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:

студентк ІІI курсу

групи КВ-21

Ярич Р.Р.

Перевірив:

Петрашенко А. В.

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**

Система обліку страхових полісів та клієнтів – це набір програмних засобів і інструментів, що забезпечують ефективне управління даними щодо клієнтів та їхніх страхових полісів. Вона охоплює функції реєстрації та зберігання інформації про клієнтів, їхні страхові продукти, умови покриття, оплату премій та інші аспекти страхування. Система дозволяє стежити за історією змін в полісах, управляти термінами дії полісів, а також формувати звітність для моніторингу та аналізу роботи страхових компаній відповідно до вимог законодавства та внутрішніх політик.

**Опис сутностей**

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

1. Клієнт (Client)

Атрибути: ідентифікатор клієнта, ім’я, електронна пошта, баланс.

Призначення: збереження даних щодо клієнтів.

1. Invoice

Атрибути: ідентифікатор для інвойсу, ціна, час початку, час кінця.

Призначення: збереження даних щодо інвойсів.

1. Тип полісу (PolicyType)

Атрибути: тип полісу.

Призначення: збереження даних щодо типу полісу.

**Опис зв’язків між сутностями**

Зв’язок «Тип поліс» - «Invoice» є зв’язком 1:N. Сутність «Invoice» не може існувати без сутності «Тип поліс».

Зв’язок «Клієнт» - «Invoice» є зв’язком 1:0…N.

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

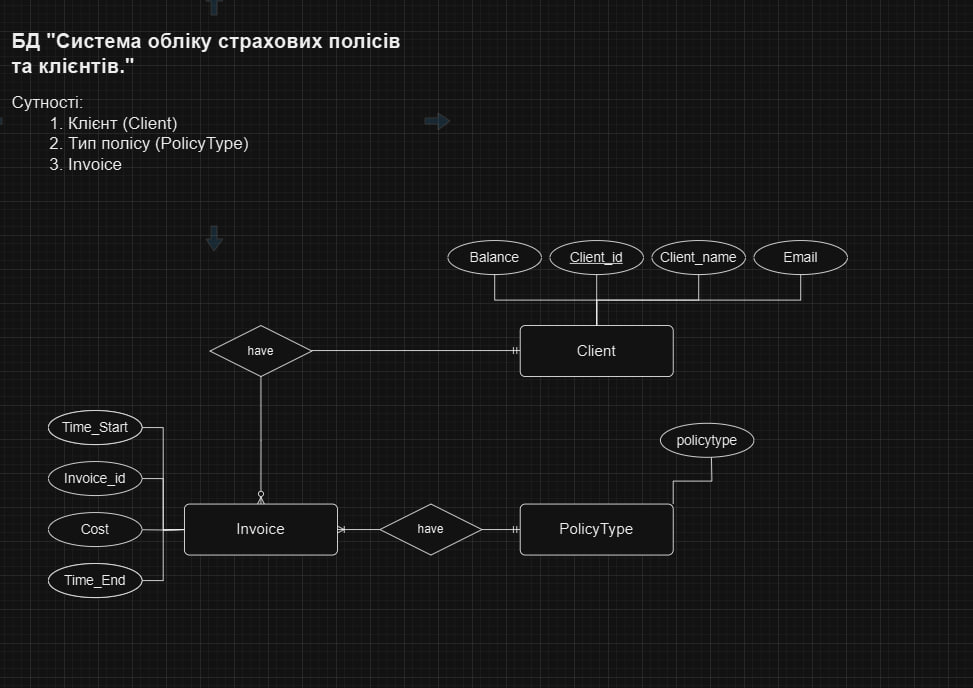


Рисунок 1 – ER-діаграма, побудована за нотацією Чена

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

Сутність Client перетворено в таблицю Clinet з первинним ключем client\_id та атрибутами client\_name, balance, email.

Сутність Invoice перетворено в таблицю Invoice з первинним ключем invoice\_id та атрибутами time\_start, time\_end, cost та зовнішній ключ client\_id та policytype.

Сутність PolicyType перетворено в таблицю PolicyTyoe з первинним ключем policytype

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

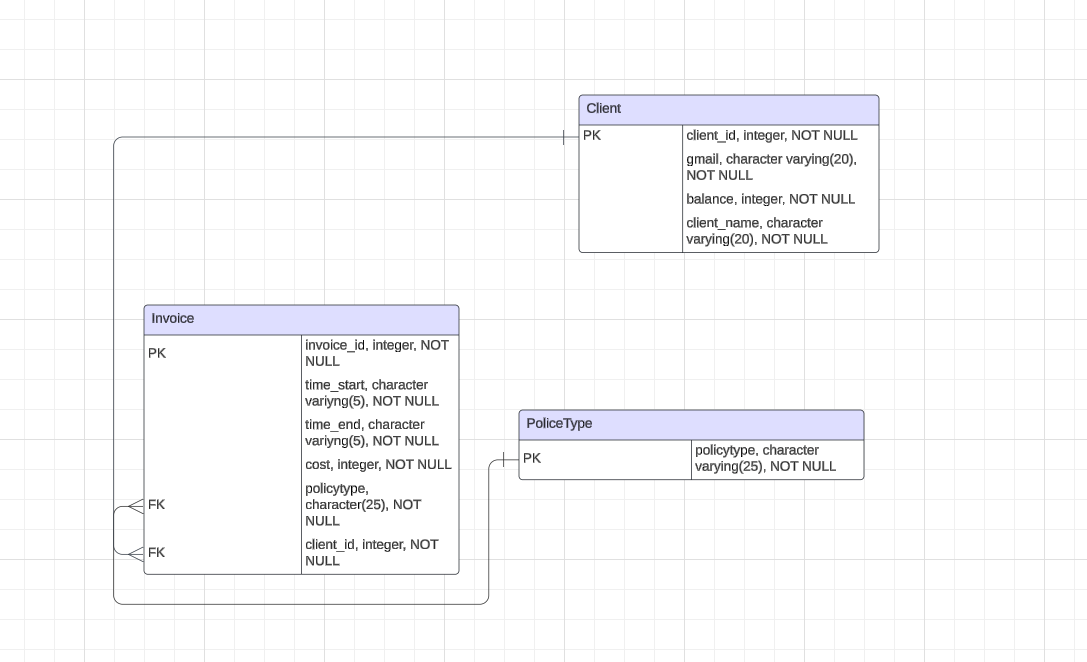


Рисунок 2 – Схема бази даних

Таблиця 1 ілюструє детальний перехід від однієї моделі до іншої.

Таблиця 1 – Опис об’єктів бази даних

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| *Client* – містить дані про клієнтів | *Client\_name* – ім’я клієнта | *character varying* (рядок) |
| *client\_id* – унікальний ідентифікатор клієнта | *integer* (число) |
| *email* – стать клієнта | *character varying* (рядок) |
| *Balance* – баланс | *integer* (число) |
| *PolicyType*– містить тип полісу | *policytype* – тип полісу | *character varying* (рядок) |
| *Invoice* – містить дані про invoice | *Invoice\_id – унікальний ідентифікатор інвойсу* | *integer* (число) |
| *Cost* – ціна | *integer* (число) |
| Time\_start– початок дії полісу | *character varying* (рядок) |
| Time\_end – кінець дії полісу | *character varying* (рядок) |

**Функціональні залежності для кожної таблиці**

**Таблиця "Client":**

1. **Client\_id → Client\_name, email, Balance**
   * Первинний ключ Client\_id визначає всі інші атрибути цієї сутності.

**Таблиця "Invoice":**

1. **invoice\_id → Time\_start, Time\_end, Cost, Client\_id, policytype**
   * Первинний ключ invoice\_id визначає всі інші атрибути цієї сутності.
   * Client\_id та policytype є зовнішніми ключами, що посилаються на таблиці "Client" і "PolicyType" відповідно.

**Таблиця "PolicyType":**

1. **policytype → invoice\_id**
   * policytype є первинним ключем цієї сутності, і визначає лише власний атрибут.

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:

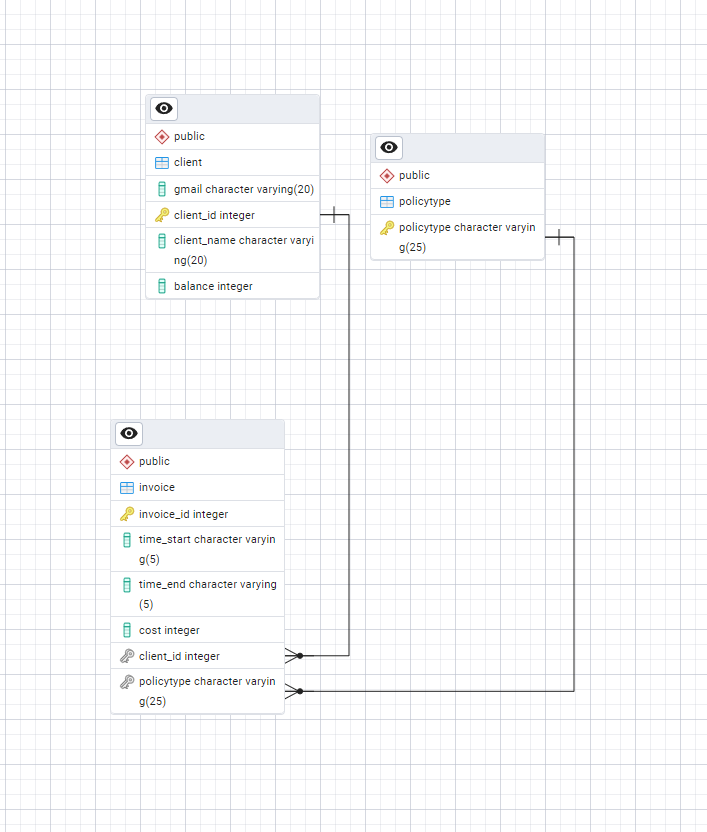
* Схема перебуває в 2НФ;
* Кожен неключовий атрибут функціонально залежить від усього ключа, а не його частини.

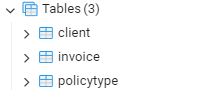
Оскільки дана схема в НФ2 та неключові атрибути не транзитивно залежать від інших неключових атрибутів, схема також в НФ3.

Висновок:

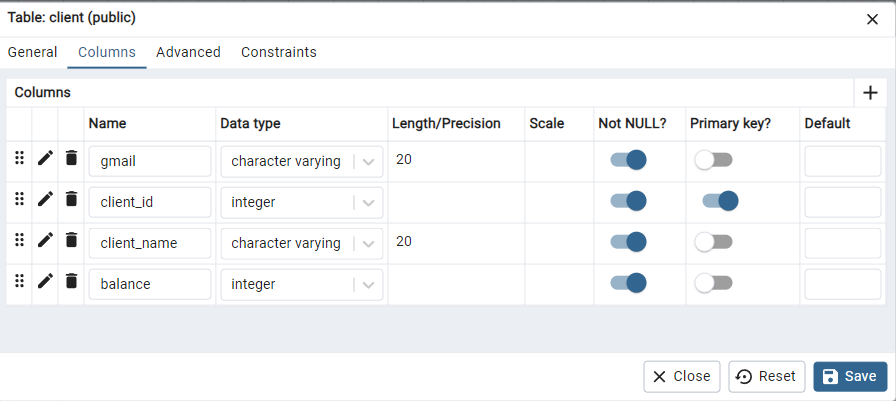
Схема бази даних відповідає нормальним формам НФ1, НФ2 та НФ3. Вона добре структурована і нормалізована, що сприяє ефективному та надійному зберіганню та обробці даних.

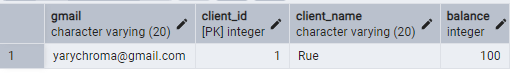
**Таблиці бази даних у pgAdmin4**

****

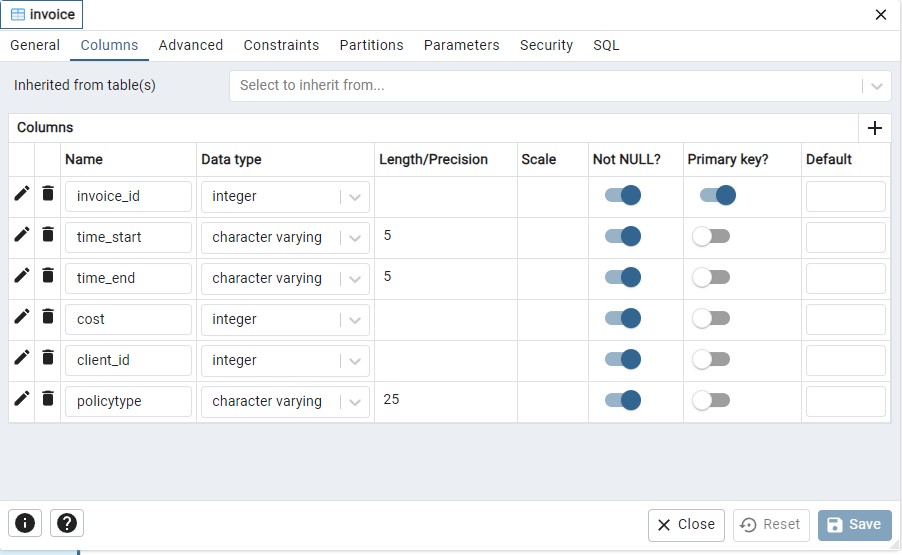
****

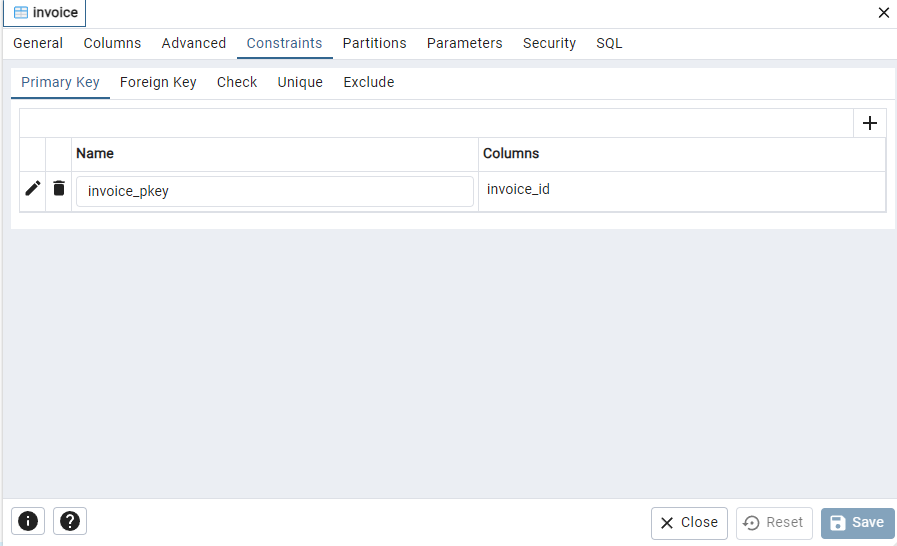
Client

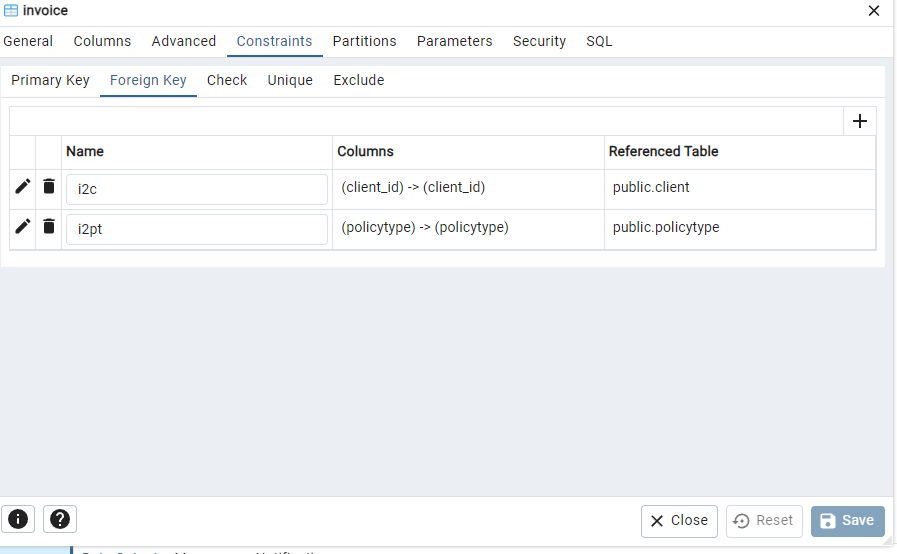


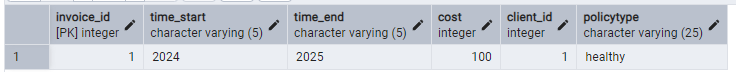


Invoice

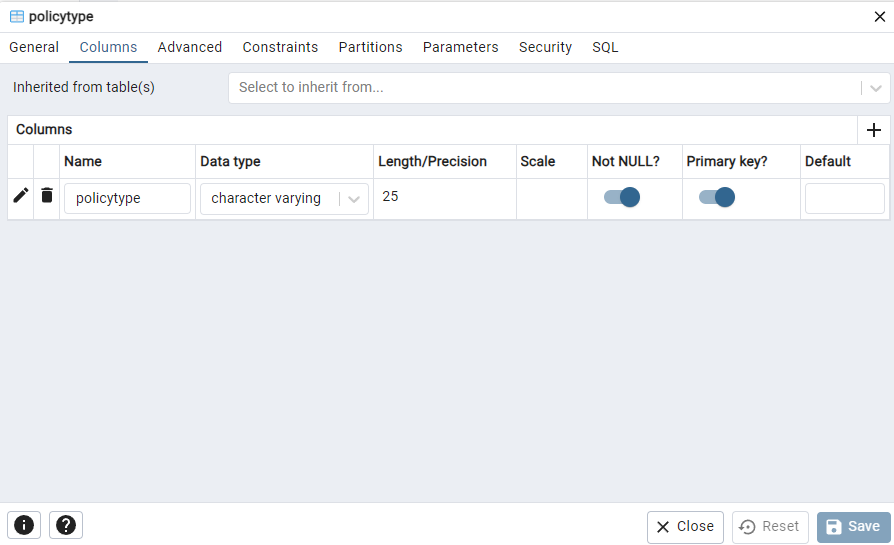
****

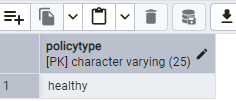






PolicyType

****

****

**Таблиці в коді SQL**

**-- This script was generated by the ERD tool in pgAdmin 4.**

**-- Please log an issue at https://github.com/pgadmin-org/pgadmin4/issues/new/choose if you find any bugs, including reproduction steps.**

**BEGIN;**

**CREATE TABLE IF NOT EXISTS public.client**

**(**

**gmail character varying(20) COLLATE pg\_catalog."default" NOT NULL,**

**client\_id integer NOT NULL,**

**client\_name character varying(20) COLLATE pg\_catalog."default" NOT NULL,**

**balance integer NOT NULL,**

**CONSTRAINT client\_pkey PRIMARY KEY (client\_id)**

**);**

**CREATE TABLE IF NOT EXISTS public.invoice**

**(**

**invoice\_id integer NOT NULL,**

**time\_start character varying(5) COLLATE pg\_catalog."default" NOT NULL,**

**time\_end character varying(5) COLLATE pg\_catalog."default" NOT NULL,**

**cost integer NOT NULL,**

**client\_id integer NOT NULL,**

**policytype character varying(25) COLLATE pg\_catalog."default" NOT NULL,**

**CONSTRAINT invoice\_pkey PRIMARY KEY (invoice\_id)**

**);**

**CREATE TABLE IF NOT EXISTS public.policytype**

**(**

**policytype character varying(25) COLLATE pg\_catalog."default" NOT NULL,**

**CONSTRAINT policytype\_pkey PRIMARY KEY (policytype)**

**);**

**ALTER TABLE IF EXISTS public.invoice**

**ADD CONSTRAINT i2c FOREIGN KEY (client\_id)**

**REFERENCES public.client (client\_id) MATCH SIMPLE**

**ON UPDATE NO ACTION**

**ON DELETE NO ACTION**

**NOT VALID;**

**ALTER TABLE IF EXISTS public.invoice**

**ADD CONSTRAINT i2pt FOREIGN KEY (policytype)**

**REFERENCES public.policytype (policytype) MATCH SIMPLE**

**ON UPDATE NO ACTION**

**ON DELETE NO ACTION**

**NOT VALID;**

**END;**

**Висновок**

Під час виконання лабораторної роботи була створена база даних для системи обліку відвідуваності студентами навчальних занять. Вона включає такі сутності: "Client", "Invoice", "PolicyType". Кожна сутність має необхідні атрибути для зберігання та управління інформацією про студентів, групи, відвідування, викладачів і заняття.

Для візуалізації взаємозв’язків між сутностями була розроблена ER-діаграма з використанням нотації Чена для позначення зв’язків і атрибутів.

Схема бази даних була проаналізована на відповідність нормальним формам (НФ1, НФ2, НФ3). Всі таблиці відповідають цим нормальним формам, що забезпечує надійне та ефективне зберігання даних.

Також було створено оновлену версію схеми бази даних із додаванням таблиці "Teacher\_Subject", яка відображає зв’язки між викладачами та заняттями.

На завершальному етапі були представлені знімки екранів із pgAdmin4, що демонструють властивості стовпців, обмеження та вміст таблиць бази даних у PostgreSQL.

Загалом, розроблена база даних повністю відповідає поставленим вимогам і завданням.

*Посилання на GitHub:*

[*https://github.com/Ethereal696/DB\_lab1*](https://github.com/Ethereal696/DB_lab1)