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-12

Левчук І. В.

Перевірив:

Павловский В. І.

Київ – 2023

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

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

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

# Модель «сутність-зв’язок» галузі Система обліку екологічних показників регіону

Обрана предметна галузь передбачає отримання і обробку потрібної інформації щодо системи обліку екологічних показників регіону

**Опис сутностей**

Згідно цієї області для побудови бази даних обраної області, були виділені такі сутності:

* Звіт (Report) з атрибутами: ідентифікатор звіту, дата заповнення, ідентифікатор регіону, ідентифікатор екологічного показника. Призначення: збереження даних щодо звітів.
* Регіон (Region) з атрибутами: ідентифікатор регіону, назва, площа. Призначення: збереження даних щодо регіонів.
* Екологічний Показник (Indicator) з атрибутами: ідентифікатор екологічного показника, назва, одиниці виміру. Призначення: збереження даних щодо занять.
* Спостерігач (Observer) з атрибутами: ідентифікатор Спостерігача, ім'я, Прізвище, Адреса. Призначення: збереження даних щодо викладачів.

**Опис зв’язків**

Зв’язок «Регіон» - «Звіт» є зв’язком 1:N. Один регіон може мати багато звітів, але кожен звіт належить лише одному регіону.

Зв’язок «Звіт» - «Екологічні показники» є зв’язком N:M. Кожен Звіт може містити багато Екологічних Показників, і кожен Показник може бути включений у багато Звітів.

Зв’язок «Регіон» - «Спостерігач» є зв’язком 1:N. Один регіон може мати багато спостерігачів, але кожен спостерігач належить лише одному р егіону.

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

Изображение выглядит как диаграмма, рисунок, зарисовка

Автоматически созданное описание

Рисунок 1 – ER-діаграма, побудована за нотацією Чена

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

Сутність Report перетворено в таблицю Report з первинним ключем report\_id та атрибутами submition\_date та зовнішніми ключами region\_id.

Сутність Region перетворено в таблицю Region з первинним ключем region\_id та атрибутами name, area.

Сутність Indicator перетворено в таблицю Indicator з первинним ключем indicator\_id та атрибутами name, Unit\_of\_Measurement.

Сутність Observer перетворено в таблицю Observer з первинним ключем observer\_id та атрибутами address, first\_name, last\_name.

Оскільки в логічній моделі безпосередній зв’язок N:M є неможливим, а в концептуальній моделі він існує між сутностями Indicator i Report, то для його реалізації було створено таблицю Indicator\_Report, з первинним ключем ind\_rep\_id, та зовнішніми ключами indicator\_id i report\_id.

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

Изображение выглядит как текст, снимок экрана, Шрифт, число

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 |
| *Report* – містить дані про звіти | report\_id – унікальний ідентифікатор звіту | *integer* (число) |
| region\_id – регіон щодо якого відноситься звіт | *integer* (число) |
| indicator\_id – ідентифікатор про який пишеться в звіті | *integer* (число) |
| submition\_date – електронна скринька студента | *date* (дата) |
| *Region* – містить дані про регіони | *Region\_id* – унікальний ідентифікатор регіону | *integer* (число) |
| *name* – назва регіону | *character varying* (рядок) |
| *area* – Площа регіону | *integer* (число) |
| *Indicator* – містить дані про екологічний індикатор | *indicator\_id* – унікальний ідентифікатор екологічного індикатора | *integer* (число) |
| *unit\_of\_measurement*– одиниці виміру індикатора | *character varying* (рядок) |
| *name* – назва індикатора | *character varying* (рядок) |
| *Observer* – містить дані про спостерігачів | *observer\_id* – унікальний ідентифікатор спостерігача | *integer* (число) |
| *first\_name* – ім’я спостерігача | *character varying* (рядок) |
| *last\_name* – прізвище спостерігача | *character varying* (рядок) |
| *adress* – адресса спостерігача | *character varying* (рядок) |
| *Teacher\_Discipline* – містить інформацію про відповідність викладача та дисципліни, яку він викладає | *Ibd\_rep\_id* – унікальний ідентифікатор відповідності | *integer* (число) |
| *observer \_id* – ідентифікатор спостерігача | *integer* (число) |
| report *id* – ідентифікатор звіту | *integer* (число) |

**Функціональні залежності для кожної таблиці**

1. Таблиця "Report":

* report\_id → submition\_date, region\_id

Ідентифікатор звіту визначає його дату подання та посилання на відповідний регіон.

1. Таблиця "Region":

* region\_id → name, area

Ідентифікатор регіону визначає його назву та площу.

1. Таблиця "Indicator":

* indicator\_id → name, unit\_of\_measurement

Ідентифікатор показника визначає його назву та одиниці виміру.

1. Таблиця "Observer":

* observer\_id → address, first\_name, last\_name

Ідентифікатор спостерігача визначає його адресу, ім'я та прізвище.

1. Таблиця "Indicator\_Report":

* ind\_rep\_id → indicator\_id, report\_id

Ідентифікатор запису в таблиці визначає його зв'язок з відповідними показниками та звітом.

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**

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

Автоматически созданное описание**

Indicator

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

Автоматически созданное описание**

Изображение выглядит как текст, линия, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Indicator\_Report

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

Автоматически созданное описание**

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

Автоматически созданное описание**

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

Автоматически созданное описание**

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

Автоматически созданное описание**

Observer

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

Автоматически созданное описание**

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

Автоматически созданное описание**

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

Автоматически созданное описание** **Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

Region

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

Автоматически созданное описание**

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

Автоматически созданное описание**

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

Автоматически созданное описание**

Report

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

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

Автоматически созданное описание**

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**Таблиці в коді SQL**

-- Table: public.Indicator

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

CREATE TABLE IF NOT EXISTS public."Indicator"

(

indicator\_id integer NOT NULL,

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

unit\_of\_measurement character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Indicator\_pkey" PRIMARY KEY (indicator\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Indicator"

OWNER to postgres;

-- Table: public.Indicator\_Report

-- DROP TABLE IF EXISTS public."Indicator\_Report";

CREATE TABLE IF NOT EXISTS public."Indicator\_Report"

(

ind\_rep\_id integer NOT NULL,

report\_id integer NOT NULL,

indicator\_id integer NOT NULL,

CONSTRAINT "Indicator\_Report\_pkey" PRIMARY KEY (ind\_rep\_id),

CONSTRAINT indicator\_id FOREIGN KEY (indicator\_id)

REFERENCES public."Indicator" (indicator\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

CONSTRAINT report\_id FOREIGN KEY (report\_id)

REFERENCES public."Report" (report\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Indicator\_Report"

OWNER to postgres;

-- Table: public.Observer

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

CREATE TABLE IF NOT EXISTS public."Observer"

(

observer\_id integer NOT NULL,

region\_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,

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

CONSTRAINT "Observer\_pkey" PRIMARY KEY (observer\_id),

CONSTRAINT "Observer\_region\_id\_fkey" FOREIGN KEY (region\_id)

REFERENCES public."Region" (region\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Observer"

OWNER to postgres;

-- Table: public.Region

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

CREATE TABLE IF NOT EXISTS public."Region"

(

region\_id integer NOT NULL,

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

area integer NOT NULL,

CONSTRAINT "Region\_pkey" PRIMARY KEY (region\_id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Region"

OWNER to postgres;

-- Table: public.Report

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

CREATE TABLE IF NOT EXISTS public."Report"

(

report\_id integer NOT NULL,

region\_id integer NOT NULL,

submition\_date date NOT NULL,

CONSTRAINT "Report\_pkey" PRIMARY KEY (report\_id),

CONSTRAINT "Report\_region\_id\_fkey" FOREIGN KEY (region\_id)

REFERENCES public."Region" (region\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Report"

OWNER to postgres;

**Висновок**

У ході виконання лабораторної роботи була розроблена база даних для системи відвідуваності занять студентами. База даних включає такі сутності: "Region", "Indicator", "Observer", "Report" та "Indicator\_Report". Кожна з цих сутностей має відповідні атрибути, які дозволяють зберігати та керувати інформацією про регіони, індикатори, наглядачами і звітами.

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 "Indicator\_Report" 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.

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