Зображення, що містить будівля, ескіз, чорно-білий, панорама

Автоматично згенерований опис

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота № 1**

з дисципліни

**«Бази даних та засоби управління»**

**«Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL»**

Виконала: студентка 3 курсу

ФПМ групи КВ-12

Бурда Є. А.

Київ – 2023

**Загальне завдання до лабораторної роботи**

*Метою роботи* є здобуття вмінь проектування бази даних та практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

***Завдання* роботи полягає у наступному:**

1. Розробити модель «сутність-зв’язок» предметної галузі, обраної студентом самостійно, відповідно до пункту «Вимоги до ER-моделі».
2. Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL.
3. Виконати нормалізацію схеми бази даних до третьої нормальної форми (3НФ).
4. Ознайомитись із інструментарієм PostgreSQL та pgAdmin 4 та внести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

***Вимоги до ER-моделі***

1. Сутності моделі предметної галузі мають містити зв’язки типу 1:N або N:M.
2. Кількість сутностей у моделі – 3-4. Кількість атрибутів у кожній сутності: від двох до п’яти.
3. Передбачити наявність зв’язку з атрибутом.
4. Для побудови ER-діаграм використовувати одну із нотацій: Чена, “Пташиної лапки (Crow’s foot)”, UML.

***Вимоги до інструментарію***

1. Створення ER-діаграм: Google Docs (Drawing) або <https://www.draw.io/> або [https://www.lucidchart.com](https://www.lucidchart.com/)
2. Середовище для створення таблиць відлагодження SQL-запитів до бази даних – pgAdmin 4.
3. СУБД - PostgreSQL 13-15 (<https://www.postgresql.org/download/>).
4. **Модель «сутність-зв'язок» галузі «медична система для збереження даних пацієнтів»**

**Сутності**

**Patient** з атрибутами : ID пацієнта, ім’я, прізвище, дата народження, адреса.

**Medical Worker** з атрибутами: ID працівника, ім'я, прізвище, спеціальність.

**Medical Record** з атрибутами: ID запису, дата та час, опис симптомів, діагноз.

**Medicine** з атрибутами: ID лікарського засобу, назва, дозування

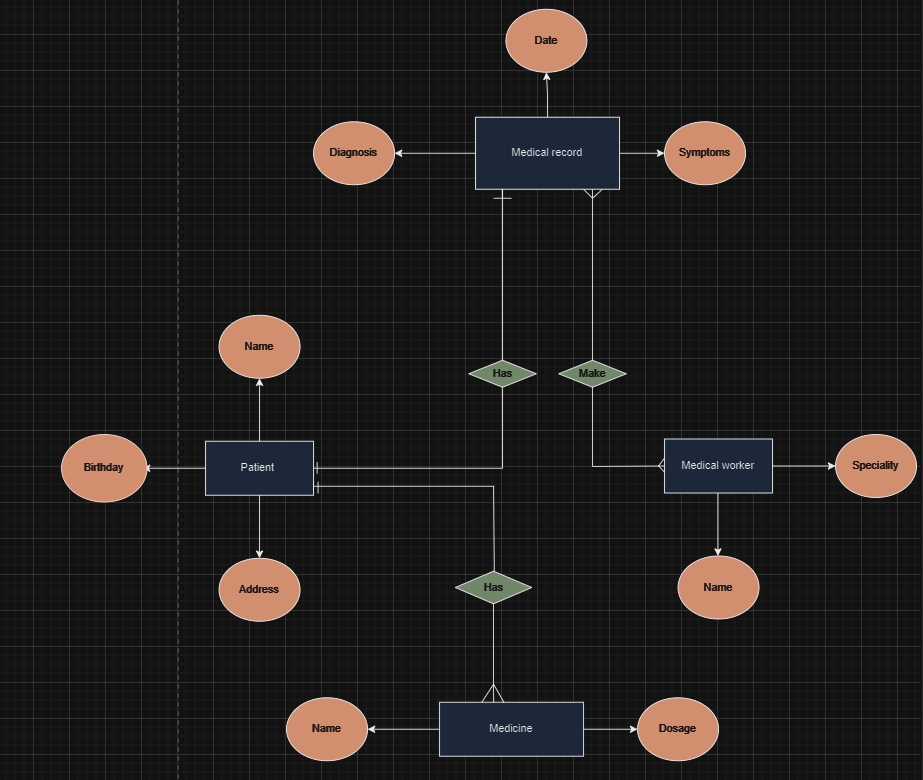
**Опис зв’язків**

Patient має medical record: вказує на те, що у кожного пацієнта є один медичний запис, та медичний запис відноситься до одного пацієнта.

Medical worker створює medical record: вказує на те, що медичний працівник може створювати багато медичних записів і багато медичних записів створені працівниками

Patient має medicine: вказує на те, що пацієнт може мати декілька видів ліків, але кожні ліки відносяться лише до одного пацієнта.

Medicine – це ліки для кожного пацієнта, вони можуть існувати без пацієнта так само як пацієнт може існувати без ліків.



ER-діаграма, побудована за нотацією Чена

Перетворення концептуальної моделі на логічну модель та схему бази даних

Сутність Patient перетворено в таблицю Patient

Сутність Medical record перетворено в таблицю Medical record

Сутність Medical worker перетворено в таблицю Medical worker

Сутність Medicine перетворено в таблицю Medicine

В моделі неможливо представити зв’язок N:M але в концептуальній моделі він існує між сутностями Medical worker i Medical record, для цього вводимо допоміжну таблицю Worker\_Record

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| Patient – користувіч, пацієнт | Patient\_ID - ID пацієнта  Name – ім’я  Birthday – дата народження  Address – адреса | **integer** (числовий)  **character varying** (рядок)  **date** (дата)  **character varying** (рядок) |
| Medical record – запис пацієнта, створений медичним працівником | Record\_ID – ID запису  Patient\_ID – ID пацієнта  Date – дата запису  Symptoms – симптоми пацієнта  Diagnosis – діагноз пацієнта | **integer** (числовий)  **date** (дата)  **character varying** (рядок)  **character varying** (рядок) |
| Medical worker – створює медичний запис | Worker\_ID – ID працівника  Name – ім’я  Speciality – спеціалізація | **integer** (числовий)  **character varying** (рядок)  **character varying** (рядок) |
| Medicine – ліки, які можуть бути в пацієнта | Medicine\_ID – ID ліків  Patient\_ID – ID пацієнта  Name – назва  Dosage – дозування | **integer** (числовий)  **character varying** (рядок)  **character varying** (рядок) |
| Worker\_Record – відповідність медичного працівника до медичного запису, який він зробив | Tab\_ID – ідентифікатор таблиці відповідності  Worker\_Name – ім’я працівника  Record\_ID – ID запису | **integer** (числовий)  **character varying** (рядок)  **integer** (числовий) |

**Зображення, що містить текст, знімок екрана, схема

Автоматично згенерований опис**

Схема бази даних

Функціональні залежності

**Patient:**

Patient\_ID – ID пацієнта

Name – ім’я

Birthday – дата народження

Address – адреса

Patient\_ID -> Name, Birthday, Address

Patient\_ID -> Name (залежить від коду користувача)

Patient\_ID -> Birthday (залежить від коду користувача)

Patient\_ID -> Address (залежить від коду користувача)

Birthday -> Address

Address -> Birthday

Patient\_ID -> Birthday -> Address (транзитивне відношення, але воно включає в себе ключовий атрибут)

Patient\_ID -> Address -> Birthday (транзитивне відношення, але воно включає в себе ключовий атрибут)

**Medical record:**

Record\_ID – ID запису

Patient\_ID – ID пацієнта

Date – дата запису

Symptoms – симптоми пацієнта

Diagnosis – діагноз пацієнта

Record\_ID -> Date, Symptoms, Diagnosis

Record\_ID -> Date

Record\_ID -> Symptoms

Record\_ID -> Diagnosis

Record\_ID -> Patient\_ID

Є декілька транзитивних відношень, але вони включають в себе ключовий атрибут.

**Medicine:**

Medicine\_ID – ID ліків

Patient\_ID – ID пацієнта

Name – назва ліків

Dosage – дозування ліків

Medicine\_ID -> Patient\_ID, Name, Dosage

Medicine\_ID -> Patient\_ID

Medicine\_ID -> Name

Medicine\_ID -> Dosage

Medical worker:

Worker\_ID – ID працівника

Name – ім’я

Speciality – спеціалізація

Worker\_ID -> Name, Speciality

Worker\_ID -> Name

Worker\_ID -> Speciality

Схема відповідає 1НФ, тому що:

* + Кожна «комірка» відношення має містити одне значення.
  + Кожен запис має бути унікальним (це властивість відношення).

Схема відповідає 2НФ, тому що:

* + Таблиця в 1NF (перша нормальна форма)
  + Кожен неключовий атрибут має функціонально залежати від усього ключа, а не від його частини
  + Якщо відношення має первинний ключ, що складається лише з одного атрибута, то відношення автоматично задовольняє 2NF.

Схема відповідає 3НФ, тому що:

* + Таблиця в 2NF (друга нормальна форма)
  + Кожен неключовий атрибут R нетранзитивно залежить від первинного ключа R.

**Схема бази даних в pgAdmin4**

**Зображення, що містить текст, знімок екрана, схема, ряд

Автоматично згенерований опис**

**Таблиці бази даних у pgAdmin4**

Patient

Зображення, що містить текст, знімок екрана, число, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

У цієї таблиці немає FK.

Medical Record

Зображення, що містить текст, знімок екрана, число, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис

Medicine

Зображення, що містить текст, знімок екрана, число, Шрифт

Автоматично згенерований опис

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Medical Worker

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Зображення, що містить текст, ряд, Шрифт, число

Автоматично згенерований опис

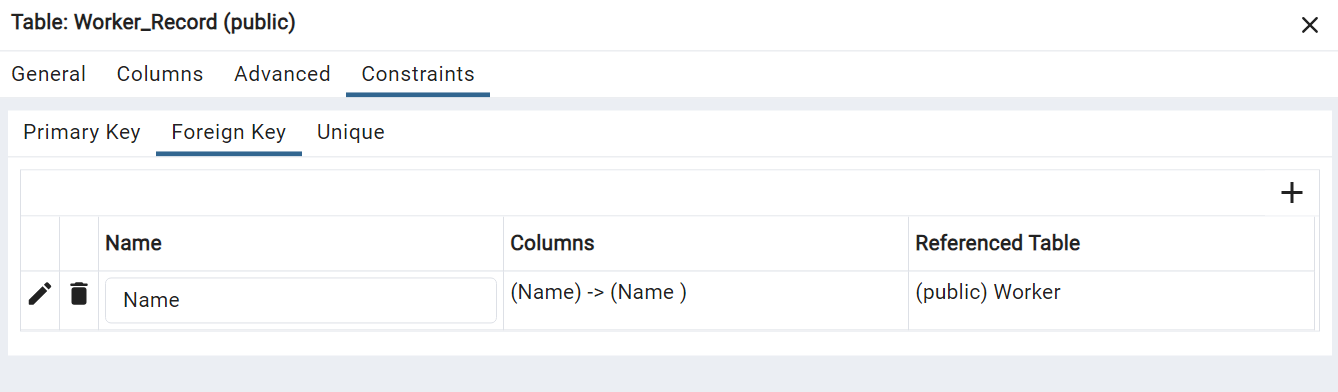
Worker\_Record

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис



Зображення, що містить текст, знімок екрана, ряд, Шрифт

Автоматично згенерований опис

**Дані в таблицях**

User

Зображення, що містить текст, Шрифт, число, ряд

Автоматично згенерований опис

Medical Record

Зображення, що містить текст, Шрифт, число, знімок екрана

Автоматично згенерований опис

Medicine

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Medical Worker

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Worker\_Record

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

**Таблиці в коді SQL**

-- Table: public.Patient

-- DROP TABLE IF EXISTS public."Patient";

CREATE TABLE IF NOT EXISTS public."Patient"

(

"Patient\_ID" integer NOT NULL,

"Name" character varying COLLATE pg\_catalog."default" NOT NULL,

"Birthday" date NOT NULL,

"Address" character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Patient\_pkey" PRIMARY KEY ("Patient\_ID")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Patient"

OWNER to postgres;

-- Table: public.Record

-- DROP TABLE IF EXISTS public."Record";

CREATE TABLE IF NOT EXISTS public."Record"

(

"Record\_ID" integer NOT NULL,

"Patient\_ID" integer,

"Date" date NOT NULL,

"Symptoms" character varying COLLATE pg\_catalog."default" NOT NULL,

"Diagnosis" character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Record\_pkey" PRIMARY KEY ("Record\_ID")

CONSTRAINT "Patient\_ID" FOREIGN KEY ("Patient\_ID")

REFERENCES public."Patient" ("Patient\_ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Record"

OWNER to postgres;

-- Table: public.Medicine

-- DROP TABLE IF EXISTS public."Medicine";

CREATE TABLE IF NOT EXISTS public."Medicine"

(

"Medicine\_ID" integer NOT NULL,

"Patient\_ID" integer,

"Name" character varying COLLATE pg\_catalog."default" NOT NULL,

"Dosage" character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Medicine\_pkey" PRIMARY KEY ("Medicine\_ID")

CONSTRAINT "Patient\_ID" FOREIGN KEY ("Patient\_ID")

REFERENCES public."Patient" ("Patient\_ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Medicine"

OWNER to postgres;

-- Table: public.Worker

-- DROP TABLE IF EXISTS public."Worker";

CREATE TABLE IF NOT EXISTS public."Worker"

(

"Worker\_ID" integer NOT NULL,

"Name" character varying COLLATE pg\_catalog."default" NOT NULL,

"Speciality" character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Worker\_pkey" PRIMARY KEY ("Worker\_ID")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Worker"

OWNER to postgres;

-- Table: public.Worker\_Record

-- DROP TABLE IF EXISTS public."Worker\_Record";

CREATE TABLE IF NOT EXISTS public."Worker\_Record"

(

"Tab\_ID" integer NOT NULL,

"Worker\_name" character varying COLLATE pg\_catalog."default",

"Record\_ID" integer,

CONSTRAINT "Worker\_Record\_pkey" PRIMARY KEY ("Tab\_ID")

CONSTRAINT "Record\_ID" FOREIGN KEY ("Record\_ID")

REFERENCES public."Record" ("Record\_ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

CONSTRAINT "Name" FOREIGN KEY ("Name")

REFERENCES public."Worker" ("Name") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Worker\_Record"

OWNER to postgres;

**Контрольні запитання**

1. Сформулювати призначення діаграм типу «сутність-зв’язок».

Ця діаграма відображає сутності (таблиці) та їх зв'язки в базі даних. Сутності представляють об'єкти, інформацію про які потрібно зберігати, а зв'язки вказують на взаємозв'язки між цими сутностями.

1. Назвати основні об’єкти схеми PostgreSQL.
   1. Таблиці (зберігається фактична інформація)
   2. Аргументи (представляють вхідні значення, які передаються для обробки)
   3. Зв’язки (допомагають визначити, як дані пов'язані між собою)
2. Навести приклади різних типів зв’язків у базах даних (1:1, 1:N, N:M).
   * 1:1 – клієнти та адреси ( кожен клієнт має одну адресу та кожна адреса належить тільки одному клієнту )
   * 1:N – товари та категорії ( кожна категорія має багато товарів, але кожен товар належить одній категорії)
   * N:M – автори та книги (кожен автор може мати багато книг та кожна книга може мати багато авторів)

<https://github.com/evaburda/lab1->

https://t.me/night\_fool