії з БД: Npgsql

Деталізоване завдання №1

А. Невдале видалення:

```

--- МЕДИЧНА СИСТЕМА ---

1. Додати
2. Видалити
3. Переглянути
4. Редагувати

5. Запустити генерацію 'рандомних' даних
6. Очистити усю базу даних

7. Пошук пацієнтів за прізвищем лікаря та датою
8. Вивести статистику по лікарях
9. Вивести статистику по лікарнях

0. Вихід
Ваш вибір: 3

1. Переглянути таблицю Пацієнт
2. Переглянути таблицю Лікар
3. Переглянути таблицю Прийом
4. Переглянути таблицю Лікарня

0. Вихід
Ваш вибір: 2

--- Список Лікарів ---
-----
ID: 1, Ім'я: Олег Петров, Спец: Педіатр, ID Лікарні: 2
ID: 2, Ім'я: Тарас Захаров, Спец: Кардіолог, ID Лікарні: 3
ID: 3, Ім'я: Володимир Попова, Спец: Хірург, ID Лікарні: 8
ID: 4, Ім'я: Володимир Петров, Спец: Хірург, ID Лікарні: 4
ID: 5, Ім'я: Володимир Гончарук, Спец: Терапевт, ID Лікарні: 3
ID: 6, Ім'я: Володимир Павлюк, Спец: Невролог, ID Лікарні: 10
ID: 7, Ім'я: Наталія Давидова, Спец: Офтальмолог, ID Лікарні: 3
ID: 8, Ім'я: Володимир Павлюк, Спец: Невролог, ID Лікарні: 8
ID: 9, Ім'я: Олег Лисенко, Спец: Офтальмолог, ID Лікарні: 10
ID: 10, Ім'я: Максим Петров, Спец: Хірург, ID Лікарні: 2
ID: 11, Ім'я: Олег Ковальчук, Спец: Педіатр, ID Лікарні: 4
ID: 12, Ім'я: Ірина Попова, Спец: Невролог, ID Лікарні: 1
ID: 13, Ім'я: Ігор Мартинюк, Спец: Педіатр, ID Лікарні: 3
ID: 14, Ім'я: Назар Мартинюк, Спец: Невролог, ID Лікарні: 8
ID: 15, Ім'я: Ольга Попова, Спец: Педіатр, ID Лікарні: 8
ID: 16, Ім'я: Наталія Павлюк, Спец: Терапевт, ID Лікарні: 8
ID: 17, Ім'я: Максим Гончарук, Спец: Хірург, ID Лікарні: 5
ID: 18, Ім'я: Ольга Лисенко, Спец: Кардіолог, ID Лікарні: 9
ID: 19, Ім'я: Тарас Іваненко, Спец: Хірург, ID Лікарні: 5
ID: 20, Ім'я: Світлана Захаров, Спец: Офтальмолог, ID Лікарні: 4
ID: 21, Ім'я: Тарас Павлюк, Спец: Проктолог, ID Лікарні: 1

Натисніть Enter для продовження...

```

Вміст дочірньої таблиці doctor до спроби видалення.

```
--- МЕДИЧНА СИСТЕМА ---

1. Додати
2. Видалити
3. Переглянути
4. Редагувати

5. Запустити генерацію 'рандомних' даних
6. Очистити усю базу даних

7. Пошук пацієнтів за прізвищем лікаря та датою
8. Вивести статистику по лікарях
9. Вивести статистику по лікарнях

0. Вихід
Ваш вибір: 2

1. Видалити пацієнта
2. Видалити лікаря
3. Видалити прийом
4. Видалити лікарню

0. Вихід
Ваш вибір: 4
--- Видалення лікарні ---
Натисніть Enter для продовження...

Введіть ID лікарні для видалення: 2
ПОМИЛКА: Неможливо видалити лікарню, оскільки за нею закріплені лікарі.
Натисніть Enter для продовження...
|
```

Результат перехоплення помилки при спробі видалення батьківського запису (Лікарня ID=2)

Б. Успішне видалення

```
--- Список Лікарів ---
ID: 1, Ім'я: Олег Петров, Спец: Педіатр, ID Лікарні: 2
ID: 2, Ім'я: Тарас Захаров, Спец: Кардіолог, ID Лікарні: 3
ID: 3, Ім'я: Володимир Попова, Спец: Хірург, ID Лікарні: 8
ID: 4, Ім'я: Володимир Петров, Спец: Хірург, ID Лікарні: 4
ID: 5, Ім'я: Володимир Гончарук, Спец: Терапевт, ID Лікарні: 3
ID: 6, Ім'я: Володимир Павлюк, Спец: Невролог, ID Лікарні: 10
ID: 7, Ім'я: Наталія Давидова, Спец: Офтальмолог, ID Лікарні: 3
ID: 8, Ім'я: Володимир Павлюк, Спец: Невролог, ID Лікарні: 8
ID: 9, Ім'я: Олег Лисенко, Спец: Офтальмолог, ID Лікарні: 10
ID: 10, Ім'я: Максим Петров, Спец: Хірург, ID Лікарні: 2
ID: 11, Ім'я: Олег Ковальчук, Спец: Педіатр, ID Лікарні: 4
ID: 12, Ім'я: Ірина Попова, Спец: Невролог, ID Лікарні: 1
ID: 13, Ім'я: Ігор Мартинюк, Спец: Педіатр, ID Лікарні: 3
ID: 14, Ім'я: Назар Мартинюк, Спец: Невролог, ID Лікарні: 8
ID: 15, Ім'я: Ольга Попова, Спец: Педіатр, ID Лікарні: 8
ID: 16, Ім'я: Наталія Павлюк, Спец: Терапевт, ID Лікарні: 8
ID: 17, Ім'я: Максим Гончарук, Спец: Хірург, ID Лікарні: 5
ID: 18, Ім'я: Ольга Лисенко, Спец: Кардіолог, ID Лікарні: 9
ID: 19, Ім'я: Тарас Іваненко, Спец: Хірург, ID Лікарні: 5
ID: 20, Ім'я: Світлана Захаров, Спец: Офтальмолог, ID Лікарні: 4
ID: 21, Ім'я: Тарас Павлюк, Спец: Проктолог, ID Лікарні: 1

Натисніть Enter для продовження...

--- Список Лікарень ---
ID: 1, Назва: Дніпровська міська лікарня, Адреса: вулиця Франка, 125
ID: 2, Назва: Дніпровська міська лікарня, Адреса: проспект Соборності, 78
ID: 3, Назва: Харківська районна лікарня, Адреса: вулиця Перемоги, 10
ID: 4, Назва: Дніпровська міська лікарня, Адреса: вулиця Перемоги, 166
ID: 5, Назва: Львівська обласна лікарня, Адреса: вулиця Миру, 152
ID: 6, Назва: Одеська обласна лікарня, Адреса: проспект Перемоги, 76
ID: 7, Назва: Харківська районна лікарня, Адреса: проспект Шевченка, 127
ID: 8, Назва: Дніпровська міська лікарня, Адреса: вулиця Миру, 107
ID: 9, Назва: Волинська обласна лікарня, Адреса: бульвар Перемоги, 49
ID: 10, Назва: Луцька обласна лікарня, Адреса: проспект Грушевського, 122
ID: 11, Назва: Київська обласна лікарня, Адреса: бульвар Грушевського, 72
ID: 12, Назва: Лікарня на видалення, Адреса: -

Натисніть Enter для продовження...
|
```

Вміст таблиці hospital до успішного видалення

```

1. Видалити пацієнта
2. Видалити лікаря
3. Видалити прийом
4. Видалити лікарню

0. Вихід
Ваш вибір: 4
--- Видалення лікарні ---
Натисніть Enter для продовження...

Введіть ID лікарні для видалення: 12
Лікарню успішно видалено.
Натисніть Enter для продовження...
|

```

```

--- Список Лікарень ---
-----
ID: 1, Назва: Дніпровська міська лікарня, Адреса: вулиця Франка, 125
ID: 2, Назва: Дніпровська міська лікарня, Адреса: проспект Соборності, 78
ID: 3, Назва: Харківська районна лікарня, Адреса: вулиця Перемоги, 10
ID: 4, Назва: Дніпровська міська лікарня, Адреса: вулиця Перемоги, 166
ID: 5, Назва: Львівська обласна лікарня, Адреса: вулиця Миру, 152
ID: 6, Назва: Одеська обласна лікарня, Адреса: проспект Перемоги, 76
ID: 7, Назва: Харківська районна лікарня, Адреса: проспект Шевченка, 127
ID: 8, Назва: Дніпровська міська лікарня, Адреса: вулиця Миру, 107
ID: 9, Назва: Волинська обласна лікарня, Адреса: бульвар Перемоги, 49
ID: 10, Назва: Луцька обласна лікарня, Адреса: проспект Грушевського, 122
ID: 11, Назва: Київська обласна лікарня, Адреса: бульвар Грушевського, 72

Натисніть Enter для продовження...
|

```

```

--- Список Лікарів ---
-----
ID: 1, Ім'я: Олег Петров, Спец: Педіатр, ID Лікарні: 2
ID: 2, Ім'я: Тарас Захаров, Спец: Кардіолог, ID Лікарні: 3
ID: 3, Ім'я: Володимир Попова, Спец: Хірург, ID Лікарні: 8
ID: 4, Ім'я: Володимир Петров, Спец: Хірург, ID Лікарні: 4
ID: 5, Ім'я: Володимир Гончарук, Спец: Терапевт, ID Лікарні: 3
ID: 6, Ім'я: Володимир Павлюк, Спец: Невролог, ID Лікарні: 10
ID: 7, Ім'я: Наталія Давидова, Спец: Офтальмолог, ID Лікарні: 3
ID: 8, Ім'я: Володимир Павлюк, Спец: Невролог, ID Лікарні: 8
ID: 9, Ім'я: Олег Лисенко, Спец: Офтальмолог, ID Лікарні: 10
ID: 10, Ім'я: Максим Петров, Спец: Хірург, ID Лікарні: 2
ID: 11, Ім'я: Олег Ковальчук, Спец: Педіатр, ID Лікарні: 4
ID: 12, Ім'я: Ірина Попова, Спец: Невролог, ID Лікарні: 1
ID: 13, Ім'я: Ігор Мартинюк, Спец: Педіатр, ID Лікарні: 3
ID: 14, Ім'я: Назар Мартинюк, Спец: Невролог, ID Лікарні: 8
ID: 15, Ім'я: Ольга Попова, Спец: Педіатр, ID Лікарні: 8
ID: 16, Ім'я: Наталія Павлюк, Спец: Терапевт, ID Лікарні: 8
ID: 17, Ім'я: Максим Гончарук, Спец: Хірург, ID Лікарні: 5
ID: 18, Ім'я: Ольга Лисенко, Спец: Кардіолог, ID Лікарні: 9
ID: 19, Ім'я: Тарас Іваненко, Спец: Хірург, ID Лікарні: 5
ID: 20, Ім'я: Світлана Захаров, Спец: Офтальмолог, ID Лікарні: 4
ID: 21, Ім'я: Тарас Павлюк, Спец: Проктолог, ID Лікарні: 1

Натисніть Enter для продовження...
|

```

Вміст дочірньої таблиці doctor після успішного видалення hospital (ID=12). Таблиця не змінилася, оскільки до видаленої лікарні не були прив'язані лікарі

```

public async Task<string> DeleteHospitalAsync(int hospitalId)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "DELETE FROM hospital WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("id", hospitalId);
        await cmd.ExecuteNonQueryAsync();
        return "Лікарню успішно видалено.";
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо видалити лікарню, оскільки за нею закріплені лікарі.";
    }
    catch (Exception ex)
    {
    }
}

```



```

    {
        return $"Загальна помилка: {ex.Message}";
    }
}

```

Причина помилки полягає у спрацьовуванні обмеження зовнішнього ключа (foreign key constraint) на рівні СУБД PostgreSQL. У таблиці doctor стовпець hospital_id є зовнішнім ключем, який посилається на id в таблиці hospital. База даних не дозволяє видалити "батьківський" запис (hospital.id = 2), доки існують "дочірні" записи (doctor), які на нього посилаються.

Програмно ця помилка була коректно оброблена. Замість аварійного завершення, програма перехопила виняток Npgsql.PostgresException, ідентифікувала його за кодом ex.SqlState == "23503" (foreign_key_violation) та вивела користувачу відповідне повідомлення про неможливість видалення.

Результат перехоплення помилки при спробі вставки запису в дочірню таблицю (appointment) з неіснуючим зовнішнім ключем (patient_id=9999).

```

public async Task<string> AddAppointmentAsync(Appointment app)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "INSERT INTO appointment (data, hour, diagnosis, patient_id,
doctor_id) " +
                "VALUES (@data, @hour, @diag, @p_id, @d_id)";
        await using var cmd = new NpgsqlCommand(sql, conn);

        cmd.Parameters.AddWithValue("data", app.Data);
        cmd.Parameters.AddWithValue("hour", app.Hour);
        cmd.Parameters.AddWithValue("diag", app.Diagnosis);
        cmd.Parameters.AddWithValue("p_id", app.PatientId);
        cmd.Parameters.AddWithValue("d_id", app.DoctorId);

        await cmd.ExecuteNonQuery();
    }
}

```

```

        return "Прийом успішно додано.";
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо додати прийом. " +
            "Переконайтеся, що пацієнт та лікар з вказаними ID існують.";
    }
}

```

Причина помилки полягає у спрацьовуванні обмеження зовнішнього ключа (foreign key constraint). Користувач намагався додати запис у таблицю appointment зі значенням patient_id=9999.

СУБД PostgreSQL перевірила, чи існує запис з id=99999 у батьківській таблиці patient. Оскільки такого запису не існує, база даних згенерувала помилку порушення цілісності (SQLSTATE 23503), заборонивши операцію INSERT.

Як видно з лістингу, ця помилка була коректно перехоплена в коді C# за допомогою блоку try-catch, і програма вивела користувачу зрозуміле повідомлення замість аварійного завершення.

Деталізоване завдання №2

```

--- МЕДИЧНА СИСТЕМА ---
1. Додати
2. Видалити
3. Переглянути
4. Редагувати

5. Запустити генерацію 'рандомних' даних
6. Очистити усю базу даних

7. Пошук пацієнтів за прізвищем лікаря та датою
8. Вивести статистику по лікарях
9. Вивести статистику по лікарнях

0. Вихід
Ваш вибір: 5
--- Генерація даних ---
Натисніть Enter для продовження...

Скільки пацієнтів/приймів згенерувати?: 10000
Запускаю генерацію... Це може зайняти час.
Натисніть Enter для продовження...

Успішно згенеровано дані. Загальний час: 12430 мс.
Натисніть Enter для продовження...

```

```

D:\Projects\RG\Rdb\bin\Debug  X  +  v
ID: 9974, Ім'я: Юлія, Прізвище: Кравченко, ДН: 19.05.2003, Тел: 0953109867
ID: 9975, Ім'я: Анна, Прізвище: Іванов, ДН: 10.04.2018, Тел: 0958223663
ID: 9976, Ім'я: Сергій, Прізвище: Коваленко, ДН: 12.04.1964, Тел: 0502740093
ID: 9977, Ім'я: Марія, Прізвище: Косарук, ДН: 23.05.2013, Тел: 0509252776
ID: 9978, Ім'я: Анна, Прізвище: Іванов, ДН: 04.05.2019, Тел: 0677076787
ID: 9979, Ім'я: Дмитро, Прізвище: Іванов, ДН: 07.03.2016, Тел: 0958272058
ID: 9980, Ім'я: Вікторія, Прізвище: Петренко, ДН: 25.08.1951, Тел: 06629665127
ID: 9981, Ім'я: Вікторія, Прізвище: Шевченко, ДН: 09.12.2011, Тел: 0660043972
ID: 9982, Ім'я: Сергій, Прізвище: Романюк, ДН: 19.06.1996, Тел: 0665229340
ID: 9983, Ім'я: Дмитро, Прізвище: Коваленко, ДН: 09.01.2016, Тел: 0678677389
ID: 9984, Ім'я: Олена, Прізвище: Романюк, ДН: 02.08.1981, Тел: 0502082828
ID: 9985, Ім'я: Дмитро, Прізвище: Косарук, ДН: 15.09.1967, Тел: 0677887654
ID: 9986, Ім'я: Тетяна, Прізвище: Мельник, ДН: 20.09.1990, Тел: 0678533533
ID: 9987, Ім'я: Іван, Прізвище: Косарук, ДН: 25.04.1976, Тел: 0663406156
ID: 9988, Ім'я: Тетяна, Прізвище: Шевченко, ДН: 04.08.1988, Тел: 0663883932
ID: 9989, Ім'я: Марія, Прізвище: Косарук, ДН: 22.07.1962, Тел: 0673126781
ID: 9990, Ім'я: Марія, Прізвище: Кравченко, ДН: 17.02.1956, Тел: 0664444433
ID: 9991, Ім'я: Юлія, Прізвище: Косарук, ДН: 22.02.1954, Тел: 0663234480
ID: 9992, Ім'я: Іван, Прізвище: Шевченко, ДН: 21.02.2004, Тел: 0675185480
ID: 9993, Ім'я: Дмитро, Прізвище: Косарук, ДН: 15.07.1953, Тел: 0958715774
ID: 9994, Ім'я: Дмитро, Прізвище: Кравченко, ДН: 14.01.1998, Тел: 0677520335
ID: 9995, Ім'я: Дмитро, Прізвище: Кравченко, ДН: 06.09.2003, Тел: 0953718527
ID: 9996, Ім'я: Юлія, Прізвище: Смирненко, ДН: 04.04.1963, Тел: 0501959724
ID: 9997, Ім'я: Андрій, Прізвище: Романюк, ДН: 15.03.2012, Тел: 0958768433
ID: 9998, Ім'я: Олександр, Прізвище: Романюк, ДН: 28.10.1999, Тел: 0509845576
ID: 9999, Ім'я: Марія, Прізвище: Мельник, ДН: 29.10.2005, Тел: 0674503645
ID: 10000, Ім'я: Сергій, Прізвище: Шевченко, ДН: 05.11.1951, Тел: 0664666509

Натисніть Enter для продовження...

```

```

D:\Projects\RG\Rdb\bin\Debug  X  +  v
ID: 1975, Ім'я: Назар Павлюк, Спец: Педіатр, ID Лікарні: 460
ID: 1976, Ім'я: Ігор Лисенко, Спец: Проктолог, ID Лікарні: 574
ID: 1977, Ім'я: Ірина Павлюк, Спец: Проктолог, ID Лікарні: 512
ID: 1978, Ім'я: Максим Мартинюк, Спец: Кардіолог, ID Лікарні: 803
ID: 1979, Ім'я: Світлана Попова, Спец: Кардіолог, ID Лікарні: 21
ID: 1980, Ім'я: Володимир Петров, Спец: Терапевт, ID Лікарні: 854
ID: 1981, Ім'я: Світлана Давидова, Спец: Кардіолог, ID Лікарні: 356
ID: 1982, Ім'я: Наталія Ковальчук, Спец: Терапевт, ID Лікарні: 812
ID: 1983, Ім'я: Наталія Петров, Спец: Терапевт, ID Лікарні: 404
ID: 1984, Ім'я: Назар Ковальчук, Спец: Терапевт, ID Лікарні: 155
ID: 1985, Ім'я: Тарас Іваненко, Спец: Кардіолог, ID Лікарні: 891
ID: 1986, Ім'я: Ігор Петров, Спец: Терапевт, ID Лікарні: 428
ID: 1987, Ім'я: Тарас Гончарук, Спец: Хірург, ID Лікарні: 301
ID: 1988, Ім'я: Ірина Петров, Спец: Педіатр, ID Лікарні: 615
ID: 1989, Ім'я: Тарас Ковальчук, Спец: Проктолог, ID Лікарні: 302
ID: 1990, Ім'я: Володимир Гончарук, Спец: Кардіолог, ID Лікарні: 563
ID: 1991, Ім'я: Максим Павлюк, Спец: Невролог, ID Лікарні: 852
ID: 1992, Ім'я: Назар Павлюк, Спец: Невролог, ID Лікарні: 959
ID: 1993, Ім'я: Наталія Захаров, Спец: Проктолог, ID Лікарні: 820
ID: 1994, Ім'я: Тарас Лисенко, Спец: Офтальмолог, ID Лікарні: 994
ID: 1995, Ім'я: Максим Павлюк, Спец: Невролог, ID Лікарні: 707
ID: 1996, Ім'я: Тарас Мартинюк, Спец: Хірург, ID Лікарні: 945
ID: 1997, Ім'я: Наталія Захаров, Спец: Офтальмолог, ID Лікарні: 502
ID: 1998, Ім'я: Ігор Лисенко, Спец: Терапевт, ID Лікарні: 237
ID: 1999, Ім'я: Тарас Ковальчук, Спец: Офтальмолог, ID Лікарні: 784
ID: 2000, Ім'я: Світлана Захаров, Спец: Проктолог, ID Лікарні: 591
ID: 2001, Ім'я: Тарас Попова, Спец: Кардіолог, ID Лікарні: 24

Натисніть Enter для продовження...

```

```
D:\Projects\VRGRdb\bin\Debug x + v
ID: 9974, Дата: 06.08.2025 12:29, Диагноз: Гастрит, ID Пациента: 6454, ID Лікаря: 1508
ID: 9975, Дата: 09.04.2025 16:58, Диагноз: Перелом, ID Пациента: 8383, ID Лікаря: 1705
ID: 9976, Дата: 20.01.2024 14:04, Диагноз: Вивих, ID Пациента: 7975, ID Лікаря: 255
ID: 9977, Дата: 21.08.2024 12:17, Диагноз: Вивих, ID Пациента: 2315, ID Лікаря: 1409
ID: 9978, Дата: 05.12.2024 8:23, Диагноз: Мігрень, ID Пациента: 6622, ID Лікаря: 1050
ID: 9979, Дата: 21.07.2024 17:18, Диагноз: Отруєння, ID Пациента: 9368, ID Лікаря: 560
ID: 9980, Дата: 28.06.2025 13:52, Диагноз: Здоровий, ID Пациента: 2885, ID Лікаря: 1428
ID: 9981, Дата: 26.12.2024 17:43, Диагноз: Вивих, ID Пациента: 819, ID Лікаря: 359
ID: 9982, Дата: 23.11.2025 14:00, Диагноз: Вивих, ID Пациента: 5673, ID Лікаря: 822
ID: 9983, Дата: 12.01.2025 15:08, Диагноз: Ангiна, ID Пациента: 6971, ID Лікаря: 1408
ID: 9984, Дата: 26.07.2025 17:14, Диагноз: Мігрень, ID Пациента: 1820, ID Лікаря: 1123
ID: 9985, Дата: 04.02.2024 13:48, Диагноз: Гастрит, ID Пациента: 8656, ID Лікаря: 557
ID: 9986, Дата: 20.02.2025 12:46, Диагноз: Отруєння, ID Пациента: 7221, ID Лікаря: 254
ID: 9987, Дата: 29.03.2024 14:03, Диагноз: ГРВІ, ID Пациента: 2118, ID Лікаря: 1182
ID: 9988, Дата: 02.02.2025 9:32, Диагноз: Ангiна, ID Пациента: 494, ID Лікаря: 1187
ID: 9989, Дата: 24.01.2025 14:57, Диагноз: ГРВІ, ID Пациента: 9396, ID Лікаря: 743
ID: 9990, Дата: 02.02.2025 11:48, Диагноз: Здоровий, ID Пациента: 4182, ID Лікаря: 36
ID: 9991, Дата: 10.04.2025 13:58, Диагноз: Мігрень, ID Пациента: 4883, ID Лікаря: 1605
ID: 9992, Дата: 09.12.2024 10:57, Диагноз: ГРВІ, ID Пациента: 1269, ID Лікаря: 939
ID: 9993, Дата: 28.03.2025 8:12, Диагноз: Ангiна, ID Пациента: 3280, ID Лікаря: 1172
ID: 9994, Дата: 02.02.2025 15:52, Диагноз: Отруєння, ID Пациента: 7116, ID Лікаря: 1372
ID: 9995, Дата: 21.04.2024 8:16, Диагноз: Мігрень, ID Пациента: 8971, ID Лікаря: 1398
ID: 9996, Дата: 30.08.2025 13:17, Диагноз: Отруєння, ID Пациента: 1898, ID Лікаря: 64
ID: 9997, Дата: 06.04.2025 16:26, Диагноз: Ангiна, ID Пациента: 8736, ID Лікаря: 1663
ID: 9998, Дата: 18.04.2025 14:01, Диагноз: Вивих, ID Пациента: 7017, ID Лікаря: 691
ID: 9999, Дата: 12.10.2025 14:40, Диагноз: Ангiна, ID Пациента: 9861, ID Лікаря: 1634
ID: 10000, Дата: 19.06.2025 11:06, Диагноз: Перелом, ID Пациента: 6972, ID Лікаря: 1956

Натисніть Enter для продовження...
```

```
D:\Projects\VRGRdb\bin\Debug x + v
ID: 975, Назва: Харківська міська лікарня, Адреса: проспект Шевченка, 98
ID: 976, Назва: Київська районна лікарня, Адреса: вулиця Соборності, 58
ID: 977, Назва: Львівська міська лікарня, Адреса: вулиця Грушевського, 6
ID: 978, Назва: Дніпровська міська лікарня, Адреса: бульвар Франка, 120
ID: 979, Назва: Харківська обласна лікарня, Адреса: проспект Шевченка, 194
ID: 980, Назва: Харківська обласна лікарня, Адреса: вулиця Франка, 147
ID: 981, Назва: Харківська міська лікарня, Адреса: бульвар Соборності, 6
ID: 982, Назва: Волинська районна лікарня, Адреса: вулиця Франка, 152
ID: 983, Назва: Київська обласна лікарня, Адреса: бульвар Соборності, 44
ID: 984, Назва: Дніпровська міська лікарня, Адреса: проспект Шевченка, 158
ID: 985, Назва: Київська районна лікарня, Адреса: вулиця Шевченка, 86
ID: 986, Назва: Волинська міська лікарня, Адреса: бульвар Грушевського, 160
ID: 987, Назва: Львівська міська лікарня, Адреса: вулиця Лесі Українки, 184
ID: 988, Назва: Луцька районна лікарня, Адреса: бульвар Лесі Українки, 180
ID: 989, Назва: Одеська районна лікарня, Адреса: проспект Шевченка, 80
ID: 990, Назва: Київська обласна лікарня, Адреса: вулиця Перемоги, 39
ID: 991, Назва: Львівська міська лікарня, Адреса: проспект Грушевського, 129
ID: 992, Назва: Львівська районна лікарня, Адреса: вулиця Грушевського, 162
ID: 993, Назва: Луцька районна лікарня, Адреса: бульвар Соборності, 175
ID: 994, Назва: Одеська обласна лікарня, Адреса: вулиця Грушевського, 196
ID: 995, Назва: Луцька міська лікарня, Адреса: вулиця Грушевського, 78
ID: 996, Назва: Дніпровська районна лікарня, Адреса: вулиця Перемоги, 173
ID: 997, Назва: Луцька районна лікарня, Адреса: вулиця Миру, 29
ID: 998, Назва: Київська обласна лікарня, Адреса: вулиця Соборності, 153
ID: 999, Назва: Харківська обласна лікарня, Адреса: бульвар Лесі Українки, 133
ID: 1000, Назва: Львівська районна лікарня, Адреса: проспект Лесі Українки, 10
ID: 1001, Назва: Одеська обласна лікарня, Адреса: бульвар Франка, 40

Натисніть Enter для продовження...
```

10000 записів пацієнтів, 10000 записів прийомів, 2000 записів лікарів і 1000 записів лікарень згенерувались разом за 12430мс.

-- Вхідний параметр (count) = 10000

-- 1. Генеруємо 10000 пацієнтів і повертаємо їх ID

WITH generated_patients AS (

INSERT INTO patient (name, surname, "day of birth", phone)

SELECT

(ARRAY['Іван', 'Петро', 'Олександр', 'Сергій', 'Андрій', 'Дмитро', 'Марія',
'Анна', 'Олена', 'Тетяна', 'Вікторія', 'Юлія'])[floor(random() * 12 + 1)::int],

(ARRAY['Мельник', 'Шевченко', 'Коваленко', 'Бондаренко', 'Ткаченко',
'Кравченко', 'Олійник', 'Петренко', 'Іванов', 'Сидоренко', 'Романюк',
'Косарук'])[floor(random() * 12 + 1)::int],

DATE '1950-01-01' + (RANDOM() * (365 * 70))::integer,
(ARRAY['050', '095', '066', '067'])[floor(random() * 4 + 1)::int] ||
LPAD((RANDOM() * 9999999)::int::text, 7, '0')

FROM generate_series(1, 10000) AS s(id)

RETURNING id

),

-- 2. Генеруємо лікарні (10000 / 10 + 1 = 1001) і повертаємо їх ID

generated_hospitals AS (

INSERT INTO hospital (name, address)

SELECT

(ARRAY['Київська', 'Львівська', 'Луцька', 'Харківська', 'Одеська',
'Дніпровська', 'Волинська'])[floor(random() * 7 + 1)::int]

|| ' ' ||
(ARRAY['районна', 'обласна', 'міська'])[floor(random() * 3 + 1)::int]

|| ' лікарня',
(ARRAY['вулиця', 'проспект', 'бульвар'])[floor(random() * 3 + 1)::int]

|| ' ' ||
(ARRAY['Шевченка', 'Лесі Українки', 'Франка', 'Грушевського', 'Перемоги',
'Миру', 'Соборності'])[floor(random() * 7 + 1)::int]

|| ' ' ||
(floor(random() * 200 + 1)::int)::text

FROM generate_series(1, 10000/ 10 + 1) AS s(id)

RETURNING id

),

-- 3. Генеруємо лікарів (10000 / 5 + 1 = 2001) і повертаємо їх ID

generated_doctors AS (

INSERT INTO doctor (name, surname, specialization, hospital_id)

SELECT

(ARRAY['Олег', 'Ігор', 'Володимир', 'Максим', 'Назар', 'Тарас', 'Ольга',
'Ірина', 'Наталія', 'Світлана'])[floor(random() * 10 + 1)::int],

(ARRAY['Петров', 'Іваненко', 'Захаров', 'Павлюк', 'Лисенко', 'Гончарук',
'Попова', 'Давидова', 'Ковальчук', 'Мартинюк'])[floor(random() * 10 + 1)::int],

```

        (ARRAY['Терапевт', 'Хірург', 'Проктолог', 'Офтальмолог', 'Кардіолог',
'Невролог', 'Педіатр'])(floor(random() * 7 + 1)::int],
        h_rand.id
FROM generate_series(1, 10000 / 5 + 1) AS s(id)

CROSS JOIN LATERAL (
    SELECT id FROM generated_hospitals
    WHERE s.id > 0
    ORDER BY random() LIMIT 1
) AS h_rand
RETURNING id
)

```

-- 4. Генеруємо 10000 прийомів, використовуючи ID, отримані на попередніх кроках

```

INSERT INTO appointment (data, hour, diagnosis, patient_id, doctor_id)
SELECT
    DATE '2024-01-01' + (RANDOM() * 700)::integer,
    MAKE_TIME(floor(random() * 10 + 8)::int, floor(random() * 60)::int,
floor(random() * 60)::double precision),
    (ARRAY['Здоровий', 'ГРБІ', 'Перелом', 'Вивих', 'Мігрень', 'Ангіна', 'Гастрит',
'Отруєння'])(floor(random() * 8 + 1)::int],
    p_rand.id,
    d_rand.id
FROM generate_series(1, 10000) AS s(id)

CROSS JOIN LATERAL (
    SELECT id FROM generated_patients
    WHERE s.id > 0
    ORDER BY random() LIMIT 1
) AS p_rand

CROSS JOIN LATERAL (
    SELECT id FROM generated_doctors
    WHERE s.id > 0
    ORDER BY random() LIMIT 1
) AS d_rand;

```

Деталізоване завдання №3

Пошуковий запит №1

```

D:\Projects\RGRdb\bin\Debug  X  +  v
--- МЕДИЧНА СИСТЕМА ---
1. Додати
2. Видалити
3. Переглянути
4. Редагувати
5. Запустити генерацію 'рандомних' даних
6. Очистити усю базу даних
7. Пошук пацієнтів за прізвищем лікаря та датою
8. Вивести статистику по лікарях
9. Вивести статистику по лікарнях
0. Вихід
Ваш вибір: 7
--- Пошук пацієнтів ---
Натисніть Enter для продовження...

Введіть прізвище лікаря (або частину): Петров
Початкова дата (RRRR-ММ-ДД) (у форматі RRRR-ММ-ДД): 2024-01-01
Кінцева дата (RRRR-ММ-ДД) (у форматі RRRR-ММ-ДД): 2026-01-01

```

```
D:\Projects\RGRdb\bin\Debug x + v
--- Пошук пацієнтів --- (знайдено 995 записів за 240 мс)
Пациєнт: Вікторія Бондаренко (Тел: 0959728786), Лікар: Петров, Кількість прийомів: 3
Пациєнт: Андрій Коваленко (Тел: 0671994466), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Андрій Кравченко (Тел: 0672896848), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Андрій Ткаченко (Тел: 0663205166), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Іван Іванов (Тел: 0660033299), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Олександр Іванов (Тел: 0676036320), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Іван Шевченко (Тел: 0951684599), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Іван Мельник (Тел: 0956881585), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Юлія Олійник (Тел: 0662780248), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Юлія Шевченко (Тел: 0671480634), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Сергій Бондаренко (Тел: 0674181196), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Олена Коваленко (Тел: 0509063072), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Анна Сидоренко (Тел: 0954553334), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Юлія Олійник (Тел: 0953884712), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Юлія Мельник (Тел: 0957465074), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Юлія Романюк (Тел: 0955174702), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Марія Іванов (Тел: 0953560409), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Олександр Косарук (Тел: 0670979761), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Андрій Сидоренко (Тел: 0507039896), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Марія Олійник (Тел: 0953280092), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Дмитро Романюк (Тел: 0678249638), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Вікторія Бондаренко (Тел: 0665720629), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Дмитро Сидоренко (Тел: 0503533640), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Вікторія Шевченко (Тел: 0505833416), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Петро Шевченко (Тел: 0667852245), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Марія Коваленко (Тел: 0957647716), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Олена Ткаченко (Тел: 0501866107), Лікар: Петров, Кількість прийомів: 2
Пациєнт: Олександр Ткаченко (Тел: 0675957857), Лікар: Петров, Кількість прийомів: 2

WHERE s.id > 0
Пациєнт: Анна Олійник (Тел: 0666281715), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Тетяна Іванов (Тел: 0674999212), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Вікторія Іванов (Тел: 0958171136), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Петро Косарук (Тел: 0956258277), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Дмитро Сидоренко (Тел: 0958061710), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Вікторія Бондаренко (Тел: 0669943894), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Іван Кравченко (Тел: 0509621661), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Юлія Коваленко (Тел: 0661764322), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Анна Бондаренко (Тел: 0952441156), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Вікторія Сидоренко (Тел: 0675391440), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Юлія Мельник (Тел: 0672054666), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Вікторія Косарук (Тел: 0670556419), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Сергій Коваленко (Тел: 0956669422), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Тетяна Ткаченко (Тел: 0677576546), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Анна Кравченко (Тел: 0505901396), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Олександр Ткаченко (Тел: 0505959755), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Олександр Кравченко (Тел: 0676621802), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Андрій Іванов (Тел: 0665381949), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Іван Романюк (Тел: 0958834816), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Петро Кравченко (Тел: 0500132864), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Тетяна Сидоренко (Тел: 0503601281), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Вікторія Кравченко (Тел: 0505753548), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Марія Коваленко (Тел: 0671352192), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Сергій Мельник (Тел: 0954979545), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Олександр Сидоренко (Тел: 0959911885), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Олена Петренко (Тел: 0669247116), Лікар: Петров, Кількість прийомів: 1
Пациєнт: Сергій Іванов (Тел: 0504220690), Лікар: Петров, Кількість прийомів: 1

Натисніть Enter для продовження...
```

Результат виконання Пошуку №1 (статистика пацієнтів у лікаря)

Пошуковий запит №2

```
D:\Projects\RGRdb\bin\Debug x + v
--- МЕДИЧНА СИСТЕМА ---
1. Додати
2. Видалити
3. Переглянути
4. Редагувати
5. Запустити генерацію 'рандомних' даних
6. Очистити усю базу даних
7. Пошук пацієнтів за прізвищем лікаря та датою
8. Вивести статистику по лікарях
9. Вивести статистику по лікарнях
0. Вихід
Ваш вибір: 8
--- Пошук: Статистика прийомів по лікарях ---
Натисніть Enter для продовження...
Введіть спеціалізацію (або частину, напр. 'хірург'): Хірург
Початкова дата (RRRR-ММ-ДД) (у форматі RRRR-ММ-ДД): 2024-01-01
Кінцева дата (RRRR-ММ-ДД) (у форматі RRRR-ММ-ДД): 2026-01-01
```



```
D:\Projects\RGRdb\bin\Debug  X + v

--- Статистика по лікарях --- (знайдено 296 записів за 161 мс)

Лікар: Наталія Павлюк (Хірург), Кількість прийомів: 12
Лікар: Наталія Захаров (Хірург), Кількість прийомів: 11
Лікар: Максим Ковальчук (Хірург), Кількість прийомів: 11
Лікар: Наталія Мартинюк (Хірург), Кількість прийомів: 11
Лікар: Наталія Гончарук (Хірург), Кількість прийомів: 11
Лікар: Світлана Давидова (Хірург), Кількість прийомів: 10
Лікар: Ольга Павлюк (Хірург), Кількість прийомів: 10
Лікар: Олег Петров (Хірург), Кількість прийомів: 10
Лікар: Олег Павлюк (Хірург), Кількість прийомів: 10
Лікар: Наталія Іваненко (Хірург), Кількість прийомів: 10
Лікар: Максим Павлюк (Хірург), Кількість прийомів: 9
Лікар: Наталія Лисенко (Хірург), Кількість прийомів: 9
Лікар: Максим Попова (Хірург), Кількість прийомів: 9
Лікар: Максим Давидова (Хірург), Кількість прийомів: 9
Лікар: Ольга Іваненко (Хірург), Кількість прийомів: 9
Лікар: Тарас Захаров (Хірург), Кількість прийомів: 9
Лікар: Ольга Мартинюк (Хірург), Кількість прийомів: 9
Лікар: Тарас Петров (Хірург), Кількість прийомів: 9
Лікар: Ольга Давидова (Хірург), Кількість прийомів: 8
Лікар: Максим Давидова (Хірург), Кількість прийомів: 8
Лікар: Тарас Мартинюк (Хірург), Кількість прийомів: 8
Лікар: Володимир Іваненко (Хірург), Кількість прийомів: 8
Лікар: Тарас Лисенко (Хірург), Кількість прийомів: 8
Лікар: Максим Попова (Хірург), Кількість прийомів: 8
Лікар: Ігор Попова (Хірург), Кількість прийомів: 8
Лікар: Ольга Гончарук (Хірург), Кількість прийомів: 8
Лікар: Ігор Ковальчук (Хірург), Кількість прийомів: 8
Лікар: Наталія Петров (Хірург), Кількість прийомів: 8
```

Результат виконання Пошуку №2 (статистика по лікарях)

Пошуковий запит №3

```
D:\Projects\RGRdb\bin\Debug  X + v

--- МЕДИЧНА СИСТЕМА ---

1. Додати
2. Видалити
3. Переглянути
4. Редагувати

5. Запустити генерацію 'рандомних' даних
6. Очистити усю базу даних

7. Пошук пацієнтів за прізвищем лікаря та датою
8. Вивести статистику по лікарях
9. Вивести статистику по лікарнях

0. Вихід
Ваш вибір: 9
--- Пошук: Статистика по лікарнях ---
Натисніть Enter для продовження...

Введіть адресу лікарні (або частину, напр. 'Шевченка'): Шевченка
Введіть діагноз (або частину, напр. 'ГРВІ'): ГРВІ
```

```
D:\Projects\RGRdb\bin\Debug  X + v

--- Статистика по лікарнях --- (знайдено 77 записів за 189 мс)

Лікарня: Дніпровська міська лікарня (проспект Шевченка, 4), Кількість унікальних пацієнтів: 5
Лікарня: Одеська обласна лікарня (проспект Шевченка, 51), Кількість унікальних пацієнтів: 4
Лікарня: Волинська обласна лікарня (проспект Шевченка, 191), Кількість унікальних пацієнтів: 4
Лікарня: Волинська районна лікарня (проспект Шевченка, 60), Кількість унікальних пацієнтів: 4
Лікарня: Луцька обласна лікарня (проспект Шевченка, 195), Кількість унікальних пацієнтів: 4
Лікарня: Харківська міська лікарня (проспект Шевченка, 98), Кількість унікальних пацієнтів: 4
Лікарня: Луцька обласна лікарня (вулиця Шевченка, 90), Кількість унікальних пацієнтів: 4
Лікарня: Одеська районна лікарня (проспект Шевченка, 80), Кількість унікальних пацієнтів: 4
Лікарня: Волинська обласна лікарня (вулиця Шевченка, 121), Кількість унікальних пацієнтів: 4
Лікарня: Київська районна лікарня (проспект Шевченка, 176), Кількість унікальних пацієнтів: 4
Лікарня: Дніпровська міська лікарня (бульвар Шевченка, 121), Кількість унікальних пацієнтів: 3
Лікарня: Дніпровська обласна лікарня (бульвар Шевченка, 14), Кількість унікальних пацієнтів: 3
Лікарня: Одеська обласна лікарня (вулиця Шевченка, 153), Кількість унікальних пацієнтів: 3
Лікарня: Волинська обласна лікарня (проспект Шевченка, 198), Кількість унікальних пацієнтів: 3
Лікарня: Одеська районна лікарня (проспект Шевченка, 81), Кількість унікальних пацієнтів: 3
Лікарня: Луцька міська лікарня (проспект Шевченка, 62), Кількість унікальних пацієнтів: 3
Лікарня: Одеська міська лікарня (вулиця Шевченка, 66), Кількість унікальних пацієнтів: 3
Лікарня: Волинська обласна лікарня (вулиця Шевченка, 172), Кількість унікальних пацієнтів: 3
Лікарня: Львівська міська лікарня (бульвар Шевченка, 128), Кількість унікальних пацієнтів: 3
Лікарня: Дніпровська районна лікарня (проспект Шевченка, 198), Кількість унікальних пацієнтів: 2
Лікарня: Луцька районна лікарня (вулиця Шевченка, 98), Кількість унікальних пацієнтів: 2
Лікарня: Луцька районна лікарня (проспект Шевченка, 189), Кількість унікальних пацієнтів: 2
Лікарня: Київська районна лікарня (проспект Шевченка, 84), Кількість унікальних пацієнтів: 2
Лікарня: Київська обласна лікарня (вулиця Шевченка, 101), Кількість унікальних пацієнтів: 2
Лікарня: Луцька районна лікарня (бульвар Шевченка, 91), Кількість унікальних пацієнтів: 2
Лікарня: Одеська обласна лікарня (проспект Шевченка, 151), Кількість унікальних пацієнтів: 2
Лікарня: Львівська районна лікарня (проспект Шевченка, 38), Кількість унікальних пацієнтів: 2
Лікарня: Львівська обласна лікарня (бульвар Шевченка, 91), Кількість унікальних пацієнтів: 2
```

Результат виконання Пошуку №3 (статистика по лікарнях)

Запит підраховує, скільки прийомів мав кожен пацієнт у лікаря з певним прізвищем у заданому діапазоні дат:

```
SELECT
    p.name, p.surname, p.phone,
    d.surname AS doctor_surname,
    COUNT(a.id) AS appointment_count
FROM patient p
JOIN appointment a ON p.id = a.patient_id
JOIN doctor d ON a.doctor_id = d.id
WHERE
    d.surname ILIKE '%Петров%' AND
    a.data BETWEEN '2024-01-01' AND '2026-01-01'
GROUP BY
    p.id, p.name, p.surname, p.phone, d.surname
ORDER BY
    appointment_count DESC;
```

Запит підраховує, скільки прийомів провів кожен лікар певної спеціалізації за вказаний період:

```
SELECT
    d.name,
    d.surname,
    d.specialization,
    COUNT(a.id) AS appointment_count
FROM doctor d
LEFT JOIN appointment a ON d.id = a.doctor_id
WHERE
    d.specialization ILIKE '%Хірург%' AND
    a.data BETWEEN '2024-01-01' AND '2026-01-01'
GROUP BY
    d.id, d.name, d.surname, d.specialization
ORDER BY
    appointment_count DESC;
```

Запит підраховує, скільки унікальних пацієнтів із певним діагнозом прийняла кожна лікарня, що знаходиться за певною адресою:

```
SELECT
    h.name,
    h.address,
    COUNT(DISTINCT a.patient_id) AS patient_count
FROM hospital h
JOIN doctor d ON h.id = d.hospital_id
JOIN appointment a ON d.id = a.doctor_id
WHERE
    h.address ILIKE '%Шевченка%' AND
    a.diagnosis ILIKE '%ГРБІ%'
GROUP BY
    h.id, h.name, h.address
ORDER BY
    patient_count DESC;
```

Деталізоване завдання №4

Модуль **Model** (реалізований у класі `DatabaseModel`) є ключовим компонентом архітектурного шаблону MVC. Його головна відповідальність — це повна інкапсуляція всієї логіки взаємодії з базою даних PostgreSQL.

Це єдиний модуль у програмі, який "знає" про існування СУБД, мови SQL та бібліотеки `Npgsql`. Він повністю приховує від `Controller` та `View` деталі реалізації запитів.

Керування підключенням

- **GetConnectionAsync**: (Приватний метод) Відкриває та повертає нове асинхронне з'єднання з базою даних PostgreSQL.

```
25 references
private async Task<NpgsqlConnection> GetConnectionAsync()
{
    var conn = new NpgsqlConnection(_connectionString);
    await conn.OpenAsync();
    return conn;
}
```

Create (Створення)

- **AddPatientAsync**: Додає нового пацієнта в таблицю `patient`.

```
public async Task AddPatientAsync(Patient patient)
{
    await using var conn = await GetConnectionAsync();
    var sql = "INSERT INTO patient (name, surname, \"day of birth\", phone) " +
        "VALUES (@name, @surname, @dob, @phone)";
    await using var cmd = new NpgsqlCommand(sql, conn);

    cmd.Parameters.AddWithValue("name", patient.Name);
    cmd.Parameters.AddWithValue("surname", patient.Surname);
    cmd.Parameters.AddWithValue("dob", patient.DayOfBirth);
    cmd.Parameters.AddWithValue("phone", patient.Phone);

    await cmd.ExecuteNonQueryAsync();
}
```

- **AddDoctorAsync**: Додає нового лікаря в таблицю `doctor`, перехоплюючи помилку, якщо вказаний `hospital_id` не існує.

```
public async Task<string> AddDoctorAsync(Doctor doctor)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "INSERT INTO doctor (name, surname, specialization, hospital_id) " +
            "VALUES (@name, @surname, @spec, @h_id)";
        await using var cmd = new NpgsqlCommand(sql, conn);

        cmd.Parameters.AddWithValue("name", doctor.Name);
        cmd.Parameters.AddWithValue("surname", doctor.Surname);
        cmd.Parameters.AddWithValue("spec", doctor.Specialization);
        cmd.Parameters.AddWithValue("h_id", doctor.HospitalId);

        await cmd.ExecuteNonQuery();
        return "Лікаря успішно додано.";
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо додати лікаря. " +
            "Переконайтеся, що лікаря з ID = " + doctor.HospitalId + " існує.";
    }
    catch (Exception ex)
    {
        return $"Загальна помилка: {ex.Message}";
    }
}
```


- **AddAppointmentAsync:** Додає новий прийом у таблицю appointment, перехоплюючи помилку, якщо patient_id або doctor_id не існують.

```
public async Task<string> AddAppointmentAsync(Appointment app)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "INSERT INTO appointment (data, hour, diagnosis, patient_id, doctor_id) " +
            "VALUES (@data, @hour, @diag, @p_id, @d_id)";
        await using var cmd = new NpgsqlCommand(sql, conn);

        cmd.Parameters.AddWithValue("data", app.Data);
        cmd.Parameters.AddWithValue("hour", app.Hour);
        cmd.Parameters.AddWithValue("diag", app.Diagnosis);
        cmd.Parameters.AddWithValue("p_id", app.PatientId);
        cmd.Parameters.AddWithValue("d_id", app.DoctorId);

        await cmd.ExecuteNonQueryAsync();
        return "Прийом успішно додано.";
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо додати прийом. " +
            "Переконайтеся, що пацієнт та лікар з вказаними ID існують.";
    }
}
```

- **AddHospitalAsync:** Додає нову лікарню в таблицю hospital.

```
public async Task<string> AddHospitalAsync(Hospital hospital)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "INSERT INTO hospital (name, address) VALUES (@name, @address)";
        await using var cmd = new NpgsqlCommand(sql, conn);

        cmd.Parameters.AddWithValue("name", hospital.Name);
        cmd.Parameters.AddWithValue("address", hospital.Address);

        await cmd.ExecuteNonQueryAsync();
        return "Лікарню успішно додано.";
    }
    catch (Exception ex)
    {
        return $"ПОМИЛКА: {ex.Message}";
    }
}
```

Read (Читання)

- **GetAllPatientsAsync:** Виконує SELECT *, отримує та повертає повний список усіх пацієнтів (List<Patient>).

```
public async Task<List<Patient>> GetAllPatientsAsync()
{
    var patients = new List<Patient>();
    await using var conn = await GetConnectionAsync();
    var sql = "SELECT id, name, surname, \"day of birth\", phone FROM patient ORDER BY id";
    await using var cmd = new NpgsqlCommand(sql, conn);

    await using var reader = await cmd.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        patients.Add(new Patient
        {
            Id = reader.GetInt32(reader.GetOrdinal("id")),
            Name = reader.IsDBNull(reader.GetOrdinal("name")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("name")),
            Surname = reader.IsDBNull(reader.GetOrdinal("surname")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("surname")),
            DayOfBirth = reader.IsDBNull(reader.GetOrdinal("day of birth")) ? DateOnly.MinValue : DateOnly.FromDateTime(reader.GetDateTime(reader.GetOrdinal("day of birth"))),
            Phone = reader.IsDBNull(reader.GetOrdinal("phone")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("phone"))
        });
    }
    return patients;
}
```

- **GetAllDoctorsAsync:** Повертає повний список усіх лікарів (List<Doctor>).

```
public async Task<List<Doctor>> GetAllDoctorsAsync()
{
    var doctors = new List<Doctor>();
    await using var conn = await GetConnectionAsync();
    var sql = "SELECT id, name, surname, specialization, hospital_id FROM doctor ORDER BY id";
    await using var cmd = new NpgsqlCommand(sql, conn);

    await using var reader = await cmd.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        doctors.Add(new Doctor
        {
            Id = reader.GetInt32(reader.GetOrdinal("id")),
            Name = reader.IsDBNull(reader.GetOrdinal("name")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("name")),
            Surname = reader.IsDBNull(reader.GetOrdinal("surname")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("surname")),
            Specialization = reader.IsDBNull(reader.GetOrdinal("specialization")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("specialization")),
            HospitalId = reader.GetInt32(reader.GetOrdinal("hospital_id"))
        });
    }
    return doctors;
}
```

- **GetAllHospitalsAsync:** Повертає повний список усіх лікарень (List<Hospital>).

```
public async Task<List<Hospital>> GetAllHospitalsAsync()
{
    var hospitals = new List<Hospital>();
    await using var conn = await GetConnectionAsync();
    var sql = "SELECT id, name, address FROM hospital ORDER BY id";
    await using var cmd = new NpgsqlCommand(sql, conn);

    await using var reader = await cmd.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        hospitals.Add(new Hospital
        {
            Id = reader.GetInt32(reader.GetOrdinal("id")),
            Name = reader.IsDBNull(reader.GetOrdinal("name")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("name")),
            Address = reader.IsDBNull(reader.GetOrdinal("address")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("address"))
        });
    }
    return hospitals;
}
```

- **GetAllAppointmentsAsync:** Повертає повний список усіх прийомів (List<Appointment>).

```
public async Task<List<Appointment>> GetAllAppointmentsAsync()
{
    var appointments = new List<Appointment>();
    await using var conn = await GetConnectionAsync();
    var sql = "SELECT id, data, hour, diagnosis, patient_id, doctor_id FROM appointment ORDER BY id";
    await using var cmd = new NpgsqlCommand(sql, conn);

    await using var reader = await cmd.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        appointments.Add(new Appointment
        {
            Id = reader.GetInt32(reader.GetOrdinal("id")),
            Data = reader.IsDBNull(reader.GetOrdinal("data")) ? DateOnly.MinValue : DateOnly.FromDateTime(reader.GetDateTime(reader.GetOrdinal("data"))),
            Hour = reader.IsDBNull(reader.GetOrdinal("hour")) ? TimeOnly.MinValue : TimeOnly.FromTimeSpan(reader.GetTimeSpan(reader.GetOrdinal("hour"))),
            Diagnosis = reader.IsDBNull(reader.GetOrdinal("diagnosis")) ? "[Немає даних]" : reader.GetString(reader.GetOrdinal("diagnosis")),
            PatientId = reader.GetInt32(reader.GetOrdinal("patient_id")),
            DoctorId = reader.GetInt32(reader.GetOrdinal("doctor_id"))
        });
    }
    return appointments;
}
```

- **GetPatientByIdAsync:** Знаходить та повертає один об'єкт Patient за його id.

```
public async Task<Patient> GetPatientByIdAsync(int id)
{
    await using var conn = await GetConnectionAsync();
    var sql = "SELECT id, name, surname, \day of birth\, phone FROM patient WHERE id = @id";
    await using var cmd = new NpgsqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("id", id);

    await using var reader = await cmd.ExecuteReaderAsync();
    if (await reader.ReadAsync())
    {
        return new Patient
        {
            Id = reader.GetInt32(reader.GetOrdinal("id")),
            Name = reader.IsDBNull(reader.GetOrdinal("name")) ? "" : reader.GetString(reader.GetOrdinal("name")),
            Surname = reader.IsDBNull(reader.GetOrdinal("surname")) ? "" : reader.GetString(reader.GetOrdinal("surname")),
            DayOfBirth = reader.IsDBNull(reader.GetOrdinal("day of birth")) ? DateOnly.MinValue : DateOnly.FromDateTime(reader.GetDateTime(reader.GetOrdinal("day of birth"))),
            Phone = reader.IsDBNull(reader.GetOrdinal("phone")) ? "" : reader.GetString(reader.GetOrdinal("phone"))
        };
    }
    return null;
}
```

- **GetDoctorByIdAsync:** Знаходить та повертає один об'єкт Doctor за його id.

```
public async Task<Doctor> GetDoctorByIdAsync(int id)
{
    await using var conn = await GetConnectionAsync();
    var sql = "SELECT id, name, surname, specialization, hospital_id FROM doctor WHERE id = @id";
    await using var cmd = new NpgsqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("id", id);

    await using var reader = await cmd.ExecuteReaderAsync();
    if (await reader.ReadAsync())
    {
        return new Doctor
        {
            Id = reader.GetInt32(reader.GetOrdinal("id")),
            Name = reader.IsDBNull(reader.GetOrdinal("name")) ? "" : reader.GetString(reader.GetOrdinal("name")),
            Surname = reader.IsDBNull(reader.GetOrdinal("surname")) ? "" : reader.GetString(reader.GetOrdinal("surname")),
            Specialization = reader.IsDBNull(reader.GetOrdinal("specialization")) ? "" : reader.GetString(reader.GetOrdinal("specialization")),
            HospitalId = reader.GetInt32(reader.GetOrdinal("hospital_id"))
        };
    }
    return null;
}
```

- **GetHospitalByIdAsync:** Знаходить та повертає один об'єкт Hospital за його id.

```
public async Task<Hospital> GetHospitalByIdAsync(int id)
{
    await using var conn = await GetConnectionAsync();
    var sql = "SELECT id, name, address FROM hospital WHERE id = @id";
    await using var cmd = new NpgsqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("id", id);

    await using var reader = await cmd.ExecuteReaderAsync();
    if (await reader.ReadAsync())
    {
        return new Hospital
        {
            Id = reader.GetInt32(reader.GetOrdinal("id")),
            Name = reader.IsDBNull(reader.GetOrdinal("name")) ? "" : reader.GetString(reader.GetOrdinal("name")),
            Address = reader.IsDBNull(reader.GetOrdinal("address")) ? "" : reader.GetString(reader.GetOrdinal("address"))
        };
    }
    return null;
}
```

- **GetAppointmentByIdAsync:** Знаходить та повертає один об'єкт Appointment за його id.

```
public async Task<Appointment> GetAppointmentByIdAsync(int id)
{
    await using var conn = await GetConnectionAsync();
    var sql = "SELECT id, data, hour, diagnosis, patient_id, doctor_id FROM appointment WHERE id = @id";
    await using var cmd = new NpgsqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("id", id);

    await using var reader = await cmd.ExecuteReaderAsync();
    if (await reader.ReadAsync())
    {
        return new Appointment
        {
            Id = reader.GetInt32(reader.GetOrdinal("id")),
            Data = reader.IsDBNull(reader.GetOrdinal("data")) ? DateOnly.MinValue : DateOnly.FromDateTime(reader.GetDateTime(reader.GetOrdinal("data"))),
            Hour = reader.IsDBNull(reader.GetOrdinal("hour")) ? TimeOnly.MinValue : TimeOnly.FromTimeSpan(reader.GetTimeSpan(reader.GetOrdinal("hour"))),
            Diagnosis = reader.IsDBNull(reader.GetOrdinal("diagnosis")) ? "" : reader.GetString(reader.GetOrdinal("diagnosis")),
            PatientId = reader.GetInt32(reader.GetOrdinal("patient_id")),
            DoctorId = reader.GetInt32(reader.GetOrdinal("doctor_id"))
        };
    }
    return null;
}
```

Update (Оновлення)

- **UpdatePatientAsync:** Оновлює дані існуючого пацієнта в базі даних за його id.

```
public async Task<string> UpdatePatientAsync(Patient patient)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "UPDATE patient SET name = @name, surname = @surname, \"day of birth\" = @dob, phone = @phone " +
            "WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);

        cmd.Parameters.AddWithValue("name", patient.Name);
        cmd.Parameters.AddWithValue("surname", patient.Surname);
        cmd.Parameters.AddWithValue("dob", patient.DayOfBirth);
        cmd.Parameters.AddWithValue("phone", patient.Phone);
        cmd.Parameters.AddWithValue("id", patient.Id);

        int rowsAffected = await cmd.ExecuteNonQueryAsync();
        return rowsAffected > 0 ? "Дані пацієнта оновлено." : "ПОМИЛКА: Пацієнта не знайдено.";
    }
    catch (Exception ex)
    {
        return $"ПОМИЛКА: {ex.Message}";
    }
}
```

- **UpdateDoctorAsync:** Оновлює дані існуючого лікаря, перехоплюючи помилку, якщо новий hospital_id є недійсним.

```

public async Task<string> UpdateDoctorAsync(Doctor doctor)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "UPDATE doctor SET name = @name, surname = @surname, specialization = @spec, hospital_id = @h_id " +
            "WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);

        cmd.Parameters.AddWithValue("name", doctor.Name);
        cmd.Parameters.AddWithValue("surname", doctor.Surname);
        cmd.Parameters.AddWithValue("spec", doctor.Specialization);
        cmd.Parameters.AddWithValue("h_id", doctor.HospitalId);
        cmd.Parameters.AddWithValue("id", doctor.Id);

        int rowsAffected = await cmd.ExecuteNonQuery();
        return rowsAffected > 0 ? "Дані лікаря оновлено." : "ПОМИЛКА: Лікаря не знайдено.";
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо оновити. Лікарня з ID = " + doctor.HospitalId + " не існує.";
    }
    catch (Exception ex)
    {
        return $"ПОМИЛКА: {ex.Message}";
    }
}

```

- **UpdateHospitalAsync:** Оновлює назву та адресу існуючої лікарні.

```

public async Task<string> UpdateHospitalAsync(Hospital hospital)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "UPDATE hospital SET name = @name, address = @address WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);

        cmd.Parameters.AddWithValue("name", hospital.Name);
        cmd.Parameters.AddWithValue("address", hospital.Address);
        cmd.Parameters.AddWithValue("id", hospital.Id);

        int rowsAffected = await cmd.ExecuteNonQuery();
        return rowsAffected > 0 ? "Дані лікарні оновлено." : "ПОМИЛКА: Лікарню не знайдено.";
    }
    catch (Exception ex)
    {
        return $"ПОМИЛКА: {ex.Message}";
    }
}

```

- **UpdateAppointmentAsync:** Оновлює всі дані існуючого прийому, перехоплюючи помилки недійсних patient_id або doctor_id.

```

public async Task<string> UpdateAppointmentAsync(Appointment app)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "UPDATE appointment SET data = @data, hour = @hour, diagnosis = @diag, " +
            "patient_id = @p_id, doctor_id = @d_id WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);

        cmd.Parameters.AddWithValue("data", app.Data);
        cmd.Parameters.AddWithValue("hour", app.Hour);
        cmd.Parameters.AddWithValue("diag", app.Diagnosis);
        cmd.Parameters.AddWithValue("p_id", app.PatientId);
        cmd.Parameters.AddWithValue("d_id", app.DoctorId);
        cmd.Parameters.AddWithValue("id", app.Id);

        int rowsAffected = await cmd.ExecuteNonQuery();
        return rowsAffected > 0 ? "Дані прийому оновлено." : "ПОМИЛКА: Прийом не знайдено.";
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо оновити. Переконайтеся, що пацієнт та лікар існують.";
    }
    catch (Exception ex)
    {
        return $"ПОМИЛКА: {ex.Message}";
    }
}

```

Delete (Видалення)

- **DeletePatientAsync:** Видаляє пацієнта за id. Перехоплює помилку (23503), якщо у пацієнта є пов'язані прийоми.

```

public async Task<string> DeletePatientAsync(int patientId)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "DELETE FROM patient WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("id", patientId);

        int rowsAffected = await cmd.ExecuteNonQueryAsync();

        if (rowsAffected > 0)
        {
            return "Пацієнта успішно видалено.";
        }
        else
        {
            return "ПОМИЛКА: Пацієнта з ID = " + patientId + " не знайдено.";
        }
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо видалити пацієнта, оскільки за ним закріплені 'прийоми'.";
    }
    catch (Exception ex)
    {
        return $"Загальна помилка: {ex.Message}";
    }
}

```

- **DeleteDoctorAsync:** Видаляє лікаря за id. Перехоплює помилку, якщо у лікаря є пов'язані прийоми.

```

public async Task<string> DeleteDoctorAsync(int doctorId)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "DELETE FROM doctor WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("id", doctorId);

        int rowsAffected = await cmd.ExecuteNonQueryAsync();

        if (rowsAffected > 0)
        {
            return "Лікаря успішно видалено.";
        }
        else
        {
            return "ПОМИЛКА: Лікаря з ID = " + doctorId + " не знайдено.";
        }
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо видалити лікаря, оскільки за ним закріплені 'прийоми'.";
    }
    catch (Exception ex)
    {
        return $"Загальна помилка: {ex.Message}";
    }
}

```

- **DeleteAppointmentAsync:** Видаляє прийом за id.

```

public async Task<string> DeleteAppointmentAsync(int appointmentId)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "DELETE FROM appointment WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("id", appointmentId);

        int rowsAffected = await cmd.ExecuteNonQueryAsync();

        if (rowsAffected > 0)
        {
            return "Прийом успішно видалено.";
        }
        else
        {
            return "ПОМИЛКА: Прийом з ID = " + appointmentId + " не знайдено.";
        }
    }
    catch (Exception ex)
    {
        return $"Загальна помилка: {ex.Message}";
    }
}

```

- **DeleteHospitalAsync:** Видаляє лікарню за id. Перехоплює помилку, якщо за лікарнею закріплені лікарі.

```
public async Task<string> DeleteHospitalAsync(int hospitalId)
{
    try
    {
        await using var conn = await GetConnectionAsync();
        var sql = "DELETE FROM hospital WHERE id = @id";
        await using var cmd = new NpgsqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("id", hospitalId);
        await cmd.ExecuteNonQueryAsync();
        return "Лікарню успішно видалено.";
    }
    catch (PostgresException ex) when (ex.SqlState == "23503")
    {
        return "ПОМИЛКА: Неможливо видалити лікарню, оскільки за нею закріплені лікарі.";
    }
    catch (Exception ex)
    {
        return $"Загальна помилка: {ex.Message}";
    }
}
```

Адміністрування та Пошук

- **GenerateRandomDataAsync**: Виконує єдиний, складний SQL-запит (з WITH та CROSS JOIN LATERAL) для заповнення всіх таблиць вказаною кількістю (count) реалістичних, випадково згенерованих даних.

```
public async Task<string> GenerateRandomDataAsync(int count)
{
    var sql = $"
    WITH generated_patients AS (
        INSERT INTO patient (name, surname, ""day of birth"", phone)
        SELECT
            (ARRAY['Іван', 'Петро', 'Олександр', 'Сергій', 'Андрій', 'Дмитро', 'Марія', 'Анна', 'Олена', 'Тетяна', 'Бікторія', 'Вілія'])[floor(random() * 12 + 1)::int],
            (ARRAY['Мельник', 'Шевченко', 'Коваленко', 'Бондаренко', 'Ткаченко', 'Кравченко', 'Олійник', 'Петренко', 'Іванов', 'Сидоренко', 'Романюк', 'Носарук'])[floor(random() * 12 + 1)::int],
            DATE '1950-01-01' + (RANDOM() * (365 * 70))::integer,
            (ARRAY['050', '095', '066', '067'])[floor(random() * 4 + 1)::int] || LPAD((RANDOM() * 9999999)::int::text, 7, '0')
        FROM generate_series(1, {count}) AS s(id)
        RETURNING id
    ),
    generated_hospitals AS (
        INSERT INTO hospital (name, address)
        SELECT
            (ARRAY['Київська', 'Львівська', 'Луцька', 'Харківська', 'Одеська', 'Дніпровська', 'Волинська'])[floor(random() * 7 + 1)::int]
            || ' ' ||
            (ARRAY['районна', 'обласна', 'міська'])[floor(random() * 3 + 1)::int]
            || ' лікарня',
            (ARRAY['вулиця', 'проспект', 'бульвар'])[floor(random() * 3 + 1)::int]
            || ' ' ||
            (ARRAY['Шевченка', 'Лесі Українки', 'Франка', 'Грушевського', 'Перемоги', 'Миру', 'Соборності'])[floor(random() * 7 + 1)::int]
            || ' ',
            (floor(random() * 200 + 1)::int)::text
        FROM generate_series(1, {count / 10 + 1}) AS s(id)
        RETURNING id
    ),
    generated_doctors AS (
        INSERT INTO doctor (name, surname, specialization, hospital_id)
        SELECT
            (ARRAY['Олександр', 'Ігор', 'Володимир', 'Максим', 'Назар', 'Тарас', 'Ольга', 'Ірина', 'Марія', 'Світлана'])[floor(random() * 10 + 1)::int],
            (ARRAY['Петров', 'Іваненко', 'Захаров', 'Павлик', 'Лисенко', 'Гончарук', 'Попова', 'Давидова', 'Ковальчук', 'Мартинюк'])[floor(random() * 10 + 1)::int],
            (ARRAY['Терапевт', 'Хірург', 'Проктолог', 'Офтальмолог', 'Кардіолог', 'Невролог', 'Педіатр'])[floor(random() * 7 + 1)::int],
            h_rand.id
        FROM generate_series(1, {count / 5 + 1}) AS s(id)

        CROSS JOIN LATERAL (
            SELECT id FROM generated_hospitals
            WHERE s.id > 0
            ORDER BY random() LIMIT 1
        ) AS h_rand

    ) AS s_rand

    RETURNING id
    )
    INSERT INTO appointment (data, hour, diagnosis, patient_id, doctor_id)
    SELECT
        DATE '2024-01-01' + (RANDOM() * 700)::integer,
        MAKE_TIME(floor(random() * 10 + 0)::int, floor(random() * 60)::int, floor(random() * 60)::double precision),
        (ARRAY['Здоровий', 'ГРВІ', 'Перелом', 'Вивих', 'Нірвань', 'Ангіна', 'Гастрит', 'Отруєння'])[floor(random() * 8 + 1)::int],
        p_rand.id,
        d_rand.id
    FROM generate_series(1, {count}) AS s(id)

    CROSS JOIN LATERAL (
        SELECT id FROM generated_patients
        WHERE s.id > 0
        ORDER BY random() LIMIT 1
    ) AS p_rand

    CROSS JOIN LATERAL (
        SELECT id FROM generated_doctors
        WHERE s.id > 0
        ORDER BY random() LIMIT 1
    ) AS d_rand;

";

    try
    {
        await using var conn = await GetConnectionAsync();
        await using var cmd = new NpgsqlCommand(sql, conn);
        cmd.CommandTimeout = 300; // 5
        var sw = Stopwatch.StartNew();
        int rowsAffected = await cmd.ExecuteNonQueryAsync();
        sw.Stop();
        return $"Успішно згенеровано дані. Загальний час: {sw.ElapsedMilliseconds} мс.";
    }
    catch (Exception ex)
    {
        return $"Помилка генерації: {ex.Message}";
    }
}
```


- **ClearAllDataAsync:** Повністю очищує всі таблиці (TRUNCATE ... CASCADE) та скидає лічильники id до 1.

```
public async Task<string> ClearAllDataAsync()
{
    try
    {
        await using var conn = await GetConnectionAsync();

        var sql = @"
        TRUNCATE hospital RESTART IDENTITY CASCADE;
        TRUNCATE patient RESTART IDENTITY CASCADE;
        ";

        await using var cmd = new NpgsqlCommand(sql, conn);
        await cmd.ExecuteNonQuery();

        return "Успіх! Усі дані з таблиць видалено, лічильники ID скинуто до 1.";
    }
    catch (Exception ex)
    {
        return $"ПОМИЛКА: Не вдалося очистити базу даних. {ex.Message}";
    }
}
```

- **SearchPatientsByDoctorAndDateAsync:** Виконує аналітичний запит (Пошук №1), який рахує (GROUP BY) кількість візитів кожного пацієнта до лікаря (за прізвищем) у межах діапазону дат. Повертає результат та час виконання.

```
public async Task<(List<string> results, long timeMs)> SearchPatientsByDoctorAndDateAsync(
    string doctorSurnamePattern, DateOnly startDate, DateOnly endDate)
{
    var results = new List<string>();
    var sw = Stopwatch.StartNew();

    var sql = @"
    SELECT
        p.name, p.surname, p.phone,
        d.surname AS doctor_surname,
        COUNT(a.id) AS appointment_count
    FROM patient p
    JOIN appointment a ON p.id = a.patient_id
    JOIN doctor d ON a.doctor_id = d.id
    WHERE
        d.surname ILIKE @surnamePattern AND
        a.data BETWEEN @startDate AND @endDate
    GROUP BY
        p.id, p.name, p.surname, p.phone, d.surname
    ORDER BY
        appointment_count DESC;
    ";

    await using var conn = await GetConnectionAsync();
    await using var cmd = new NpgsqlCommand(sql, conn);

    cmd.Parameters.AddWithValue("surnamePattern", $"{doctorSurnamePattern}%");
    cmd.Parameters.AddWithValue("startDate", startDate);
    cmd.Parameters.AddWithValue("endDate", endDate);

    await using var reader = await cmd.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        results.Add(
            $"Пацієнт: {reader["name"]} {reader["surname"]} (Тел: {reader["phone"]}), " +
            $"Лікар: {reader["doctor_surname"]}, " +
            $"Кількість прийомів: {reader["appointment_count"]}";
    }

    sw.Stop();
    return (results, sw.ElapsedMilliseconds);
}
```

- **SearchDoctorStatisticsAsync:** Виконує аналітичний запит (Пошук №2), який рахує (GROUP BY) кількість прийомів для лікарів певної спеціалізації у межах діапазону дат. Повертає результат та час виконання.

```

public async Task<List<string> results, long timeMs> SearchDoctorStatisticsAsync(
    string specializationPattern, DateOnly startDate, DateOnly endDate)
{
    var results = new List<string>();
    var sw = Stopwatch.StartNew();

    var sql = @"
SELECT
    d.name,
    d.surname,
    d.specialization,
    COUNT(a.id) AS appointment_count
FROM doctor d
LEFT JOIN appointment a ON d.id = a.doctor_id
WHERE
    d.specialization ILIKE @specPattern AND
    a.data BETWEEN @startDate AND @endDate
GROUP BY
    d.id, d.name, d.surname, d.specialization
ORDER BY
    appointment_count DESC;
";

    await using var conn = await GetConnectionAsync();
    await using var cmd = new NpgsqlCommand(sql, conn);

    cmd.Parameters.AddWithValue("specPattern", $"{specializationPattern}%");
    cmd.Parameters.AddWithValue("startDate", startDate);
    cmd.Parameters.AddWithValue("endDate", endDate);

    await using var reader = await cmd.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        results.Add(
            $"Лікар: {reader["name"]} {reader["surname"]} ({reader["specialization"]}), " +
            $"Кількість прийомів: {reader["appointment_count"]}");
    }

    sw.Stop();
    return (results, sw.ElapsedMilliseconds);
}

```

- **SearchHospitalStatisticsAsync:** Виконує аналітичний запит (Пошук №3), який рахує (COUNT(DISTINCT)) кількість унікальних пацієнтів з певним діагнозом для лікарень за певною адресою. Повертає результат та час виконання.

```

public async Task<List<string> results, long timeMs> SearchHospitalStatisticsAsync(
    string addressPattern, string diagnosisPattern)
{
    var results = new List<string>();
    var sw = Stopwatch.StartNew();

    var sql = @"
SELECT
    h.name,
    h.address,
    COUNT(DISTINCT a.patient_id) AS patient_count
FROM hospital h
JOIN doctor d ON h.id = d.hospital_id
JOIN appointment a ON d.id = a.doctor_id
WHERE
    h.address ILIKE @addressPattern AND
    a.diagnosis ILIKE @diagnosisPattern
GROUP BY
    h.id, h.name, h.address
ORDER BY
    patient_count DESC;
";

    await using var conn = await GetConnectionAsync();
    await using var cmd = new NpgsqlCommand(sql, conn);

    cmd.Parameters.AddWithValue("addressPattern", $"{addressPattern}%");
    cmd.Parameters.AddWithValue("diagnosisPattern", $"{diagnosisPattern}%");

    await using var reader = await cmd.ExecuteReaderAsync();
    while (await reader.ReadAsync())
    {
        results.Add(
            $"Лікарня: {reader["name"]} ({reader["address"]}), " +
            $"Кількість унікальних пацієнтів: {reader["patient_count"]}");
    }

    sw.Stop();
    return (results, sw.ElapsedMilliseconds);
}

```


Висновок

Під час виконання цієї розрахунково-графічної роботи було успішно розроблено та реалізовано консольний додаток на мові C# для взаємодії з базою даних «Медична система», створеною в СУБД PostgreSQL.

Проект було побудовано з чітким дотриманням архітектурного шаблону **MVC (Model-View-Controller)**, де кожен компонент було винесено в окремий файл: Model інкапсулював всю логіку роботи з базою, View відповідав за консольний інтерфейс, а Controller керував логікою програми. Взаємодія з базою даних відбувалася виключно засобами "**чистого**" SQL з використанням бібліотеки Npgsql, без залучення ORM, як того вимагало завдання.

У ході роботи було реалізовано повний набір **CRUD-операцій** (Create, Read, Update, Delete) для всіх сутностей бази даних (patient, doctor, hospital, appointment). Особливу увагу було приділено коректній обробці **обмежень цілісності даних**:

1. Реалізовано перехоплення (try...catch) специфічних винятків PostgreSQLException (код 23503) при спробі **видалення** "батьківських" записів (наприклад, hospital), що мають залежні "дочірні" записи (doctor).
2. Аналогічно оброблено помилки при вставці "дочірніх" записів (appointment) з неіснуючими зовнішніми ключами (patient_id або doctor_id).

Це дозволило уникнути аварійного завершення програми та виводити користувачу зрозумілі повідомлення про помилки.

Також було реалізовано функціонал **пакетної генерації** великої кількості (10 000+) реалістичних, рандомізованих даних. Ця операція виконується одним складним SQL-запитом, який використовує CTE (WITH), CROSS JOIN LATERAL та RETURNING id для коректної обробки зовнішніх ключів безпосередньо на стороні сервера.

На згенерованих даних було протестовано **три складні пошукові запити**, які, згідно з завданням, використовують JOIN кількох таблиць, фільтрацію WHERE за параметрами користувача та агрегацію GROUP BY (COUNT, COUNT(DISTINCT)). Програма вимірює та виводить час виконання кожного з цих запитів у мілісекундах.

У результаті було здобуто практичні навички з проєктування архітектури додатків (MVC), роботи з .NET-драйверами баз даних (Npgsql), написання складних SQL-запитів, реалізації надійного механізму обробки помилок

цілісності даних та тестування продуктивності запитів на великих обсягах даних.