A black and white drawing of a building

Description automatically generated

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського» ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №1**

з дисципліни **Бази даних і засоби управління**

*на тему: “Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL”*

Виконав:

студент ІІI курсу

групи КВ-22

Савельєв О.С

Перевірив:

Павловский В. І.

Київ – 2024

**Мета:** здобуття вмінь проектування бази даних та практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

**Завдання:**

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.

**Опис предметної області**

Предметна область – система обліку медичних препаратів, представлена на наведених діаграмах, охоплює сукупність лікарських засобів, їх характеристик, виробників, категорій та хвороб, які ці препарати лікують або для яких вони протипоказані. Основна мета системи – створити структуру для зберігання та управління інформацією про медичні препарати, що дозволить ефективно організувати дані про їх виробництво, класифікацію, використання та протипоказання.

1. Ліки (Medicine)

Атрибути: medicine\_id, назва ліків (medicine\_name).

Призначення: збереження даних про ліки.

2. Категорія (Category)

Атрибути: category\_id, назва категорії (category\_name).

Призначення: збереження даних про категорії ліків.

3. Специфікації (Specs)

Атрибути: specs\_id, назва ліцензії (license\_name), умови видачі (terms\_of\_issue), форма випуску (medicine\_form).

Призначення: збереження специфікацій ліків.

4. Виробник (Manufacturer)

Атрибути: manufacturer\_id, назва виробника (manufacturer\_name), місце розташування виробничих потужностей (location\_of\_facilities).

Призначення: збереження інформації про виробників ліків.

5. Хвороба (Disease)

Атрибути: disease\_id, назва хвороби (disease\_name).

Призначення: збереження інформації про хвороби, які можуть бути лікувані або протипоказані для певних ліків.

**Опис зв’язків між сутностями:**

1. Зв'язок «Ліки» - «Категорія»:

Тип зв’язку: 1:N.

Опис: Одні ліки можуть належати до однієї категорії (category\_id), але одна категорія може мати багато ліків.

2. Зв'язок «Ліки» - «Специфікації»:

Тип зв’язку: 1:1.

Опис: Кожен набір ліків має конкретні специфікації (specs\_id).

3. Зв'язок «Ліки» - «Виробник»:

Тип зв’язку: 1:N.

Опис: Одні ліки можуть бути виготовлені лише одним виробником (manufacturer\_id), але один виробник може виготовляти багато ліків.

4. Зв'язок «Ліки» - «Хвороба (Лікування)»:

Тип зв’язку: 1:N.

Опис: Одні ліки можуть лікувати багато хвороб, але кожна хвороба може лікуватися багатьма ліками (treats).

5. Зв'язок «Ліки» - «Хвороба (Протипоказання)»:

Тип зв’язку: 1:N.

Опис: Одні ліки можуть мати протипоказання для багатьох хвороб, але кожна хвороба може мати протипоказання лише для певних ліків (contradicts).Изображение выглядит как диаграмма, снимок экрана, текст

Автоматически созданное описание

Рисунок 1 – ER-діаграма, побудована за нотацією Чена

**Перетворення концептуальної моделі у логічну модель та схему бази даних**

Сутність Medicine перетворено в таблицю Medicine з первинним ключем Medicine\_id та атрибутами Medicine\_name (назва препарату) та зовнішніми ключами Specs\_id, Category\_id, Teats\_id (ідентифікатори характеристик, категорій та захворювань, які лікуються).

Сутність Category перетворено в таблицю Category з первинним ключем Category\_id та атрибутами Category\_name (назва категорії).

Сутність Specs перетворено в таблицю Specs з первинним ключем Specs\_id та атрибутами License\_name (дані про ліцензію), Terms\_of\_issue (умови видачі), Medicine\_form (форма препарату).

Сутність Manufacturer перетворено в таблицю Manufacturer з первинним ключем Manufacturer\_id та атрибутами Manufacturer\_name (назва виробника) та Location\_of\_facilities (місцезнаходження виробничих потужностей).

Сутність Disease перетворено в таблицю Disease з первинним ключем Disease\_id та атрибутами Disease\_name (назва захворювання).

Оскільки в логічній моделі безпосередній зв’язок N:M існує між сутностями Medicine та Disease (ліки, які лікують конкретні захворювання), для його реалізації була створена таблиця Treats з первинним ключем tab\_id, що містить зовнішні ключі Medicine\_id та Disease\_id, що пов’язують ліки з відповідними захворюваннями.

Оскільки також існує зв’язок N:M між сутностями Medicine та Manufacturer (ліки, які виробляються конкретними виробниками), для його реалізації була створена таблиця Made\_by з первинним ключем tab\_id, що містить зовнішні ключі Medicine\_id та Manufacturer\_id, що пов’язують ліки з виробниками.

Графічне подання логічної моделі «Сутність-зв’язок» зображено на рисунку 2.

Изображение выглядит как текст, снимок экрана, диаграмма

Автоматически созданное описание

Рисунок 2 – Схема бази даних

Таблиця 1 ілюструє детальний перехід від однієї моделі до іншої.

Таблиця 1 – Опис об’єктів бази даних

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| *Medicine*– містить дані перпарати | *Medicine\_id* – унікальний ідентифікатор препарату | *integer* (число) |
| *Specs\_id* – унікальний ідентифікатор характеристик перпарату | *integer* (число) |
| *Category\_id* –ідентифікатор категорії перпарату | *integer* (число) |
| *Treats\_id* – Хвороба при якій препарат виписують | *integer* (число) |
| *Contradicts\_id* – Хвороба при якій препарат протипоказаний | *integer* (число) |
| *Medicine\_name* | *character varying* (рядок) |
| *Category*– містить дані про групу | *Category \_id* – унікальний ідентифікатор групи | *integer* (число) |
| *Category\_name* – назва групи | *character varying* (рядок) |
| *Disease*– містить дані про хворобу | *Disease\_id* – унікальний ідентифікатор хвороби | *integer* (число) |
| *Disease\_name – Назва хвороби* | *character varying* (рядок) |
| *Teacher* – містить дані про викладачів | *teacher\_id* – унікальний ідентифікатор викладача | *integer* (число) |
| *first\_name* – ім’я викладача | *character varying* (рядок) |
| *last\_name* – прізвище викладача | *character varying* (рядок) |
| *email* – електронна скринька викладача | *character varying* (рядок) |
| *Specs* – містить дані про препарат | *Spec\_id* – унікальний ідентифікатор характеристик | *integer* (число) |
| *License\_name* – Данні про ліцензю препарата | *character varying* (рядок) |
| *Terms\_of\_issue–* форма відпуску препарату за рецептом чи без | Bool(логічна змінна) |
| Medicine\_form– форма лікарського засобу | *character varying* (рядок) |
| *Manufacture*r – містить інформацію про виробника | *Manufacturer\_id* – ідентифікатор виробника | *integer* (число) |
| *Manufacturer\_Name* – Імя виробника | *character varying* (рядок) |
| *Location of facilities –* ідентифікатор препарату | *character varying* (рядок) |
| *Med\_Man\_junktion* –  Джанкшн таблиця для відповідності виробник лікарський засіб | *tab\_id* – унікальний ідентифікатор відповідності | *integer* (число) |
| *Manufacturer\_id* – ідентифікатор виробника | *integer* (число) |
| *Medicine\_id* – ідентифікатор препарату | *integer* (число) |

**Функціональні залежності для кожної таблиці**

1. - Medicine\_id → { Specs\_id, Category\_id, Treats\_id, Contradicts\_id, Medicine\_name, Manufacturer\_id }

2. Category:

- Category\_id → { category\_name }

3. Disease:

- Disease\_id → { Disease\_name }

4. Manufacturer:

- Manufacturer\_id → { Manufacturer\_Name, Location\_of\_facilities }

5. Specs:

- Specs\_id → { License\_name, Terms\_of\_issue, Medicine\_form }

6. Med\_Man\_junction:

- Med\_Man\_junction\_id → { Manufacturer\_id, Medicine\_id }

Ці функціональні залежності вказують на те, які атрибути в кожній таблиці визначаються від інших атрибутів. Це важливо для нормалізації та управління базою даних.

Транзитивні функціональні залежності виникають, коли один атрибут функціонально визначає інший через інший атрибут. Іншими словами, якщо A визначає B, а B визначає C, то ми можемо сказати, що A транзитивно визначає C. Дані таблиці не мають транзитивних функціональних залежностей.

**Відповідність схеми нормальним формам**

1. Щоб задовільнити умови 1НФ кожен атрибут в таблиці має бути атомарним, тобто:

* Кожна клітинка містить єдине значення;
* Кожен запис є унікальним.

Дана схема відповідає 1НФ.

1. Щоб схема відповідала 2НФ повинні виконуватись умови:

* Схема перебуває в 1НФ;
* Кожний неключовий атрибут функціонально залежить від цілого ключа.

У даній схемі кожна таблиця має власний унікальний ідентифікатор (ключ). Кожний атрибут у кожній таблиці залежить від цього унікального ідентифікатора. Тобто, схема також в НФ2.

1. Щоб схема відповідала 3НФ повинні виконуватись умови:

* Схема перебуває в 2НФ;
* Кожен неключовий атрибут функціонально залежить від усього ключа, а не його частини.

Оскільки дана схема в НФ2 та неключові атрибути не транзитивно залежать від інших неключових атрибутів, схема також в НФ3.

Висновок:

Схема бази даних відповідає нормальним формам НФ1, НФ2 та НФ3. Вона добре структурована і нормалізована, що сприяє ефективному та надійному зберіганню та обробці даних.

**Таблиці бази даних у pgAdmin4**

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

Categories

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

**Изображение выглядит как текст, линия, Шрифт, число

Автоматически созданное описание**

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Diseases

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание**

**Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание**

Manufacturers

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание**

**Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание**

**Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание**

Medicines

**Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание**

**Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание**

**Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание**

Specs

**Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание**

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Man\_Med\_tab

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Изображение выглядит как текст, диаграмма, линия, Параллельный

Автоматически созданное описание

**Таблиці в коді SQL**

BEGIN;

CREATE TABLE IF NOT EXISTS public."Categories"

(

"Category\_ID" integer NOT NULL,

"Category\_Name" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Category\_ID" PRIMARY KEY ("Category\_ID"),

CONSTRAINT "Categoties\_Category\_Name\_key" UNIQUE ("Category\_Name")

);

CREATE TABLE IF NOT EXISTS public."Diseases"

(

"Disease\_id" integer NOT NULL,

"Disease\_name" character varying(50) COLLATE pg\_catalog."default",

CONSTRAINT "Disease\_id\_prkey" PRIMARY KEY ("Disease\_id"),

CONSTRAINT "Name" UNIQUE ("Disease\_name")

);

CREATE TABLE IF NOT EXISTS public."Manufactures"

(

"Manufacturer\_id" integer NOT NULL,

"Manufacturer\_name" character varying(128) COLLATE pg\_catalog."default" NOT NULL,

"Location of facilities" character varying(128) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Manufcaturer\_id\_prkey" PRIMARY KEY ("Manufacturer\_id")

);

CREATE TABLE IF NOT EXISTS public."Medicines"

(

"Medicine\_id" integer NOT NULL,

"Manufacturer\_id" integer NOT NULL,

"Specs\_id" integer NOT NULL,

"Category\_id" integer NOT NULL,

"Treats\_id" integer NOT NULL,

"Contradicts\_id" integer NOT NULL,

"Medicine\_name" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Med\_id\_prk" PRIMARY KEY ("Medicine\_id"),

CONSTRAINT "Specs\_id" UNIQUE ("Specs\_id")

);

CREATE TABLE IF NOT EXISTS public."Specs"

(

"Specs\_id" integer NOT NULL,

"Lisence\_name" character varying(128) COLLATE pg\_catalog."default" NOT NULL,

"Term\_of\_issue" boolean NOT NULL,

"Medicine\_form" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Specs\_id\_prkey" PRIMARY KEY ("Specs\_id")

);

CREATE TABLE IF NOT EXISTS public.man\_med\_tab

(

"Tab\_id" integer NOT NULL,

"Medicine\_id" integer NOT NULL,

"Manufacturer\_id" integer NOT NULL

);

ALTER TABLE IF EXISTS public."Medicines"

ADD CONSTRAINT "Category\_fk" FOREIGN KEY ("Category\_id")

REFERENCES public."Categories" ("Category\_ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public."Medicines"

ADD CONSTRAINT "Contradict\_fk" FOREIGN KEY ("Contradicts\_id")

REFERENCES public."Diseases" ("Disease\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public."Medicines"

ADD CONSTRAINT "Manufacturer\_fk" FOREIGN KEY ("Manufacturer\_id")

REFERENCES public."Manufactures" ("Manufacturer\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public."Medicines"

ADD CONSTRAINT "Specs\_fk" FOREIGN KEY ("Specs\_id")

REFERENCES public."Specs" ("Specs\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

CREATE INDEX IF NOT EXISTS "Specs\_id"

ON public."Medicines"("Specs\_id");

ALTER TABLE IF EXISTS public."Medicines"

ADD CONSTRAINT "Treats\_fk" FOREIGN KEY ("Treats\_id")

REFERENCES public."Diseases" ("Disease\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.man\_med\_tab

ADD CONSTRAINT "Man\_fk" FOREIGN KEY ("Manufacturer\_id")

REFERENCES public."Manufactures" ("Manufacturer\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.man\_med\_tab

ADD CONSTRAINT "Med\_fk" FOREIGN KEY ("Medicine\_id")

REFERENCES public."Medicines" ("Medicine\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION;

END;

**Висновок**

У ході виконання лабораторної роботи була розроблена база даних для системи обліку медичних препаратів. База даних включає такі сутності: "Medicine", "Category", "Manufacturer", "Specs", "Disease", "Treats", "Contradicts" та "Med\_Man\_tab". Кожна з цих сутностей має відповідні атрибути, які дозволяють зберігати та керувати інформацією про медичні препарати, їх категорії, виробників, характеристики, хвороби, які ці препарати лікують або для яких вони протипоказані.

ER-діаграма бази даних була підготовлена для візуального відображення взаємозв'язків між сутностями. Нотація Чена була використана для позначення зв'язків та атрибутів.

Схема бази даних пройшла аналіз на відповідність нормальним формам НФ1, НФ2 та НФ3. Всі таблиці відповідають цим нормальним формам, що гарантує надійне та ефективне зберігання даних.

Окрім цього, була додана таблиця "Med\_Man\_tab", яка відображає зв’язки між виробниками та медичними препаратами, дозволяючи зберігати інформацію про відповідність кожного препарату його виробнику.

На останньому етапі були надані копії екранів з pgAdmin4, що демонструють властивості стовпців, обмеження та вміст таблиць бази даних у PostgreSQL.

Загалом, розроблена база даних відповідає поставленим завданням та вимогам, забезпечуючи структуроване та ефективне зберігання інформації про медичні препарати та їх використання.

*Посилання на GitHub:*

*https://github.com/AlexandrCC/Data\_Bases\_LR\_1.git*