

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №3**

з дисципліни

**«Бази даних і засоби управління»**

Тема: «**Засоби оптимізації роботи СУБД PostgreSQL**»

Виконав: студент 3 курсу

ФПМ групи КВ-83

Хаустович Олександр

Перевірив: Павловський В.І.

Київ – 2020

**Ознайомлення з базовими операціями СУБД PostgreSQL**

***Метою роботи*** є здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

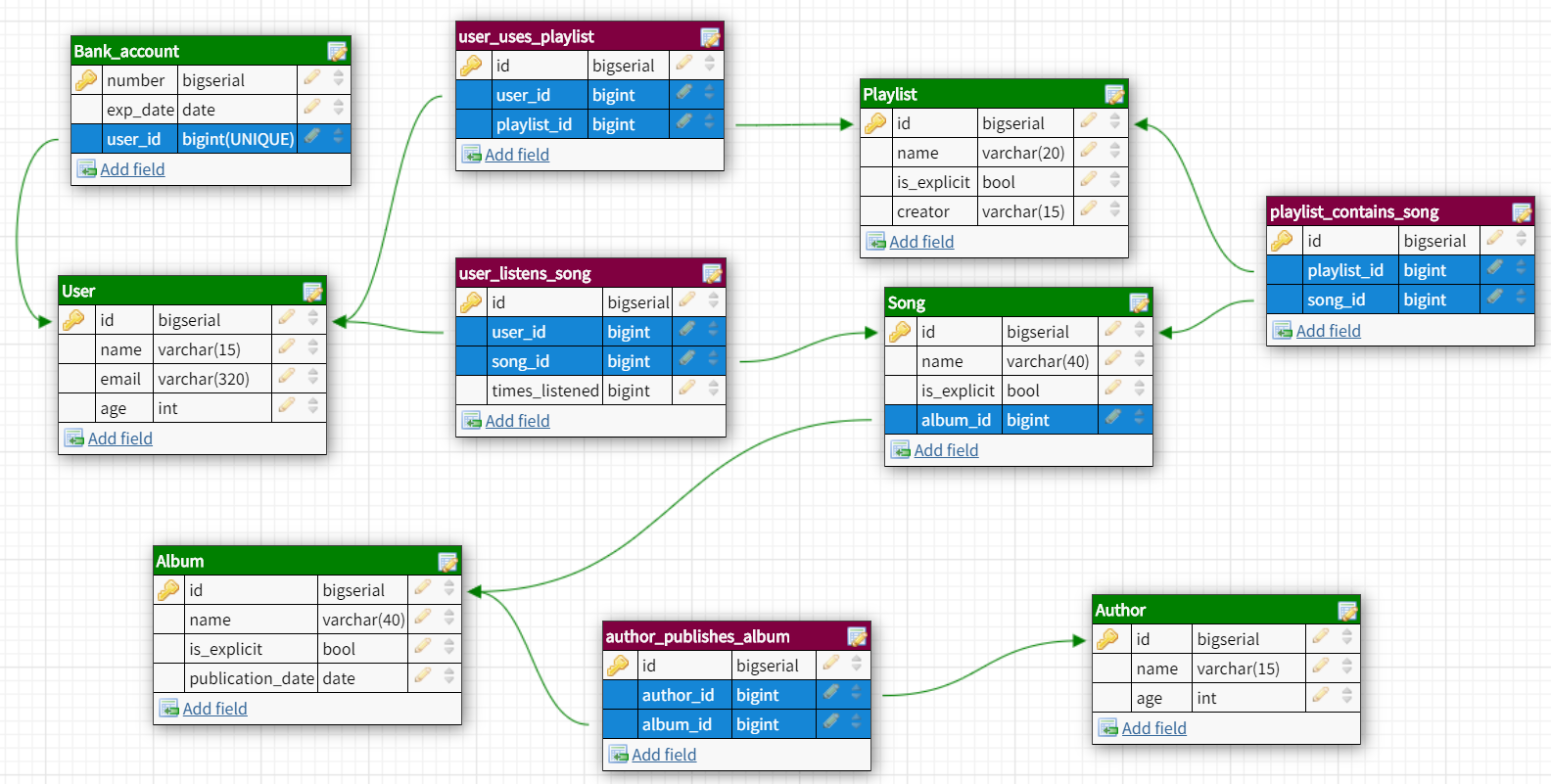
***Загальне*** ***завдання*** роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

***Варіант:***

***Предметна область:*** база даних для музичного стрiмiнгового сервісу.

**Звіт**

**Нормалізована логічна модель даних БД** 

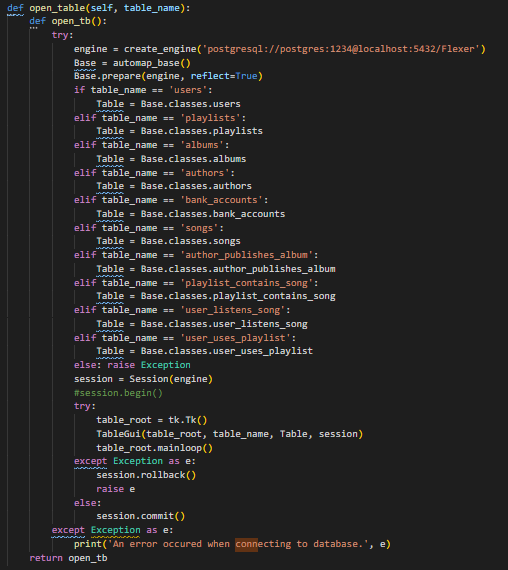
Нижче наведено схему нормалізованої бази даних спроектованої в Лабораторній роботі №1.

Рисунок 1.1 — Схема нормалізованої бази даних PostgreSQL на основі ER-моделі предметної області "Музичний стрiмiнговий сервіс".

*Примітка*. При побудові схеми БД використано сервіс Dbdesigner

Середовище розробки – Visual Studio Code. Мова програмування – Python 3.8. Бібліотека роботи з БД – SQLAlchemy

**1.** **Реалізація ORM**

Розглянемо зміни, внесені в програму для реалізації ORM

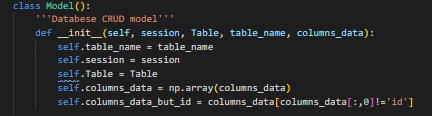
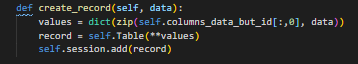
Рисунок 1.1 — Підключення до бази даних та автоматичне генерування відповідних моделей.

Рисунок 1.2 — Ініціалізація універсальної моделі.



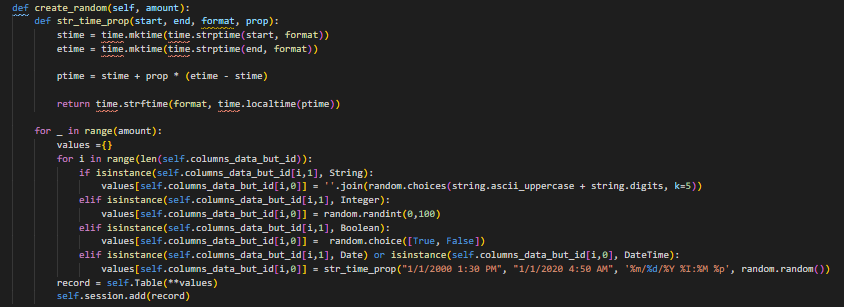
Рисунок 1.3 — Метод додавання даних до таблиці.

Рисунок 1.4 — Метод додавання рандомізованих даних до таблиці.

Рисунок 1.5 — Метод зчитування даних з таблиці.

Рисунок 1.6 — Метод оновлення даних в таблиці.

Рисунок 1.8 — Метод видалення даних з таблиці.

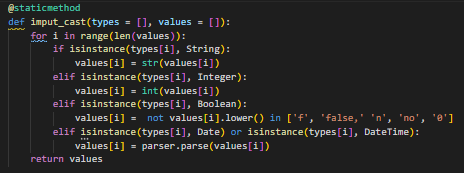


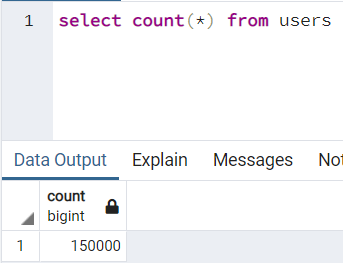
Рисунок 1.9 — Метод приведення типів даних.

Рисунок 1.10 — Метод отримання запису за параметром.

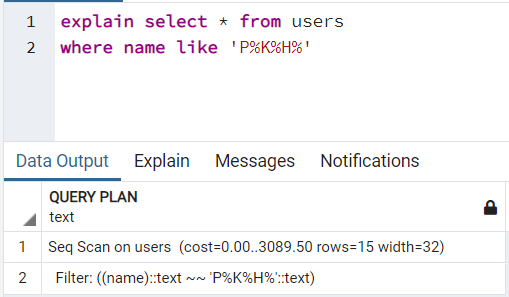
2. Аналіз індексів

**GIN**

GIN означає узагальнений інвертований індекс. GIN призначений для обробки випадків, коли елементи, що підлягають індексації, є складеними значеннями, а запити, що обробляються індексом, повинні шукати значення елементів, які відображаються в складених елементах.

Для цього завдання створимо 150000 рядків в таблиці user.

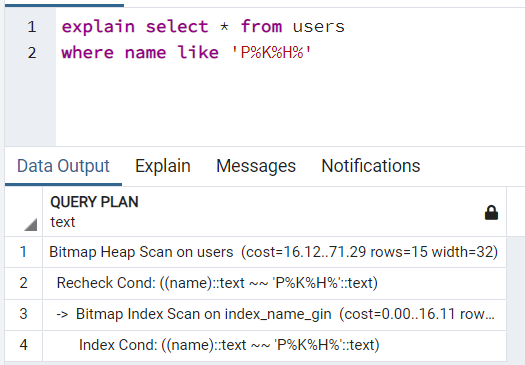
Для перевірки давайте виконаємо вибірку усіх записів, де   
name = 'P%F%H%'. В результаті ми отримали cost = 3089 при Seq Scan.



Тепер створимо GIN індекс.



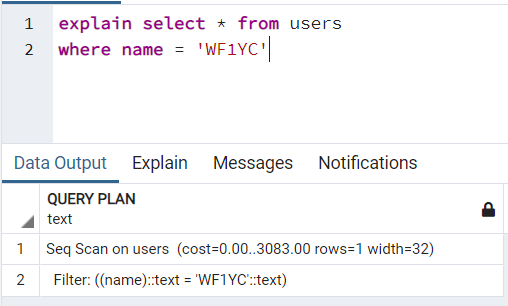
Тепер виконаємо той самий запит. І отримуємо cost = 16 при Bitmap Index Scan.



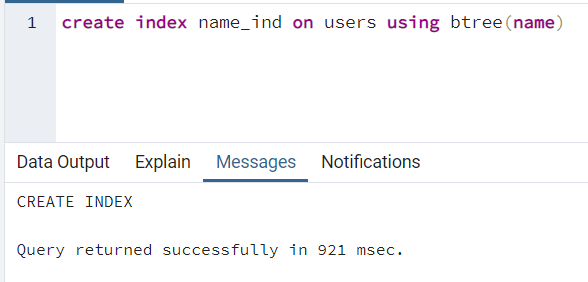
При використанні GIN індексу отримали пришвидшення в 193 рази.

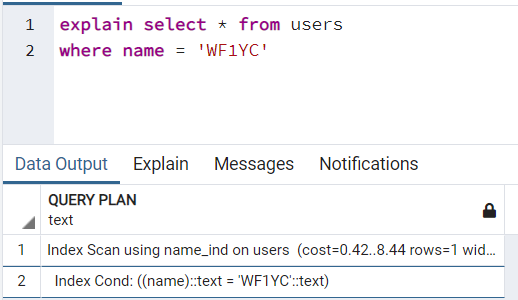
**BTree**

BTree індекс ефективний при пошуку по ключах, при використані умов( greater than, less than), пошуку по рядку або діапазону.

Тестувати цей приклад будемо на стовпчику name. Бо цей індекс ефективний при пошуку по ключах або діапазону. Зробимо пошук name=’ P%FC%H%’;

Отримали cost=3083 при Seq Scan.

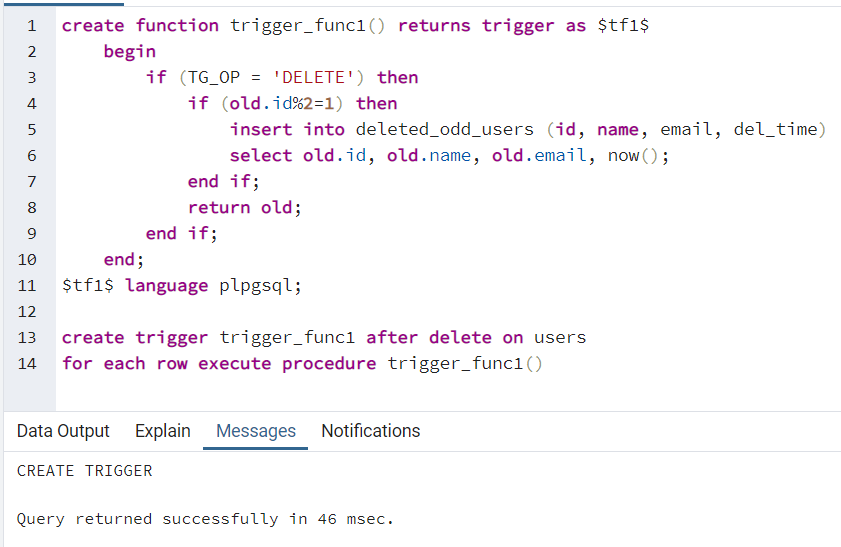
Тепер створимо BTree індекс.

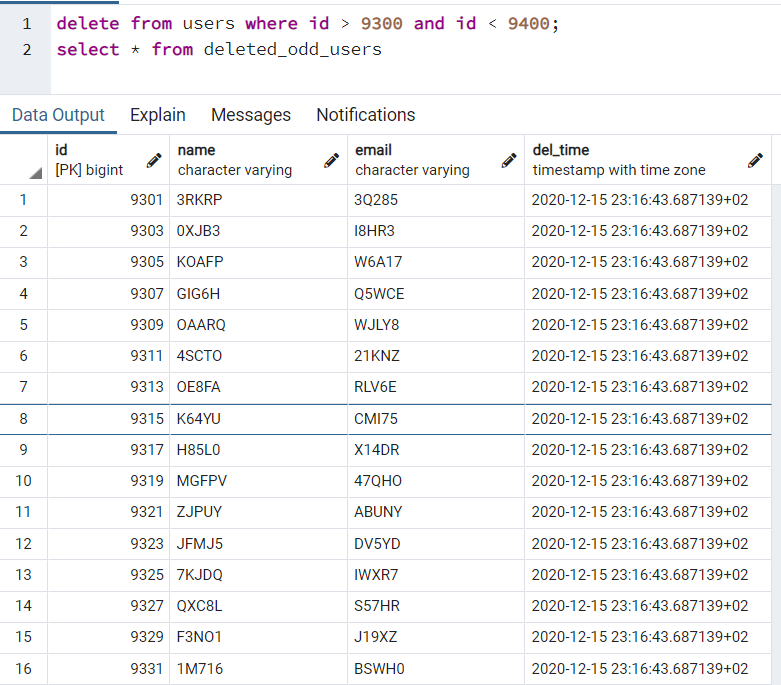
Тепер виконаємо той самий explain select запит. І отримуємо Cost = 8 при Index Scan.

При використанні Btree індексу отримали пришвидшення в 385 разів. І сам індекс є дуже універсальним і його можна використовувати майже в будь-яких ситуаціях.

3. Створення тригерів

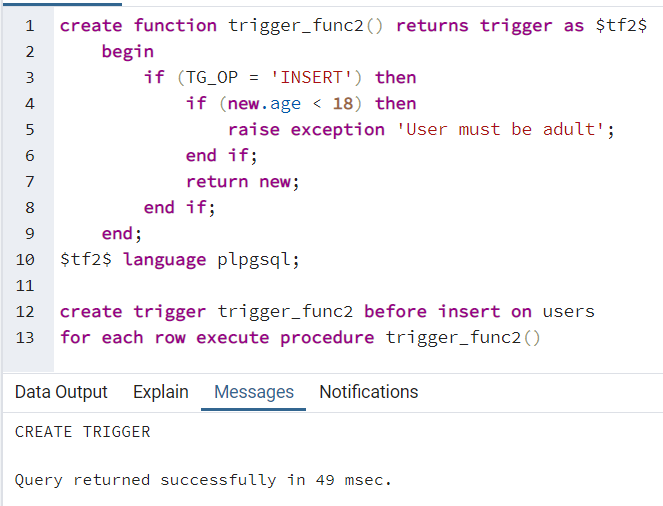
Створимо перший тригер за завданням: after delete.

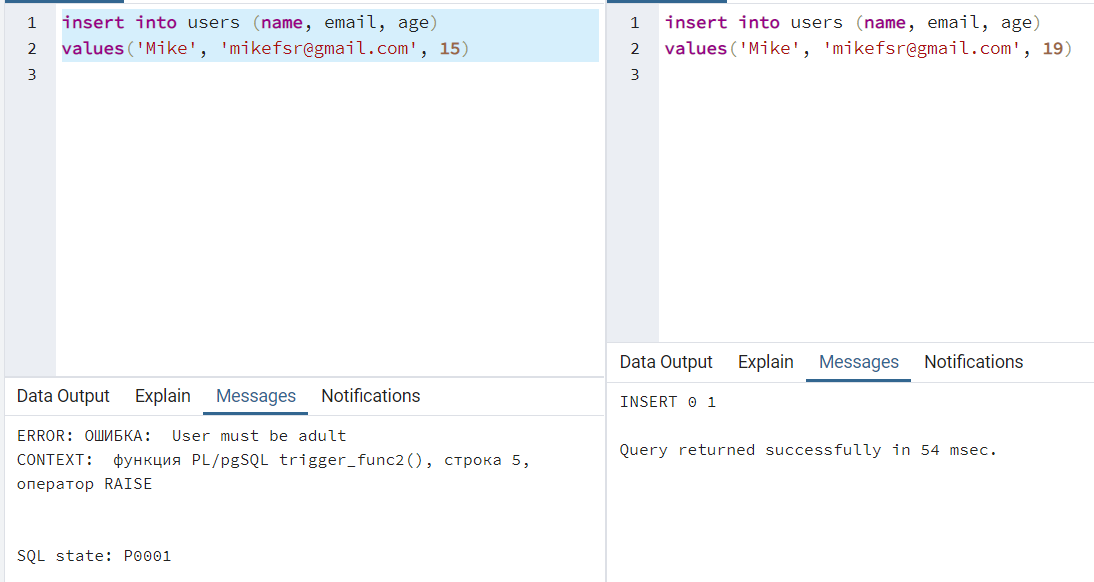
Тригер веде лог видалених рядків з таблиці users, у яких id – непарне число.

Перевіримо правильність роботи тригера.

Як бачимо, тригер виконує свою функцію.

Створимо другий тригер за завданням: insert.

Нехай тригер не дозволяє додавати записи про неповнолітніх користувачів.

Перевіримо правильність роботи тригера.

Як бачимо, тригер працює правильно