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”*

Виконала:

студентка ІІI курсу

групи КВ-12

Ступницька С.М.

Перевірив:

Павловский В. І.

Київ – 2023

**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**

Дана предметна область – система обліку відвідуваності на заняттях університету. Вона охоплює сукупність предметів та інструментів, які спрямовані на ведення записів щодо присутніх та відсутніх студентів під час навчального процесу. Це допоможе моніторити ефективність навчання кожного студента з кожного предмету та формувати звітність згідно до вимог.

**Опис сутностей**

Для побудови бази даних обраної області, були виділені такі сутності:

1. Студент (Student)

Атрибути: ідентифікатор студента, ім’я, прізвище, електронна скринька.

Призначення: збереження даних щодо студентів.

1. Група (Group)

Атрибути: ідентифікатор групи, назва, курс, кількість студентів.

Призначення: збереження даних щодо групи.

1. Дисципліна (Discipline)

Атрибути: ідентифікатор дисципліни, предмет, аудиторія, дата та час.

Призначення: збереження даних щодо занять.

1. Викладач (Teacher)

Атрибути: ідентифікатор викладача, ім’я, прізвище, електронна скринька.

Призначення: збереження даних щодо викладачів.

1. Відвідуваність (Attendance)

Атрибути: ідентифікатор кожного запису, ідентифікатор студента, , ідентифікатор дисципліни, ідентифікатор викладача, дата.

Призначення: збереження даних щодо відвідуваності.

**Опис зв’язків між сутностями**

Зв’язок «Група» - «Студент» є зв’язком 1:N. Один студент може бути лише в одній групі, але в одній групі може бути багато студентів.

Зв’язок «Викладач» - «Дисципліна» є зв’язком N:M. Один викладач може викладати багато дисциплін, і одна дисципліна може бути проведена багатьма викладачами.

Зв’язок «Студент» - «Відвідуваність» є зв’язком 1:N. Один студент може мати багато записів в теці відвідуваності, а може й не мати, але один запис з теки відвідуваності відповідає лише одному студенту.

Зв’язок «Відвідуваність» - «Дисципліна» є зв’язком 1:N. Одна дисципліна може мати багато записів щодо відвідуваності, але один запис з теки відвідуваності відповідає лише одній дисципліні.

Зв’язок «Відвідуваність» - «Викладач» є зв’язком 1:N. Один викладач може вносити багато записів щодо відвідуваності, але один запис з теки відвідуваності відповідає лише одному викладачу.

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

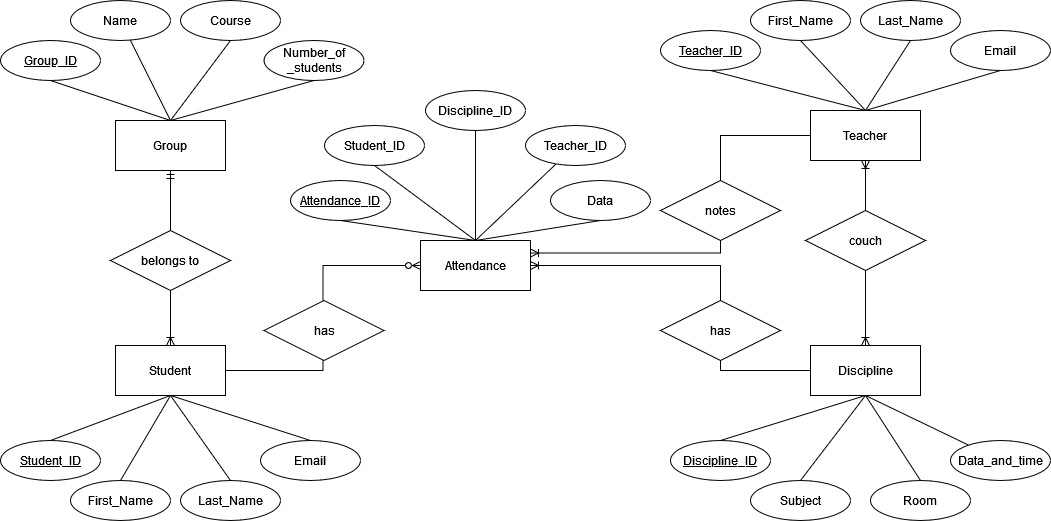


Рисунок 1 – ER-діаграма, побудована за нотацією Чена

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

Сутність Student перетворено в таблицю Student з первинним ключем student\_id та атрибутами first\_name, last\_name, email та зовнішнім ключем group\_id.

Сутність Group перетворено в таблицю Group з первинним ключем group\_id та атрибутами name, course, number\_of\_students.

Сутність Disciplinne перетворено в таблицю Discipline з первинним ключем discipline\_id та атрибутами subject, room, data\_and\_time.

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

Сутність Attendance перетворено в таблицю Attendence з первинним ключем attedence\_id та атрибутом data та зовнішнім ключем student\_id, discipline\_id, teacher\_id.

Оскільки в логічній моделі безпосередній зв’язок N:M є неможливим, а в концептуальній моделі він існує між сутностями Teacher i Discipline, то для його реалізації було створено таблицю Teacher\_Discipline, з первинним ключем tab\_id, та зовнішніми ключами teacher\_id i discipline\_id.

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

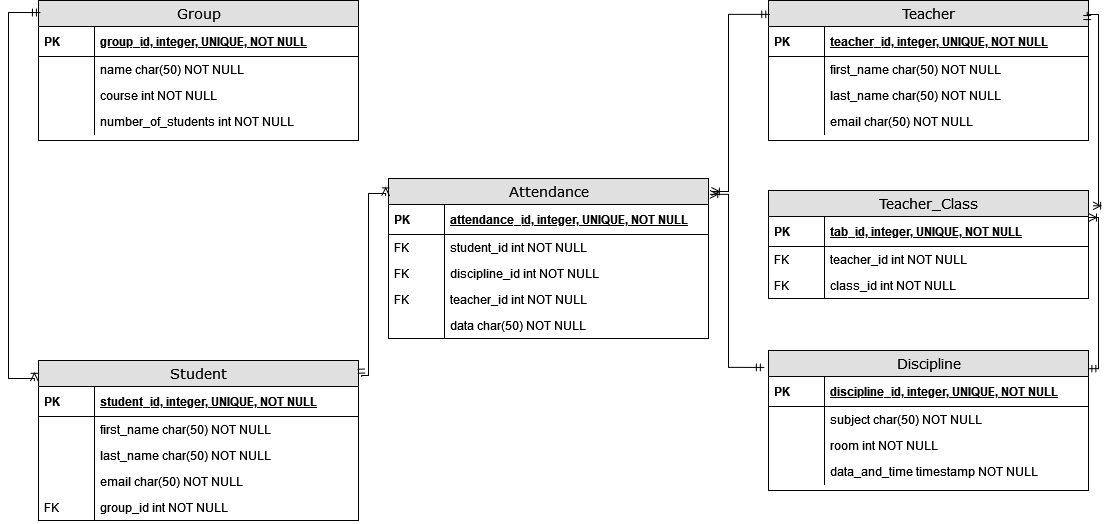
****

Рисунок 2 – Схема бази даних

Таблиця 1 ілюструє детальний перехід від однієї моделі до іншої.

Таблиця 1 – Опис об’єктів бази даних

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| *Student* – містить дані про студентів | *student\_id* – унікальний ідентифікатор студента | *integer* (число) |
| *first\_name* – ім’я студента | *character varying* (рядок) |
| *last\_name* – прізвище студента | *character varying* (рядок) |
| *email* – електронна скринька студента | *character varying* (рядок) |
| *group\_id* – група до якої належить студент | *integer* (число) |
| *Group* – містить дані про групи | *group\_id* – унікальний ідентифікатор групи | *integer* (число) |
| *name* – назва групи | *character varying* (рядок) |
| *course* – курс групи | *integer* (число) |
| *number\_of\_students* – кількість студентів у групі | *integer* (число) |
| *Attendance* – містить дані про відвідування занять студентами | *attedence\_id* – унікальний ідентифікатор запису про відвідуваність | *integer* (число) |
| *student\_id* – ідентифікатор студента | *integer* (число) |
| *discipline\_id* – ідентифікатор заняття | *integer* (число) |
| *teacher\_id –* ідентифікатор викладача | *integer* (число) |
| *data* – дата заняття | *data* (дата) |
| *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) |
| *Discipline* – містить дані про дисципліну | *discipline\_id* – унікальний ідентифікатор дисципліни | *integer* (число) |
| *subject* – предмет з якого відбувається заняття | *character varying* (рядок) |
| *room –* аудиторія в якій відбувається заняття | *integer* (число) |
| *data\_and\_time* – дата та час заняття | *timestamp* (дата та час) |
| *Teacher\_Discipline* – містить інформацію про відповідність викладача та дисципліни, яку він викладає | *tab\_id* – унікальний ідентифікатор відповідності | *integer* (число) |
| *teacher\_id* – ідентифікатор викладача | *integer* (число) |
| *discipline\_id* – ідентифікатор заняття | *integer* (число) |

**Функціональні залежності для кожної таблиці**

1. Student:

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

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

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

2. Group:

- group\_id -> { name, course, number\_of\_students }

3. Attendance:

- attendance\_ID -> { student\_id, discipline\_id, data}

- student\_id, discipline\_id, teacher\_id -> { attendance\_ID, data }

4. Teacher:

- teacher\_id -> { first\_name, last\_name, email }

5. Discipline:

- discipline\_id -> { subject, room, date\_and\_time }

- subject, room, date\_and\_time -> { discipline\_id }

6. Teacher\_Discipline:

- tab\_ID -> { teacher\_id, discipline\_id }

- teacher\_id -> { tab\_ID, discipline\_id }

- discipline\_id -> { tab\_ID, teacher\_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 не первинний атрибут має бути не транзитивно залежним від кожного ключа.

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

Висновок:

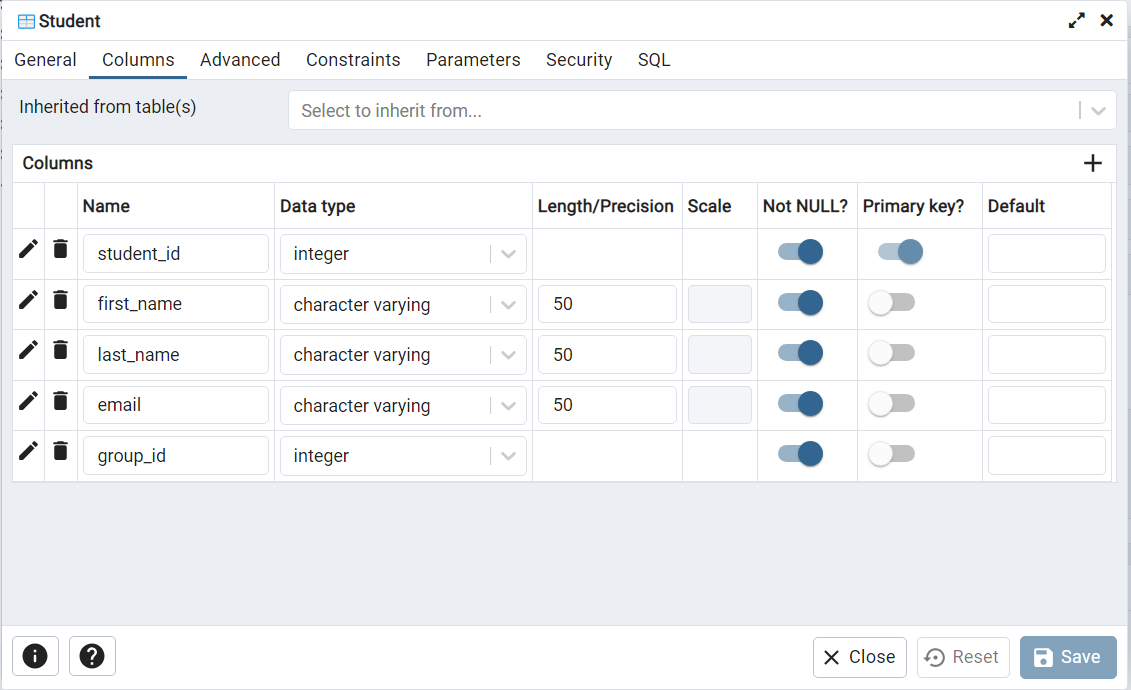
Схема бази даних відповідає нормальним формам НФ1, НФ2 та НФ3. Вона добре структурована і нормалізована, що сприяє ефективному та надійному зберіганню та обробці даних.

**Таблиці бази даних у pgAdmin4**

**A screenshot of a computer

Description automatically generated**

Student

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

Group

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

Discipline

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

Teacher

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

Attendance

A screenshot of a computer

Description automatically generated **A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Teacher\_Discipline

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**Таблиці в коді SQL**

-- Table: Attandance.Student

-- DROP TABLE IF EXISTS "Attandance"."Student";

CREATE TABLE IF NOT EXISTS "Attandance"."Student"

(

student\_id integer NOT NULL,

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

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

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

group\_id integer NOT NULL,

CONSTRAINT student\_id\_prkey PRIMARY KEY (student\_id),

CONSTRAINT group\_id\_frkey FOREIGN KEY (group\_id)

REFERENCES "Attandance"."Group" (group\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS "Attandance"."Student"

OWNER to postgres;

-- Table: Attandance.Group

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

CREATE TABLE IF NOT EXISTS "Attandance"."Group"

(

group\_id integer NOT NULL,

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

course integer NOT NULL,

number\_of\_students integer NOT NULL,

CONSTRAINT group\_id\_prkey PRIMARY KEY (group\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS "Attandance"."Group"

OWNER to postgres;

-- Table: Attandance.Discipline

-- DROP TABLE IF EXISTS "Attandance"."Discipline";

CREATE TABLE IF NOT EXISTS "Attandance"."Discipline"

(

discipline\_id integer NOT NULL,

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

room integer NOT NULL,

data\_and\_time timestamp with time zone NOT NULL,

CONSTRAINT discipline\_id\_prkey PRIMARY KEY (discipline\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS "Attandance"."Discipline"

OWNER to postgres;

-- Table: Attandance.Teacher

-- DROP TABLE IF EXISTS "Attandance"."Teacher";

CREATE TABLE IF NOT EXISTS "Attandance"."Teacher"

(

teacher\_id integer NOT NULL,

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

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

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

CONSTRAINT teacher\_id\_prkey PRIMARY KEY (teacher\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS "Attandance"."Teacher"

OWNER to postgres;

-- Table: Attandance.Teacher\_Discipline

-- DROP TABLE IF EXISTS "Attandance"."Teacher\_Discipline";

CREATE TABLE IF NOT EXISTS "Attandance"."Teacher\_Discipline"

(

tab\_id integer NOT NULL,

teacher\_id integer NOT NULL,

discipline\_id integer NOT NULL,

CONSTRAINT tab\_id\_prkey PRIMARY KEY (tab\_id),

CONSTRAINT discipline\_id\_frkey FOREIGN KEY (discipline\_id)

REFERENCES "Attandance"."Discipline" (discipline\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT teacher\_id\_frkey FOREIGN KEY (teacher\_id)

REFERENCES "Attandance"."Teacher" (teacher\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS "Attandance"."Teacher\_Discipline"

OWNER to postgres;

-- Table: Attandance.Attendance

-- DROP TABLE IF EXISTS "Attandance"."Attendance";

CREATE TABLE IF NOT EXISTS "Attandance"."Attendance"

(

attendance\_id integer NOT NULL,

student\_id integer NOT NULL,

discipline\_id integer NOT NULL,

teacher\_id integer NOT NULL,

data date NOT NULL,

CONSTRAINT attendance\_id\_prkey PRIMARY KEY (attendance\_id),

CONSTRAINT discipline\_id\_frkey FOREIGN KEY (discipline\_id)

REFERENCES "Attandance"."Group" (group\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT student\_id\_frkey FOREIGN KEY (student\_id)

REFERENCES "Attandance"."Student" (student\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS "Attandance"."Attendance"

OWNER to postgres;

**Висновок**

У ході виконання лабораторної роботи була розроблена база даних для системи відвідуваності занять студентами. База даних включає такі сутності: "Student", "Group", "Attendance", "Teacher", "Discipline" та "Teacher\_Discipline". Кожна з цих сутностей має відповідні атрибути, які дозволяють зберігати та керувати інформацією про студентів, групи, відвідування, викладачів та заняття.

ER-діаграма бази даних була підготовлена для візуального відображення взаємозв'язків між сутностями. Нотація Чена 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.

Крім того, була надана оновлена версія схеми бази даних, яка включає в себе таблицю "Teacher\_Discipline" для відображення взаємозв'язків між викладачами та заняттями.

На останньому етапі було надано копії екранів з pgAdmin4, що демонструють властивості стовпців та обмеження, а також вміст таблиць бази даних у PostgreSQL.

У цілому, розроблена база даних відповідає поставленим завданням та вимогам.