

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

# Лабораторна робота №1

***з дисципліни «Бази даних»***

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

Виконав студентгрупи: КВ-33

ПІБ: Ткаченко Володимир

Перевірив: Павловський В. І.

**Київ 2025**

**Постановка задачі:**

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.

***Завдання №1:***

**Опис предметної галузі:**

Тема моєї бази даних: Система обліку відвідуваності на заняттях університету

**При створенні даної бази даних я виділив такі сутності:**

1. **Група студентів (Student\_group)** – представляє академічні групи:

* Атрибути: id групи, назва групи, факультет.

1. **Студент (Student)** – представляє окремого студента:

* Атрибути: id студента, ім’я, прізвище, електронна пошта, id групи.

1. **Викладач (Teacher)** – представляє викладачів університету:

* Атрибути: id викладача, ім’я, прізвище.

1. **Предмет (Subject)** – представляє навчальні дисципліни:

* Атрибути: id предмета, назва предмета.

1. **Призначення викладача (Teacher\_Subject)** – відображає, які викладачі ведуть які предмети:

* Атрибути: id викладача, id предмета.

1. **Заняття (Class)** – представляє окреме проведене заняття:

* Атрибути: id заняття, id предмета, id викладача, дата та час заняття, тип заняття (лекція, практика тощо).

1. **Відвідуваність (Attendance)** – представляє факт відвідування студентом конкретного заняття:

* Атрибути: id запису, id заняття, id студента, статус (присутній, відсутній, запізнився).

**Зв’язки:**

Зв’язок «Група студентів» – «Студент»:

* Тип зв'язку: 1 до N (одна група може містити багато студентів; один студент належить лише одній групі).

Зв’язок «Студент» - «Заняття»:

* Тип зв'язку: M до N (один студент може відвідувати багато занять; одне заняття може відвідати багато студентів).

Зв’язок «Предмет» - «Заняття»:

* Тип зв'язку: 1 до N (один предмет може мати багато занять; одне заняття належить лише одному предмету).

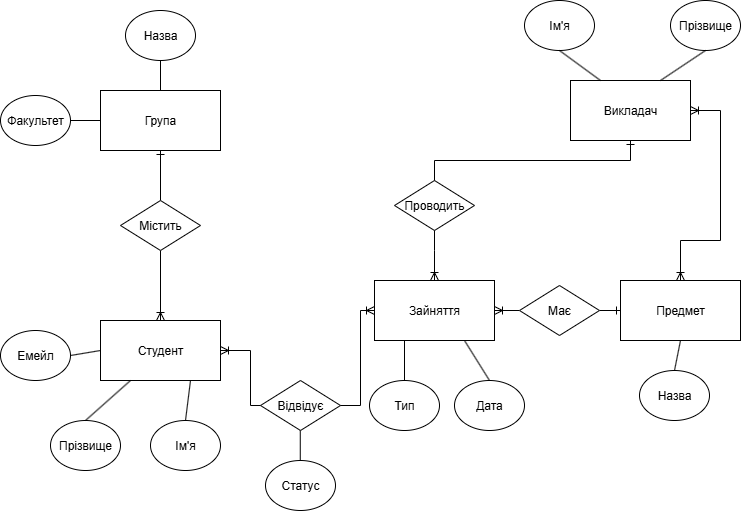
Зв’язок «Викладач» - «Заняття»:

* Тип зв'язку: 1 до N (один викладач може проводити багато занять; одне заняття веде лише один викладач).

Зв’язок «Викладач» - «Предмет»:

* Тип зв'язку: M до N (один викладач може викладати багато предметів, і один предмет може викладатися різними викладачами).

Графічне подання концептуальної моделі «Сутність-зв’язок» зображено на рисунку 1



*Рисунок 1 – ER-діаграма, побудована за нотацією” Пташина лапка”*

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

Сутність **"Група"** перетворено в таблицю **Student\_group** з первинним ключем **group\_id** та атрибутами **group\_name**, **faculty**.

Сутність **"Студент"** перетворено в таблицю **Student** з первинним ключем **student\_id** та атрибутами **last\_name**, **first\_name**, **email**. У таблиці є зовнішній ключ **group\_id**, який пов’язаний з таблицею **Student\_group**.

Сутність **"Викладач"** перетворено в таблицю **Teacher** з первинним ключем **teacher\_id** та атрибутами **last\_name**, **first\_name**.

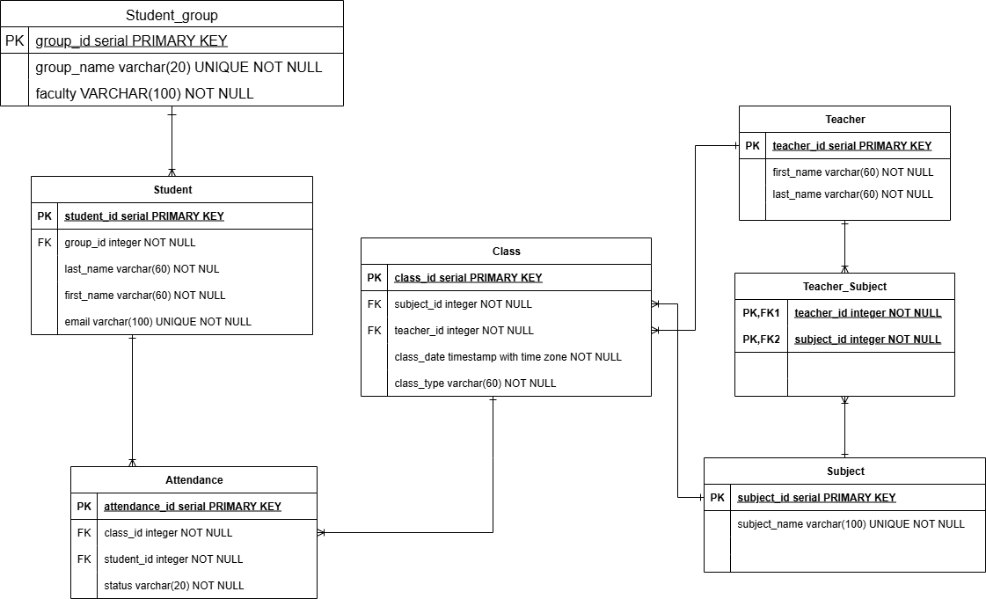
Сутність **"Предмет"** перетворено в таблицю **Subject** з первинним ключем **subject\_id** та атрибутом **subject\_name**.

Зв’язок M:N між викладачем і предметом зумовив появу додаткової таблиці **Teacher\_Subject** з комбінованим первинним ключем (**teacher\_id**, **subject\_id**) і зовнішніми ключами, які пов’язані з таблицями **Teacher** та **Subject** відповідно.

Сутність **"Зайняття"** перетворено в таблицю **Class** з первинним ключем **class\_id** та атрибутами **class\_date**, **class\_type**. У таблиці є зовнішні ключі **subject\_id** та **teacher\_id**, які пов’язані з таблицями **Subject** та **Teacher** відповідно.

Зв’язок M:N між студентом і заняттям зумовив появу додаткової таблиці **Attendance** з первинним ключем **attendance\_id** та атрибутом **status**. У таблиці є зовнішні ключі **class\_id** та **student\_id**, які пов’язані з таблицями **Class** та **Student** відповідно.

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



*Рисунок 2 – Схема бази даних*

Таблиця 1 ілюструє детальний перехід від однієї моделі до іншої.

|  |  |  |  |
| --- | --- | --- | --- |
| Сутність | | Атрибут | Тип атрибуту |
| Student – містить дані про студентів | | student\_id – унікальний ідентифікатор студента | serial (числовий) |
| *first\_name* - ім’я студента | character varying (рядок) |
| *last\_name* – прізвище студента | character varying (рядок) |
| *email* – електронна скринька користувача | character varying (рядок) |
| *group\_id – ідентифікатор групи, до якої належить студент* | integer (зовнішній ключ до Групи) |
| *Student\_group* – містить дані про групи студентів | | *group\_id –* унікальний ідентифікатор групи | serial (числовий) |
| *group\_name* – назва групи | character varying (рядок) |
| *faculty* – факултет, до якого належить група | character varying (рядок) |
| *Teacher* - містить дані про викладачів | | *teacher\_id*– унікальний ідентифікатор викладача | serial (числовий) |
| *first\_name* - ім’я викладача | character varying (рядок) |
| *last\_name* – прізвище викладача | character varying (рядок) |
| *Subject* – містить дані про предмет | | *subject\_id*- унікальний ідентифікатор предмету | serial (числовий) |
| *subject\_name* – назва предмету | character varying (рядок) |
| *Teacher\_Subject –* містить інформацію про про те, які предмети ведуть які викладачі | | *teacher\_id*- ідентифікатор викладача | integer (зовнішній ключ до Викладача) |
|  | | *subject\_id –* ідентифікатор предмету | integer (зовнішній ключ до Викладача) |
|  | |
|  | |
| *Class –* містить інформацію про заняття | | *class\_id* - унікальний ідентифікатор заняття | serial (числовий) |
|  | | *subject\_id –* ідентифікатор предмету | integer (зовнішній ключ до Предмету) |
|  | | *teacher\_id –* ідентифікатор викладача | integer (зовнішній ключ до Викладача) |
|  | | *class\_date –* дата проведення заняття | timestamp (дата та час) |
|  | | *class\_type –* тип заняття ( лабораторна, лекція і тп.) | character varying (рядок) |
| *Attendance –* містить інформацію про відвідування студентів на заняттях | *attendance\_id –* унікальний ідентифікатор запису про відвідування | | serial (числовий) |
| *class\_id –* ідентифікатор заняття | | integer (зовнішній ключ до Заняття) |
| *student\_id –* ідентифікатор студента | | integer (зовнішній ключ до Студента) |
| *status –* статус відвідування (присутній, запізнився, відсутній і тп.) | | character varying (рядок) |

*Таблиця 1 – Опис об’єктів бази даних*

**Функціональні залежності для кожної таблиці**

1. Student\_group:

group\_id -> group\_name, faculty

2. Student:

student\_id -> first\_name, last\_name, email, group\_id

group\_id -> first\_name, last\_name, email

3. Subject:

subject\_id -> subject\_name

4. Teacher\_Subject:

(teacher\_id, subject\_id) -> NULL (немає інших атрибутів окрім ключів, тому ця таблиця є чисто зв’язковою)

5. Class:

class\_id -> subject\_id, teacher\_id, class\_date, class\_type

subject\_id -> teacher\_id, class\_date, class\_type (не завжди вірно, бо різні викладачі можуть вести один предмет)

6. Attendance:

attendance\_id -> class\_id, student\_id, status

(class\_id, student\_id) -> status

Ці функціональні залежності вказують на те, які атрибути в кожній таблиці визначаються від інших атрибутів у моделі бази даних. Це важливо для забезпечення нормалізації та ефективного управління базою даних, оскільки правильне визначення функціональних залежностей дозволяє уникнути надмірності даних і забезпечує їхню цілісність.

Транзитивні функціональні залежності виникають тоді, коли один атрибут визначає інший через третій атрибут. Іншими словами, якщо атрибут A визначає атрибут B, а атрибут B визначає атрибут C, то ми можемо стверджувати, що атрибут A транзитивно визначає атрибут C. Однак у даній моделі транзитивних функціональних залежностей не виявлено, що вказує на відсутність надмірності даних і сприяє ефективному функціонуванню бази даних.

**Відповідність схеми нормальним формам**

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

* Усі атрибути атомарні (не подільні на менші частини).
* Немає повторюваних груп стовпців або масивів у таблиці.

Кожен атрибут у моїй моделі бази даних є атомарним, тобто не розбивається на кілька значень у одному полі (наприклад, прізвище і ім'я зберігаються окремо). Таким чином, схема ***відповідає вимогам 1НФ.***

1. Щоб схема відповідала 2НФ повинні виконуватись умови:

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

Схема бази даних перебуває в 1НФ. Усі неключові атрибути функціонально залежать від первинного ключа в кожній таблиці, і не має часткових залежностей. У таблиці із комбінованим ключем (Teacher\_Subject) немає додаткових атрибутів, тому часткових залежностей бути не може.Таким чином, моя схема ***відповідає 2НФ.***

1. Щоб схема відповідала 3НФ повинні виконуватись умови:

* Схема перебуває в 2НФ;
* Не неключові атрибути не залежать один від одного, а лише від ключа.

Оскільки схема в НФ2 та немає транзитивних залежностей між неключовими атрибутами, то неключові атрибути залежать безпосередньо від ключа, а не через інші неключові атрибути. У таблиці Student атрибут group\_id залежить від student\_id (ключа), і group\_name та faculty зберігаються в іншій таблиці – транзитивної залежності немає. У таблиці Class ідентифікатори subject\_id і teacher\_id залежать від class\_id, і ми не зберігаємо ім’я викладача чи назву предмета там же, тому немає дублювання чи транзитивних залежностей. У Attendance статус залежить тільки від комбінації (class\_id, student\_id) – транзитивність теж відсутня. Тобто схема також ***відповідає 3НФ***.

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.

Схема бази даних у pgAdmin 4 зображено на рисунку 3.

**A screenshot of a computer screen

AI-generated content may be incorrect.**

Рисунок 3 - Схема бази даних у pgAdmin 4

**Таблиці бази даних у pgAdmin4**

A screenshot of a computer

AI-generated content may be incorrect.

**Student\_group**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

У таблиці немає зовнішніх ключів

A screenshot of a computer

AI-generated content may be incorrect.

**Student**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Teacher**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

У цієї таблиці немає зовнішніх ключів (FK)

A screenshot of a phone

AI-generated content may be incorrect.

Subject

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

У цієї таблиці немає зовнішніх ключів (FK)

A screenshot of a computer

AI-generated content may be incorrect.

Teacher\_Subject

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Class**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Attendance**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**SQL код для таблиць**

BEGIN;

CREATE TABLE IF NOT EXISTS public.Student\_group

(

group\_id serial PRIMARY KEY,

group\_name varchar(20) UNIQUE NOT NULL,

faculty VARCHAR(100) NOT NULL

);

CREATE TABLE IF NOT EXISTS public.Student

(

student\_id serial PRIMARY KEY,

first\_name varchar(60) NOT NULL,

last\_name varchar(60) NOT NULL,

email varchar(100) UNIQUE NOT NULL,

group\_id integer NOT NULL,

FOREIGN KEY (group\_id) REFERENCES public.Student\_group (group\_id)

ON UPDATE NO ACTION

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS public.Teacher

(

teacher\_id serial PRIMARY KEY,

first\_name varchar(60) NOT NULL,

last\_name varchar(60) NOT NULL

);

CREATE TABLE IF NOT EXISTS public.Subject

(

subject\_id serial PRIMARY KEY,

subject\_name varchar(100) UNIQUE NOT NULL

);

CREATE TABLE IF NOT EXISTS public.Teacher\_Subject

(

teacher\_id integer NOT NULL,

subject\_id integer NOT NULL,

PRIMARY KEY (teacher\_id, subject\_id),

FOREIGN KEY (teacher\_id) REFERENCES public.Teacher (teacher\_id)

ON UPDATE NO ACTION

ON DELETE CASCADE,

FOREIGN KEY (subject\_id) REFERENCES public.Subject (subject\_id)

ON UPDATE NO ACTION

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS public.Class

(

class\_id serial PRIMARY KEY,

subject\_id integer NOT NULL,

teacher\_id integer NOT NULL,

class\_date timestamp with time zone NOT NULL,

class\_type varchar(60) NOT NULL,

FOREIGN KEY (subject\_id) REFERENCES public.Subject (subject\_id)

ON UPDATE NO ACTION

ON DELETE CASCADE,

FOREIGN KEY (teacher\_id) REFERENCES public.Teacher (teacher\_id)

ON UPDATE NO ACTION

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS public.Attendance

(

attendance\_id serial PRIMARY KEY,

class\_id integer NOT NULL,

student\_id integer NOT NULL,

status varchar(20) NOT NULL,

FOREIGN KEY (class\_id) REFERENCES public.Class (class\_id)

ON UPDATE NO ACTION

ON DELETE CASCADE,

FOREIGN KEY (student\_id) REFERENCES public.Student (student\_id)

ON UPDATE NO ACTION

ON DELETE CASCADE

);

END;

Посилання на GitHub:

https://github.com/Boooooiii/Lab1