

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»**

Виконав: студент 3 курсу

ФПМ групи КВ-12

Галюк М.Е.

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/>).

**Модель «сутність-зв'язок» галузі «здача та оренда нерухомості»**

**Сутності**

User з атрибутами: ID користувача, ім'я, прізвище, пошта, номер телефону.

Property з атрибутами: ID нерухомості, назва нерухомості, місце розташування нерухомості, ціна оренди нерухомості, к-сть кімнат.

Contract з атрибутами: ID договору, ID користувача, ID нерухомості, дата укладання договору.

Group з атрибутами: ID групи, ID користувача, стрічка новин, коментарі від користувачів під новиною.

**Опис звя’язків**

User – це користувач, який є зацікавленим в пошуку нерухомості. Один User може мати декілька контрактів та об 'єктів нерухомості.

Property – це об’єкт нерухомості на оренду та орендодавець в одній сутності.

Контракт – це договір між користувачем та орендодавцем на оренду приміщення. Контракт не може відбуватись без користувача або нерухомості. Контракт відбувається лише з одним користувачем, а от у користувача може бути багато контрактів. Стороною орендодавця є сам об'єкт нерухомості, тобто Property.

Group – це спільнота користувачів, де вони мають змогу переглядати певні короткі новини. Group може існувати без користувачів, так само як користувач може не входити до жодної групи.

A diagram of a company

Description automatically generated

Рисунок 1 – ER-діаграма, побудована за нотацією Чена

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

Сутність User було перетворено у таблицю User.

Сутність Propertyбуло перетворено у таблицю Property.

Сутність Groupбуло перетворено у таблицю Group.

Для зручності представлення, зв'язок Contract введено як допоміжну таблицю Contract.

В логічній моделі неможливий безпосередній зв'язок М:N, а в концептуальній моделі він існує між сутностями User і Group. Для його представлення було введено допоміжну таблицю User\_Group.

Назви таблиць і полів було перекладено англійською, для зручності наведено таблицю нижче:

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| **User –** користувач, орендар | **User\_ID –** ID користувача  **Name -** ім'я  **Email** - пошта **Phone** – номер телефону | **integer** (числовий)  **character varying** (рядок)  **character varying** (рядок)  **integer** (числовий) |
| **Property -** об'єкт нерухомості, орендодавець | **Property \_ID** – ID нерухомості  **Contract \_ID** – ID договору  **Location** – місце розташування  **Rental\_Price** – ціна оренди  **Num\_Of\_Rooms** – к-сть кімнат | **integer** (числовий)  **integer** (числовий)  **character varying** (рядок)  **integer** (числовий)  **integer** (числовий) |
| **Contract –** договір між орендарем та орендодавцем | **User\_ID –** ID користувача  **Contract \_ID** – ID договору  **Date –** дата укладання договору | **integer** (числовий)  **integer** (числовий)  **date** (дата) |
| **Group –** група людей з новинами | **Group\_Name –** назва групи  **News –** новина групи **Date –** дата новини | **character varying** (рядок)  **character varying** (рядок)  **date** (дата) |
| **User\_Group –** містить інформацію про відповідність користувача до групи, в якій він присутній | **Tab\_ID –** ідентифікатор таблиці відповідності  **Group\_Name –** назва групи  **User\_ID –** ID користувача | **integer** (числовий)    **character varying** (рядок)  **integer** (числовий) |

A screenshot of a computer

Description automatically generated

Рисунок 2 – Схема бази даних

**Функціональні залежності**

**User:**

**User\_ID –** ID користувача

**Name -** ім'я

**Email** - пошта  
**Phone** – номер телефону

User\_ID → Name, Email, Phone

User\_ID → Name (ім’я залежить від коду користувача)

User\_ID → Email(пошта залежить від коду користувача)

User\_ID → Phone(номер телефону залежить від коду користувача)

Email→ Phone

Phone→Email

User\_ID→Email→ Phone (транзитивне відношення, але воно включає в себе ключовий атрибут)

User\_ID→Phone→Email (транзитивне відношення, але воно включає в себе ключовий атрибут)

**Property:**

**Property \_ID** – ID нерухомості

**Contract \_ID** – ID договору

**Location** – місце розташування

**Rental\_Price** – ціна оренди

**Num\_Of\_Rooms** – к-сть кімнат

Property \_ID → Name, Location, Rental\_Price, Num\_Of\_Rooms

Property \_ID → Location

Property \_ID → Rental\_Price

Property \_ID → Num\_Of\_Rooms

Property \_ID → Contract \_ID

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

**Contract:**

**User\_ID –** ID користувача

**Contract \_ID** – ID договору

**Date –** дата укладання договору

Contract \_ID→ User\_ID, Date

Contract \_ID→ User\_ID

Contract \_ID→ Date

**Group:**

**Group\_Name** – назва групи

**News** – новини

**Date –** дата новини

Group\_Name →News, Date

Group\_Name → News

Group\_Name → Date

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 непростий атрибут R нетранзитивно залежить від кожного ключа R.

**A diagram of a computer

Description automatically generated**

Рисунок 3 – Схема бази даних у pgAdmin4

**Таблиці бази даних у pgAdmin4**

**User**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

Рисунок 4 – таблиця User

У цієї таблиці немає FK.

**Property**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Рисунок 5 – таблиця Property

**Contract**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

Рисунок 6 – таблиця Contract

**Group**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Рисунок 7 – таблиця Group

У цієї таблиці немає FK.

**Дані в таблицях**

**A screenshot of a computer

Description automatically generated**

Рисунок 8 – дані в таблиці User

**A screenshot of a computer

Description automatically generated**

Рисунок 9 – дані в таблиці Property

**A screenshot of a computer

Description automatically generated**

Рисунок 10 – дані в таблиці Contract

A screenshot of a computer

Description automatically generated

Рисунок 11 – дані в таблиці Group

A screenshot of a computer

Description automatically generated

Рисунок 12 – дані в таблиці User\_Group

**Таблиці в коді SQL**

-- Table: public.User

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

CREATE TABLE IF NOT EXISTS public."User"

(

"User\_ID" integer NOT NULL,

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

"Phone" integer NOT NULL,

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

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

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."User"

OWNER to postgres;

-- Table: public.Property

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

CREATE TABLE IF NOT EXISTS public."Property"

(

"Property\_ID" integer NOT NULL,

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

"Rental\_Price" integer NOT NULL,

"Num\_Of\_Rooms" integer NOT NULL,

"Contract\_ID" integer,

CONSTRAINT "Property\_pkey" PRIMARY KEY ("Property\_ID"),

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

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

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Property"

OWNER to postgres;

-- Table: public.Group

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

CREATE TABLE IF NOT EXISTS public."Group"

(

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

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

"Date\_Of\_News" date NOT NULL,

CONSTRAINT "Group\_pkey" PRIMARY KEY ("Group\_Name")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Group"

OWNER to postgres;

-- Table: public.Contract

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

CREATE TABLE IF NOT EXISTS public."Contract"

(

"Contract\_ID" integer NOT NULL,

"Date" date NOT NULL,

"User\_ID" integer NOT NULL,

CONSTRAINT "Contract\_pkey" PRIMARY KEY ("Contract\_ID"),

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

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

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Contract"

OWNER to postgres;

-- Table: public.User\_Group

-- DROP TABLE IF EXISTS public."User\_Group";

CREATE TABLE IF NOT EXISTS public."User\_Group"

(

"Tab\_ID" integer NOT NULL,

"User\_ID" integer,

"Group\_Name" character varying COLLATE pg\_catalog."default",

CONSTRAINT "User\_Group\_pkey" PRIMARY KEY ("Tab\_ID"),

CONSTRAINT "Group\_Name" FOREIGN KEY ("Group\_Name")

REFERENCES public."Group" ("Group\_Name") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

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

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

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."User\_Group"

OWNER to postgres;

**Контрольні запитання**

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

Для відображення сутностей та їх аргументів, а також зв’язків між сутностями. За допомогою візуалізації на діаграмі можна легше зрозуміти ці аспекти.

1. Назвати основні об’єкти схеми PostgreSQL.
   * Таблиці
   * Аргументи
   * Зв’язки
2. Навести приклади різних типів зв’язків у базах даних (1:1, 1:N, N:M).
   * 1:1 – студент та студентський квиток(ID)
   * 1:N – мати та діти
   * N:M – студент та студентський гурток

GitHub – https://github.com/kifs4/DB\_3\_course

Telegram - https://t.me/kifs4