

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 студент групи: КВ-33

ПІБ: Малойван В. Р.

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 бази даних: Мобільний додаток для фітнес-трекінгу та здоров’я.

Коротка характеристика: Додаток призначений для відстеження фізичної активності користувачів, контролю стану здоров’я, а також для взаємодії з іншими користувачами у соціальному форматі. Система дозволяє реєструвати тренування, зберігати дані про вправи та показники здоров’я, підтримувати соціальні зв’язки.

**Розробка концептуальної моделі**

**При створенні даної бази даних виділено такі сутності:**

1. **Користувач (User)** – представляє клієнтів мобільного додатку:

* Атрибути: ім'я, прізвище, електронна пошта, дата реєстрації, телефон.

1. **Тренування (Workout)** – представляє заняття спортом, які виконує користувач:

* Атрибути: тип тренування (біг, силові вправи, йога тощо), дата, час.

1. **Відвідування тренувань (User\_Workout)** – відображає факт участі користувачів у тренуваннях:

* Атрибути: дата відвідування, час відвідування.

1. **Показники здоров’я (Health Metrics)** –представляє дані про фізичний стан користувача:

* Атрибути: кількість кроків, пульс, витрачені калорії, дата вимірювання.

**Зв’язки:**

Зв’язок «Користувач» - «Тренування»:

* Тип зв’язку: M до N (один користувач може брати участь у багатьох тренуваннях; одне тренування може виконуватись багатьма користувачами).
* Реалізується через проміжну сутність User\_Workout.

Зв’язок «Відвідування тренувань» - «Показники здоров’я»:

* Тип зв’язку: 1 до 1 (кожне відвідування має лише один набір показників; кожен набір показників відноситься лише до одного відвідування).

Зв’язок «Користувач» - «Користувач»:

* Тип зв’язку: 1 до N (кожен користувач може мати багато друзів, і кожен інший користувач також може бути другом багатьох).
* Атрибути зв’язку: статус (підтверджено/очікує), дата (дата встановлення дружби).

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

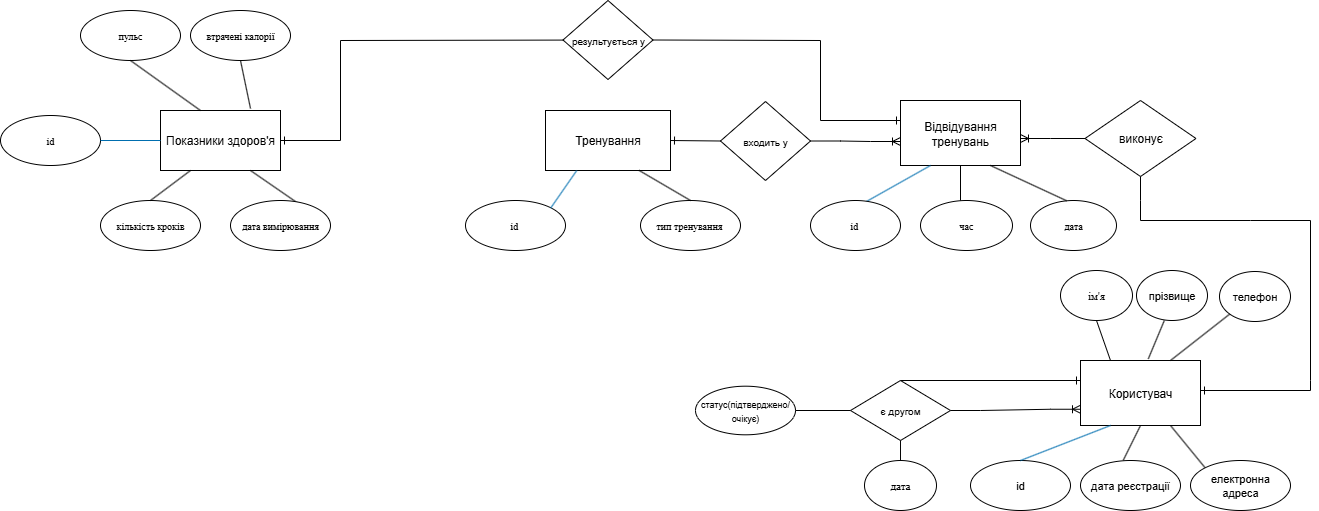


Рисунок 1 – Графічне подання концептуальної моделі «Сутність-зв’язок»

***Завдання №2:***

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

Сутність “Користувач” перетворено в таблицю User з первинним ключем user\_id.

Атрибути таблиці: first\_name (ім’я), last\_name (прізвище), email (електронна пошта), phone (номер телефону), date\_registration (дата реєстрації користувача).

У таблиці User зовнішніх ключів не визначено.

Сутність “Тренування” перетворено в таблицю Workout з первинним ключем workout\_id.

Атрибути таблиці: type\_workout (тип тренування).

У таблиці зовнішніх ключів не визначено.

Сутність “Відвідування тренувань” перетворено в таблицю User\_Workout(з первинним ключем user\_workout\_id), яка реалізує зв’язок M:N між користувачами та тренуваннями.

Атрибути таблиці: date, time.

У таблиці User\_Workout зовнішні ключі user\_user\_id та workout\_workout\_id відображають належність відвідування конкретному користувачу та тренуванню, забезпечуючи реалізацію зв’язку M:N “Користувач – Тренування”.

Сутність “Показники здоров’я” перетворено в таблицю Health Metrics з первинним ключем metrics\_id.

Атрибути таблиці: steps (кількість кроків), pulse (пульс), calories (витрачені калорії), measurement\_date (дата вимірювання).

У таблиці Health Metrics зовнішній ключ user\_workout\_id забезпечує зв’язок 1:1 із сутністю User\_Workout, що дозволяє фіксувати показники здоров’я у контексті конкретного відвідування тренування.

Зв’язок “Дружба” реалізовано через таблицю Friendship, яка моделює соціальні відносини між користувачами.

У таблиці використовується складений первинний ключ (user\_id1, user\_id2), що ідентифікує пару користувачів.

Атрибути таблиці: status (стан дружби: підтверджено/очікує), date (дата встановлення дружби).

Обидва зовнішні ключі user\_id1 та user\_id2 посилаються на User (user\_id), забезпечуючи реалізацію унарного зв’язку 1:N “Користувач – Користувач”.

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

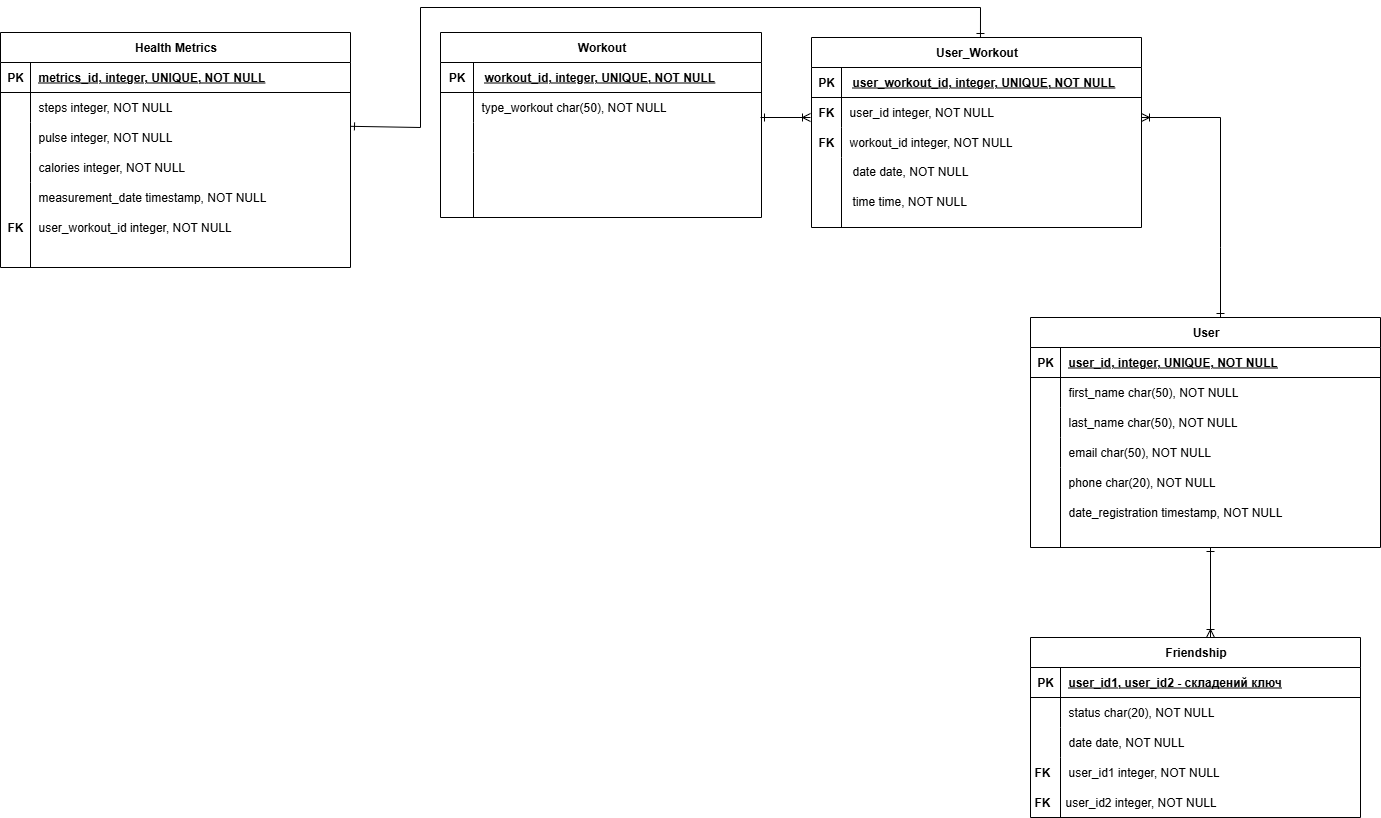


Рисунок 2 – Графічне подання логічної моделі «Сутність-зв’язок»

Таблиця 1 ілюструє детальний перехід від однієї моделі до іншої.

Таблиця 1 – Опис об’єктів бази даних.

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| User – містить дані про користувача | *user\_id* – унікальний ідентифікатор користувача | integer (числовий) |
| *first\_name* – ім’я користувача | character varying (рядок) |
| *last\_name* – прізвище користувача | character varying (рядок) |
| *email –* електронна пошта користувача | character varying (рядок) |
| *phone –* номер телефону користувача | character varying (рядок) |
| *date\_registration –* дата реєстрації користувача | timestamp (дата та час) |
| Workout– містить дані про тренування | *workout\_id* – унікальний ідентифікатор тренування | integer (числовий) |
| *type\_workout* – тип тренування | character varying (рядок) |
| User\_Workout– містить дані про участь користувачів у тренуваннях (відвідування) | *user\_workout\_id* – унікальний ідентифікатор відвідування | integer (числовий) |
| *user\_user\_id* – ідентифікатор користувача | integer (зовнішній ключ до Користувача) |
| *workout\_workout\_id* – ідентифікатор тренування | integer (зовнішній ключ до Тренування) |
| *date –* дата відвідування тренування | date (дата) |
| *time –*  час відвідування тренування | time(час) |
| Health Metrics– містить дані про фізичний стан користувача | *metrics\_id* – унікальний ідентифікатор вимірювання | integer (числовий) |
| *steps* – кількість кроків | integer (числовий) |
| *pulse* – пульс користувача | integer (числовий) |
| *calories* – витрачені калорії | integer (числовий) |
| *measurement\_date* – дата вимірювання | timestamp (дата та час) |
| *user\_ workout\_id* – ідентифікатор відвідування тренування | integer (зовнішній ключ до Відвідування тренувань) |

|  |  |  |
| --- | --- | --- |
| Зв’язок | Атрибут | Тип атрибуту |
| Friendship – містить дані про дружбу | *(user\_id1, user\_id2)* – унікальний ідентифікатор дружби(складений первинний ключ) | integer (числовий) |
| *user\_id1* – ідентифікатор першого користувача | integer (зовнішній ключ до Користувача) |
| *user\_id2* – ідентифікатор другого користувача | integer (зовнішній ключ до Користувача) |
| *date –* дата, коли користувачі стали друзями | date (дата) |
| *status* – статус дружби (підтверджено/очікує) | character varying (рядок) |

**Функціональні залежності для кожної таблиці**

1. User:

*user\_id* → first\_name, last\_name, email, phone, date\_registration

2. Workout:

*workout\_id* → type\_workout

3. User\_Workout:

*user\_workout\_id* → user\_user\_id, workout\_workout\_id, date, time

4. Health Metrics:

*metrics\_id* → steps, pulse, calories, measurement\_date, user\_workout\_id

5. Friendship:

*(user\_id1, user\_id2)* → status, date

Функціональні залежності відображають співвідношення між ключовими та неключовими атрибутами в таблицях бази даних. Вони визначають, які саме поля однозначно встановлюються на основі значень ключа. Правильне визначення таких залежностей є необхідною умовою для нормалізації схеми, оскільки дозволяє уникнути дублювання даних, зменшує ризик аномалій під час оновлення та забезпечує цілісність інформації.

Транзитивна залежність виникає у випадку, коли один атрибут визначає інший не безпосередньо, а через проміжний атрибут (наприклад, якщо A → B і B → C, то маємо A → C).

У розробленій моделі всі атрибути напряму залежать від своїх ключів:

* + у User – від user\_id;
  + у Workout – від workout\_id;
  + у User\_Workout – від складеної комбінації (user\_workout\_id);
  + у Health Metrics – від metrics\_id;
  + у Friendship – від складеної комбінації (user\_id1, user\_id2).

Жодного випадку транзитивної залежності немає. Це означає, що структура не створює надмірності даних і відповідає вимогам нормалізації.

**Відповідність нормальним формам**

*Перша нормальна форма (1НФ)*

Усі атрибути мають атомарний характер (одне поле *–* одне значення).

Кожен рядок унікальний завдяки первинним ключам (user\_id, workout\_id, metrics\_id, user\_workout\_id) або складеному ключу (user\_id1, user\_id2) у таблиці Friendship.

Отже, схема задовольняє умови 1НФ.

*Друга нормальна форма (2НФ)*

Модель перебуває в 1НФ. Усі неключові атрибути повністю залежать від первинних ключів:

* + у таблицях User, Workout і Health Metrics кожен атрибут визначається відповідним ключем повністю;
  + у таблиці Friendship атрибути status і date визначаються лише повною комбінацією (user\_id1, user\_id2). Часткових залежностей немає.
  + у таблиці User\_Workout усі атрибути (user\_user\_id, workout\_workout\_id, date, time) залежать від первинного ключа user\_workout\_id. Часткових залежностей немає.

Таким чином, схема відповідає умовам 2НФ.

*Третя нормальна форма (3НФ)*

Модель перебуває в 2НФ:

* + жоден з неключових атрибутів не визначає інший, а всі вони безпосередньо залежать від первинного ключа своєї таблиці.
  + у таблиці Health Metrics атрибути steps, pulse, calories, measurement\_date залежать від metrics\_id, а не від зовнішнього ключа user\_workout\_id, що виключає транзитивні залежності;
  + у таблиці Friendship та User\_Workout також немає транзитивних залежностей.

Отже, схема відповідає умовам 3НФ.

**Висновок**

Розроблена модель бази даних відповідає вимогам 1НФ, 2НФ та 3НФ:

* + забезпечує атомарність і унікальність записів (1НФ);
  + виключає часткові залежності та повністю підпорядковує атрибути первинним ключам (2НФ);
  + не містить транзитивних залежностей, що гарантує відсутність надмірності даних (3НФ).

Така структура є академічно правильною, логічно узгодженою та стійкою до аномалій при вставці, оновленні й видаленні даних. Структура придатна до масштабування та подальшого розвитку без ризику втрати цілісності.

Схема бази даних у pgAdmin 4 зображено на рисунку 3.

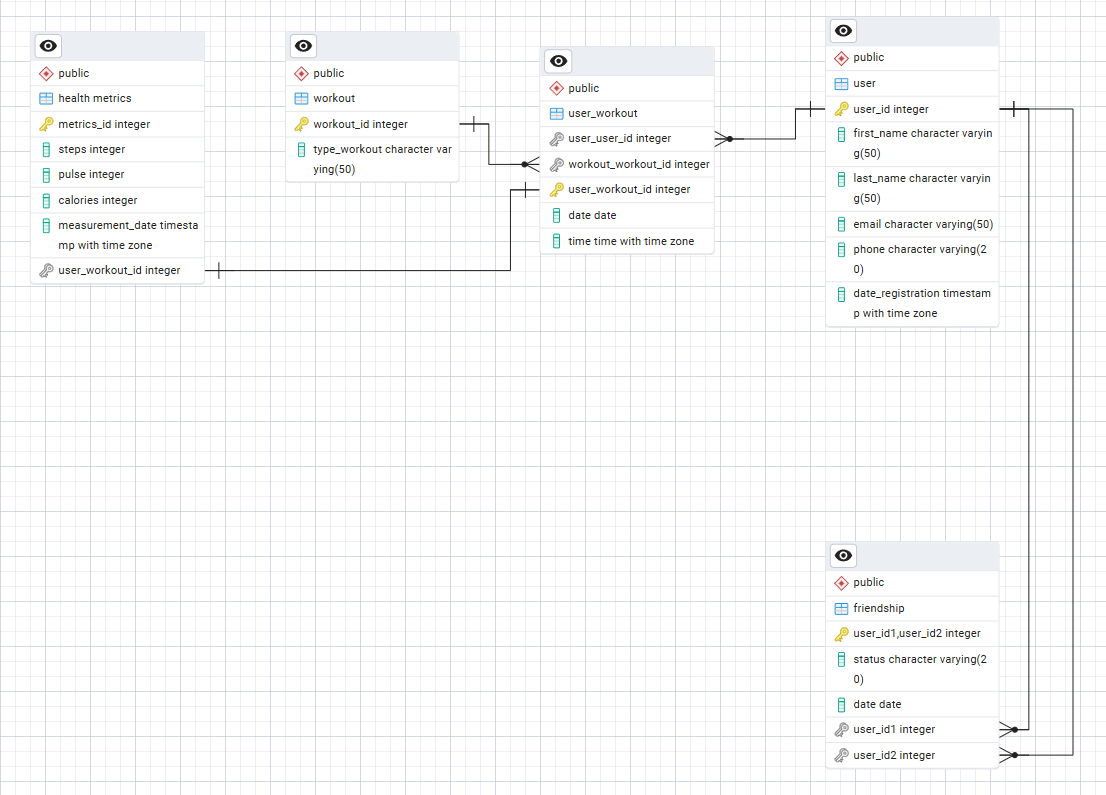
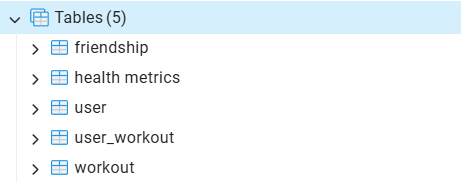
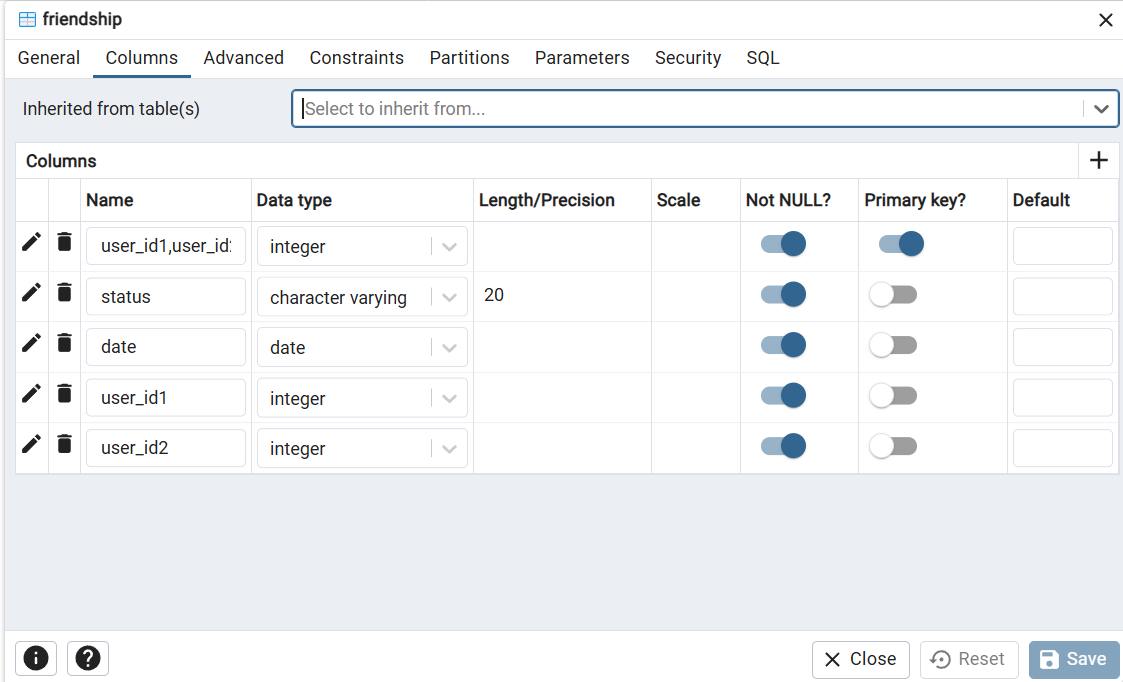


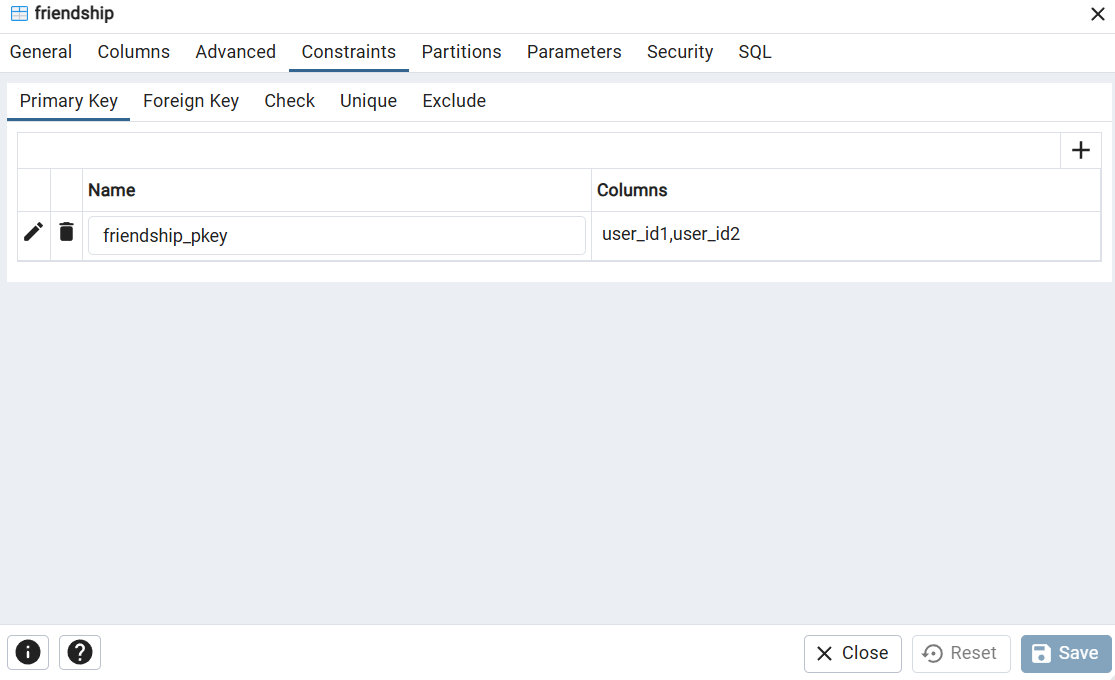
Рисунок 3 *–* Схема бази даних у pgAdmin 4

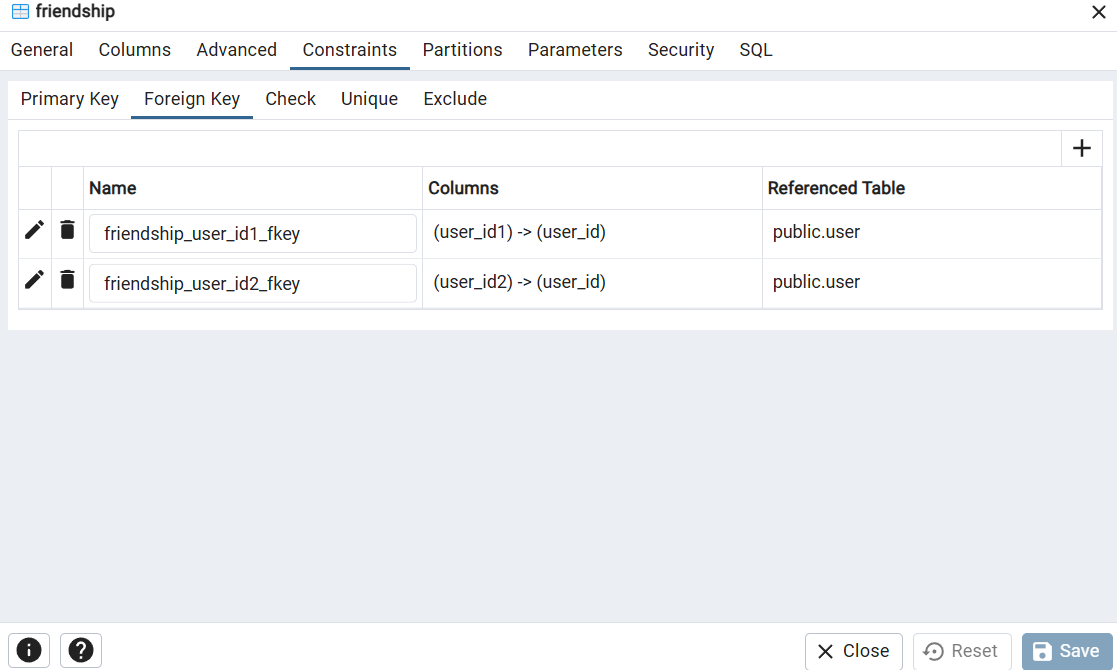
**Таблиці бази даних у pgAdmin4**

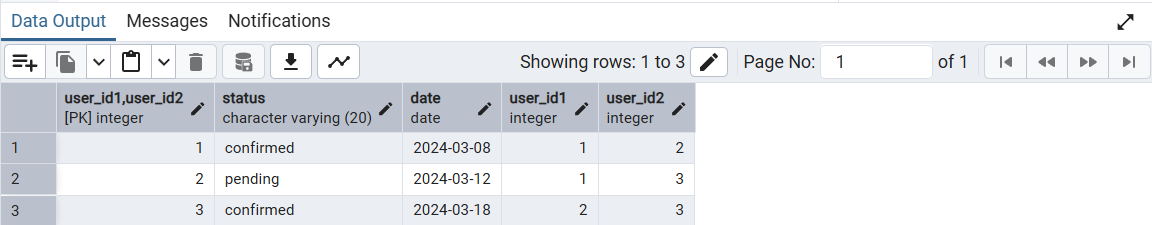


Friendship

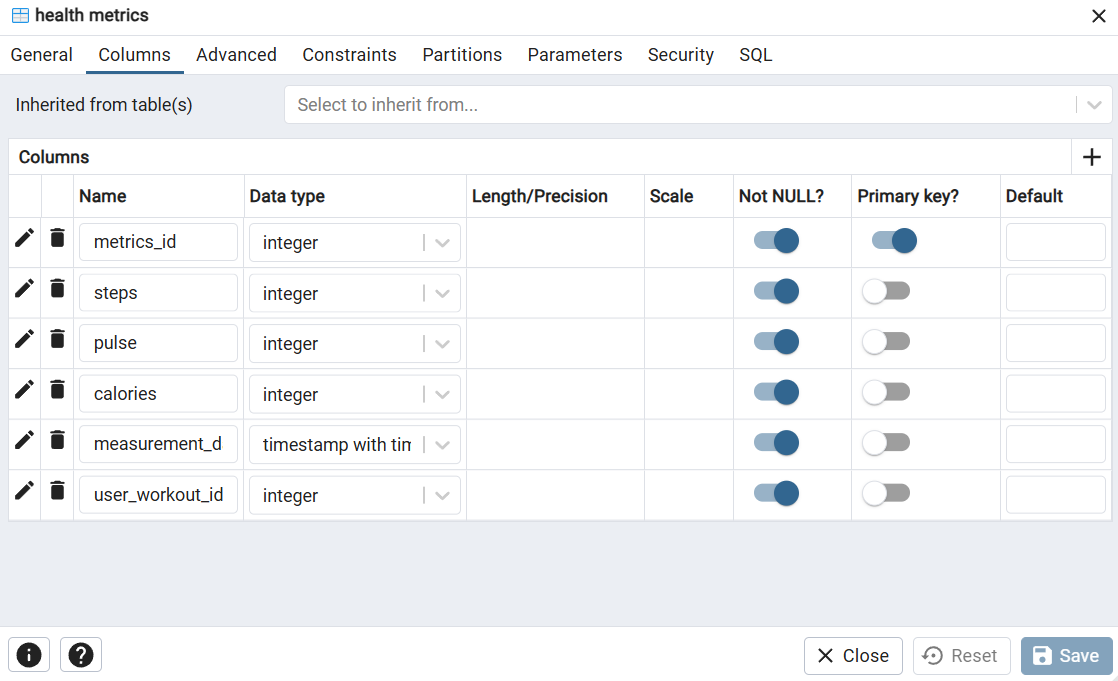


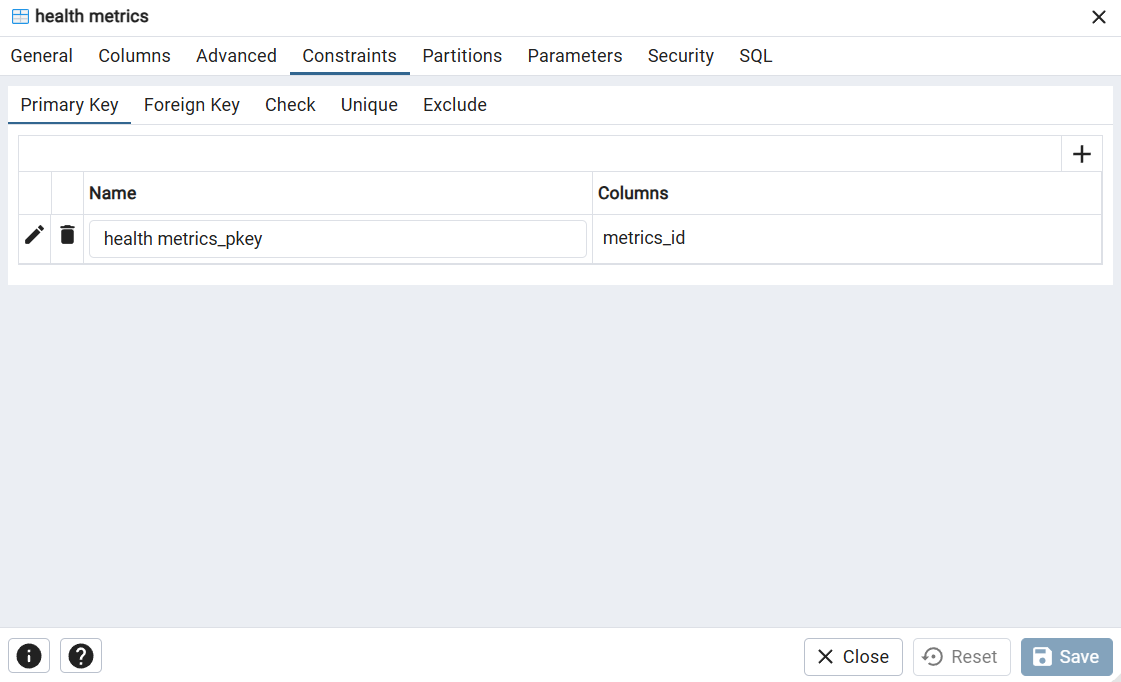


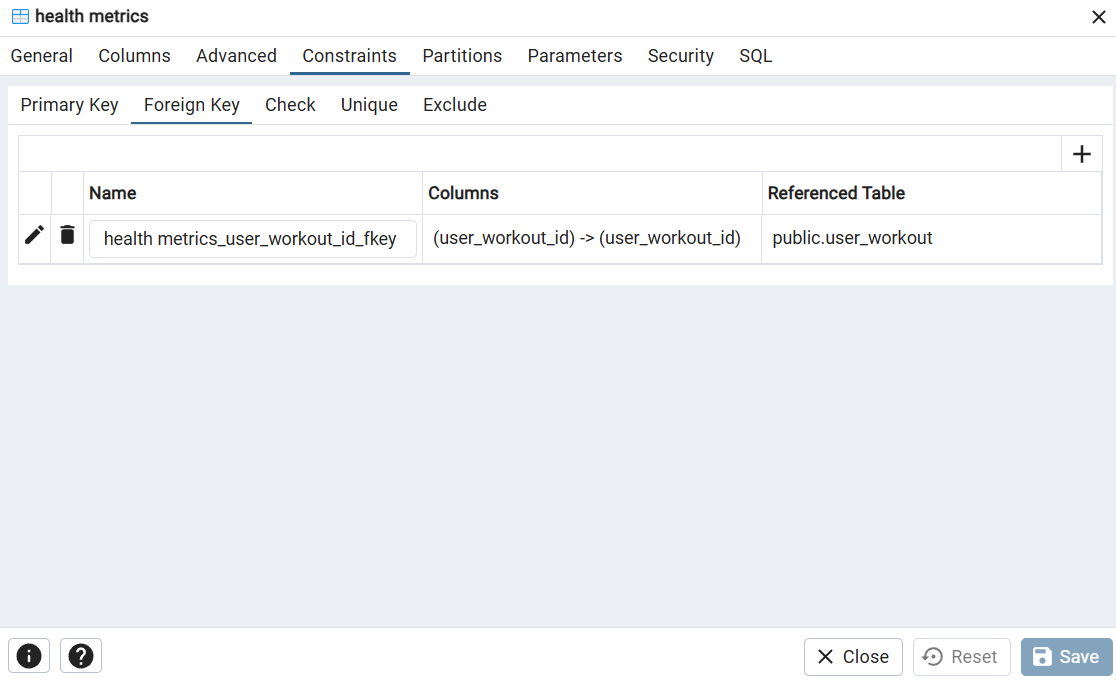


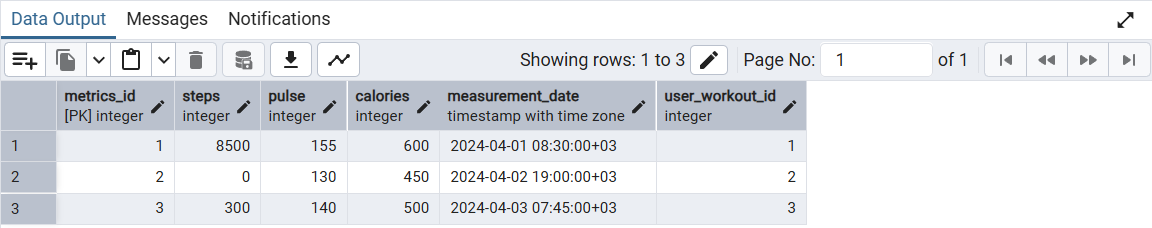


Health Metrics

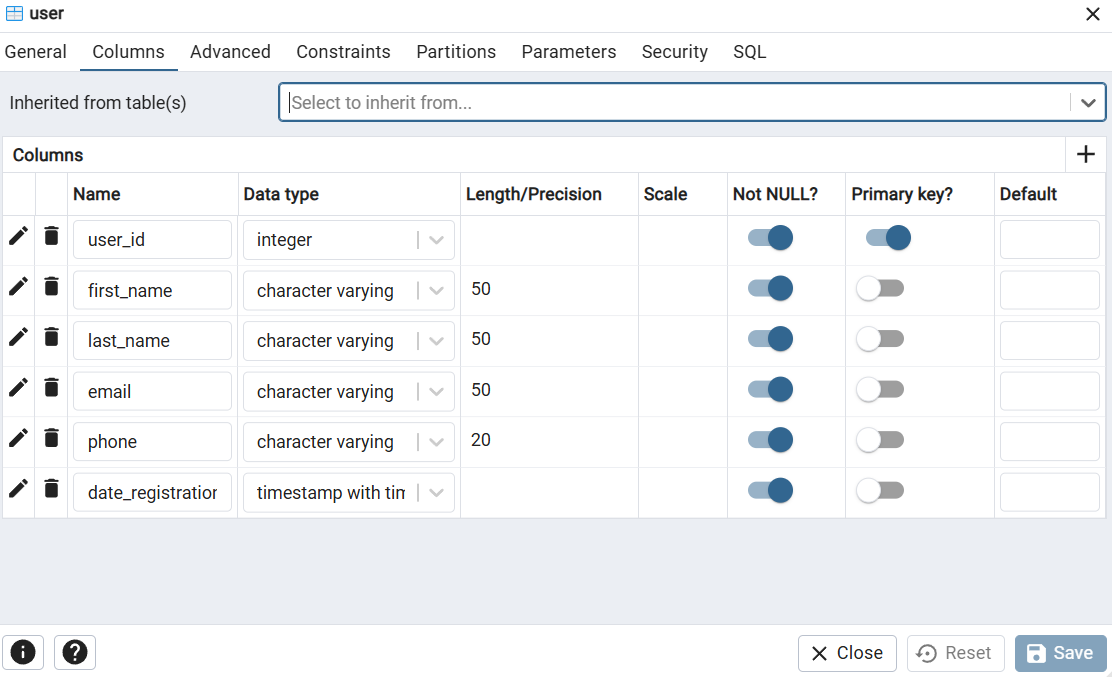


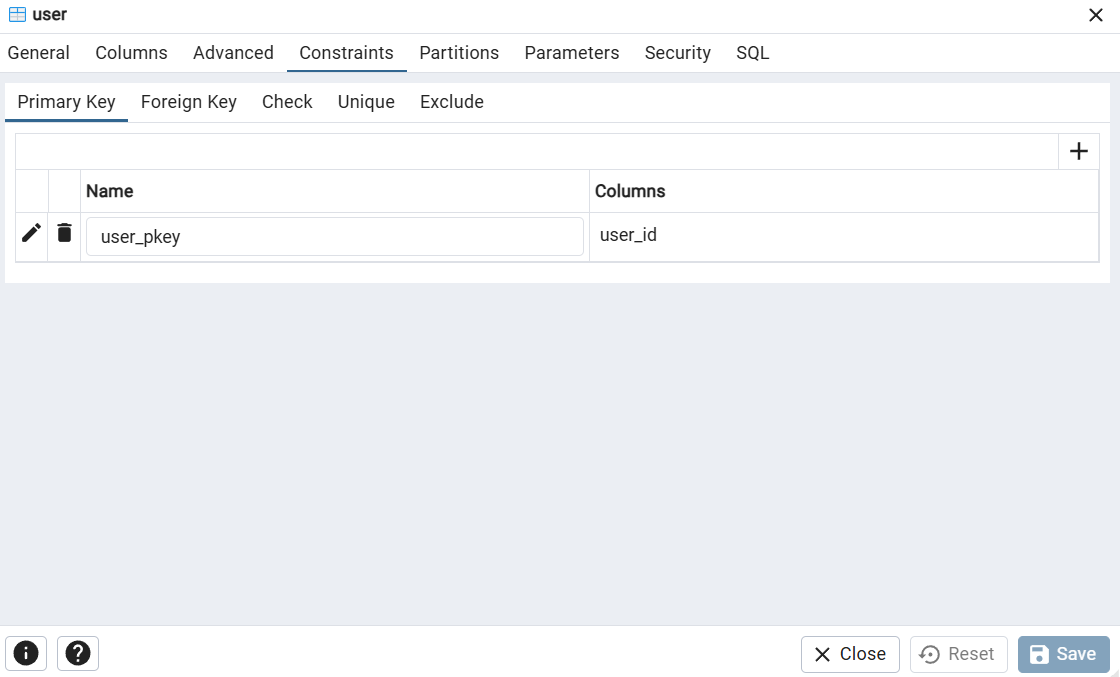




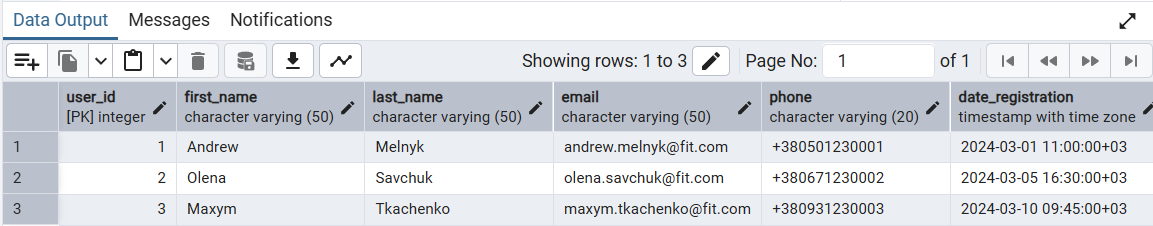


User

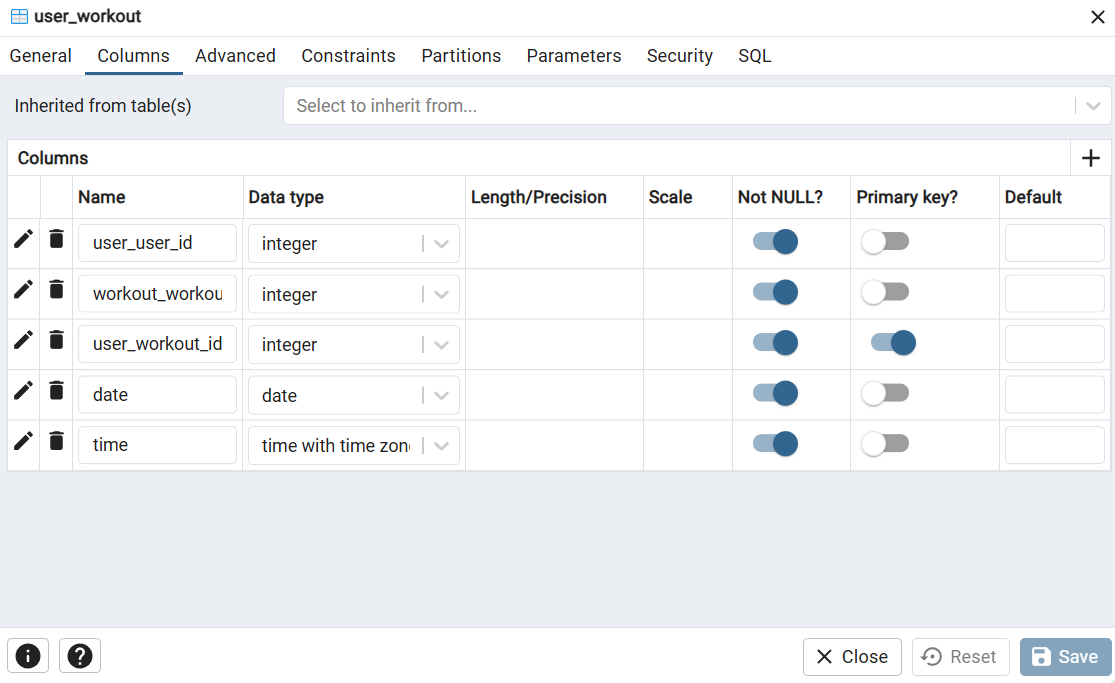


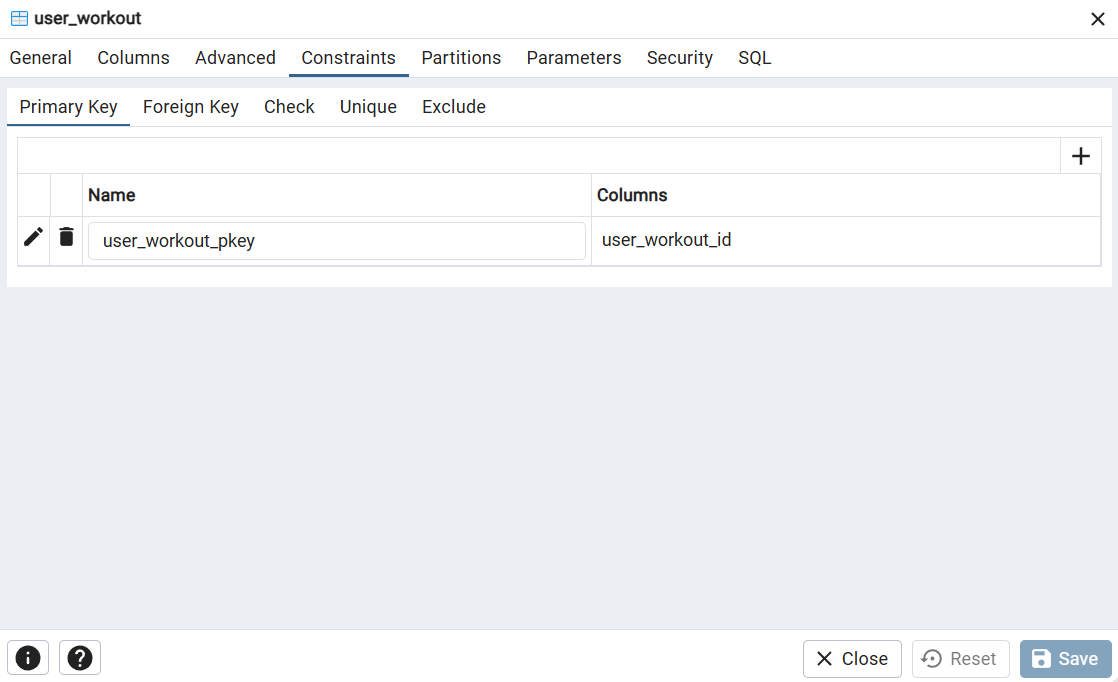


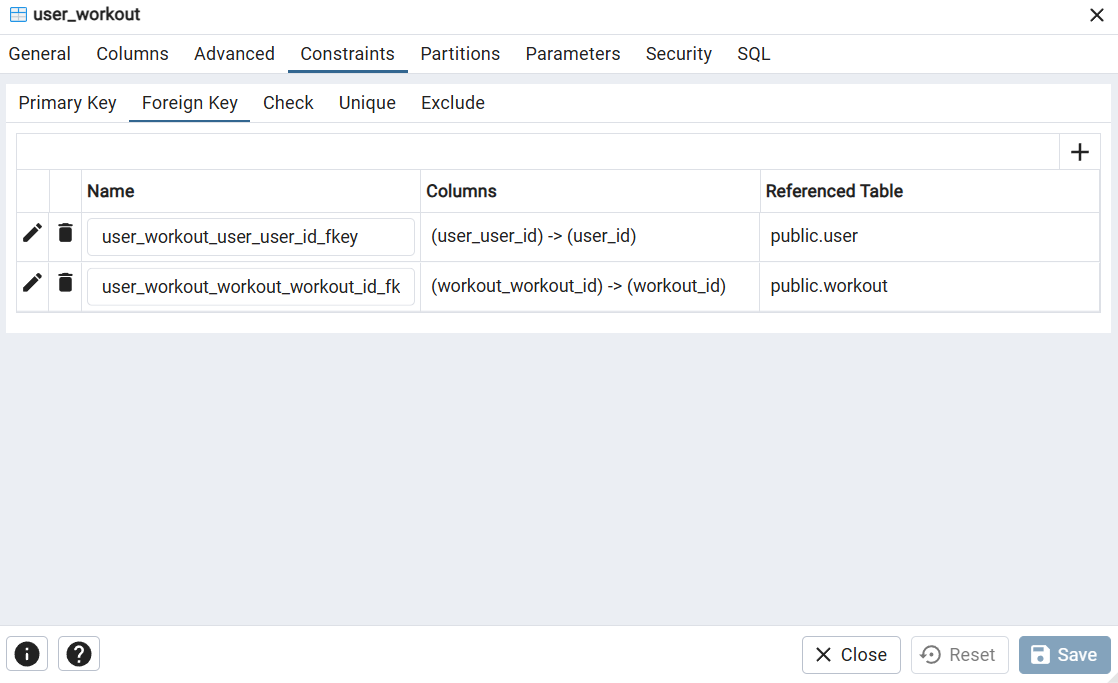
У таблиці User немає зовнішніх ключів(FK).

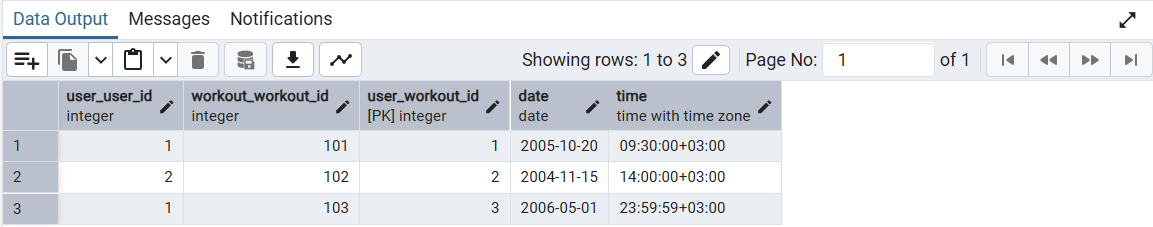


User\_Workout

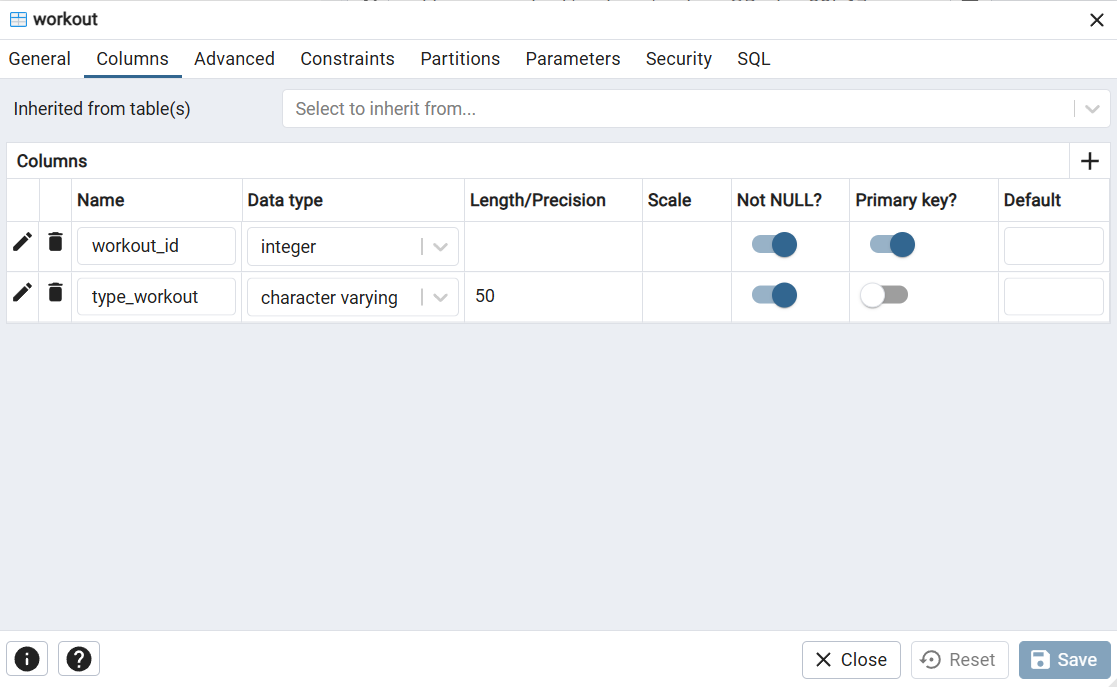


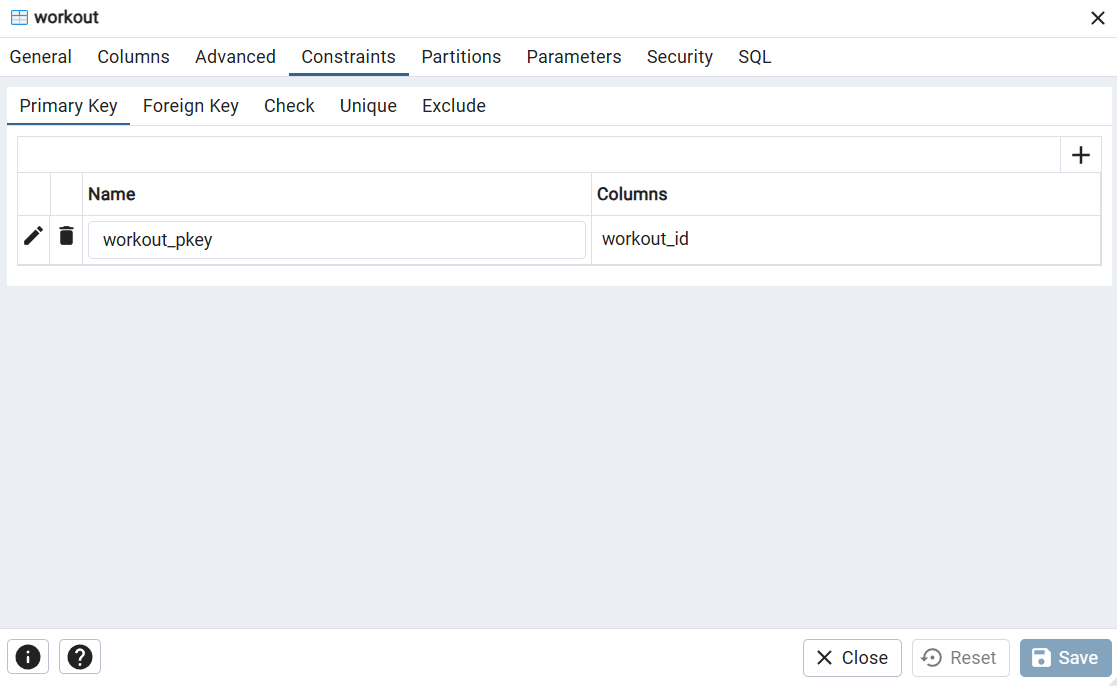




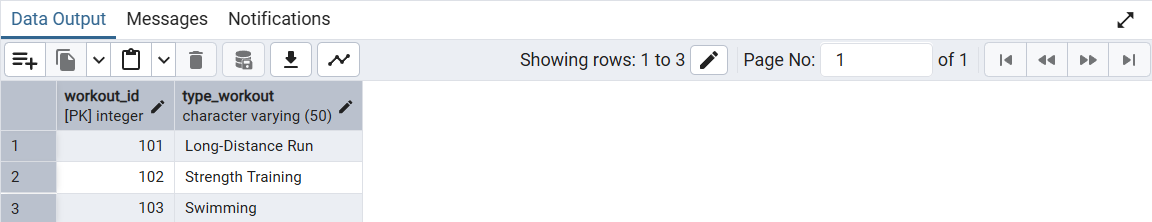


Workout





У таблиці Workout немає зовнішніх ключів(FK).



**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."user"

(

user\_id integer NOT NULL,

first\_name character varying(50) NOT NULL,

last\_name character varying(50) NOT NULL,

email character varying(50) NOT NULL,

phone character varying(20) NOT NULL,

date\_registration timestamp with time zone NOT NULL,

PRIMARY KEY (user\_id)

);

CREATE TABLE IF NOT EXISTS public.friendship

(

"user\_id1,user\_id2" integer NOT NULL,

status character varying(20) NOT NULL,

date date NOT NULL,

user\_id1 integer NOT NULL,

user\_id2 integer NOT NULL,

PRIMARY KEY ("user\_id1,user\_id2")

);

CREATE TABLE IF NOT EXISTS public.workout

(

workout\_id integer NOT NULL,

type\_workout character varying(50) NOT NULL,

PRIMARY KEY (workout\_id)

);

CREATE TABLE IF NOT EXISTS public."health metrics"

(

metrics\_id integer NOT NULL,

steps integer NOT NULL,

pulse integer NOT NULL,

calories integer NOT NULL,

measurement\_date timestamp with time zone NOT NULL,

user\_workout\_id integer NOT NULL,

PRIMARY KEY (metrics\_id),

UNIQUE (user\_workout\_id)

);

CREATE TABLE IF NOT EXISTS public.user\_workout

(

user\_user\_id integer NOT NULL,

workout\_workout\_id integer NOT NULL,

user\_workout\_id integer NOT NULL,

date date NOT NULL,

"time" time with time zone NOT NULL,

PRIMARY KEY (user\_workout\_id)

);

ALTER TABLE IF EXISTS public.friendship

ADD FOREIGN KEY (user\_id1)

REFERENCES public."user" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public.friendship

ADD FOREIGN KEY (user\_id2)

REFERENCES public."user" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."health metrics"

ADD FOREIGN KEY (user\_workout\_id)

REFERENCES public.user\_workout (user\_workout\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public.user\_workout

ADD FOREIGN KEY (user\_user\_id)

REFERENCES public."user" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public.user\_workout

ADD FOREIGN KEY (workout\_workout\_id)

REFERENCES public.workout (workout\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

END;

**Посилання на GitHub:** https://github.com/MaloivanVladyslav/vladlab\_project