

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-84

Байдаус Михайло

Перевірив:

Київ – 2020

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

*Завдання* роботи полягає у наступному:

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

*Вимоги до пункту завдання №1*

Для перетворення функцій, що реалізують запити до об’єктної бази даних, необхідно встановити бібліотеку sqlAlchemy, налаштувати програму на роботу з ORM, розробити класи-сутності для об’єктів-сутностей, представлених відповідними таблицями БД та пов’язаних зв’язками 1:М, М:М та 1:1 виконати опис схеми бази даних. Особливу увагу приділити контролю зовнішніх зв’язків між таблицями засобами ORM.

Замінити виклики запитів мовою SQL на відповідні запити засобами SQLAlchemy по роботі з об’єктами. Обов’язковим є реалізація вставки, вилучення та редагування екземплярів класів-сутностей. Розробка запитів на генерацію даних та пошук екземплярів класів-сутностей вітається, але не є обов’язковою.

Інтерфейси функцій (вхідні та вихідні аргументи функцій модуля “Модель”) мають залишитись без змін.

*Вимоги до пункту завдання №2*

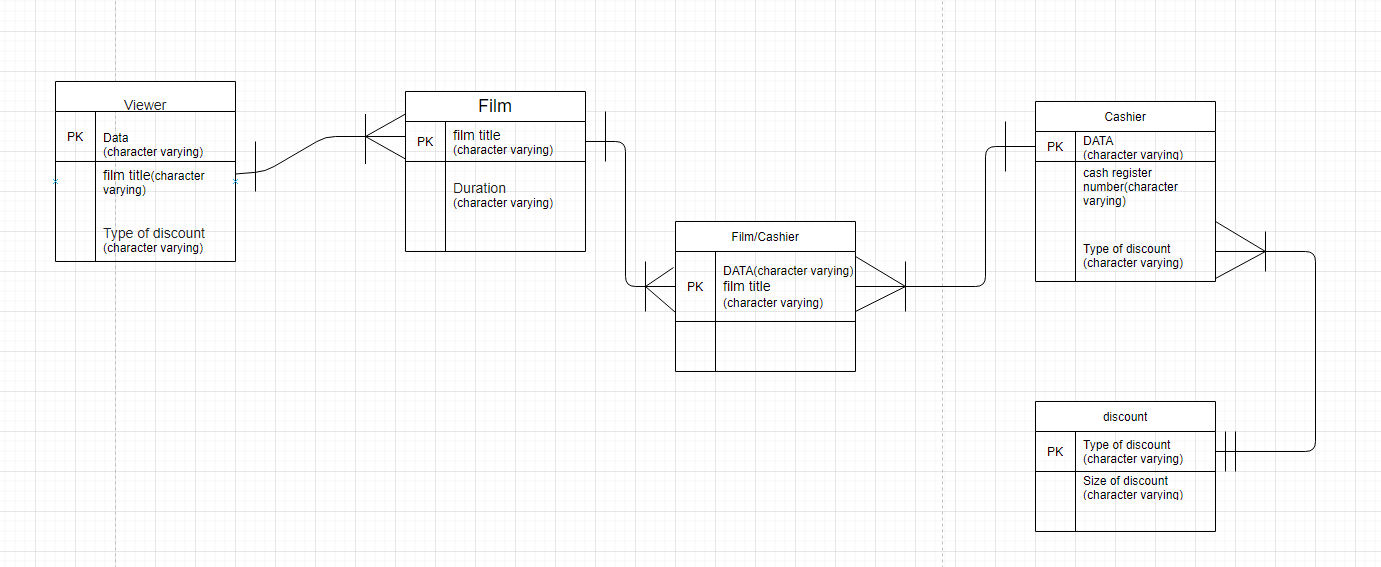
Відповідно до варіанту індексування продемонструвати на прикладах запитів SQL SELECT підвищення швидкодії їх виконання з використанням індексів, а також пояснити чому для деяких випадків індексування використовувати недоцільно. При цьому для наочного представлення слід використати функцію генерування рандомізованих даних з лабораторної роботи №2, створивши необхідну кількість тестових даних. Навести 4-5 прикладів запитів SELECT (із виведенням результуючих даних), що містять фільтрацію, агрегатні функції, групування та сортування (у необхідних комбінаціях).

*Вимоги до пункту завдання №3*

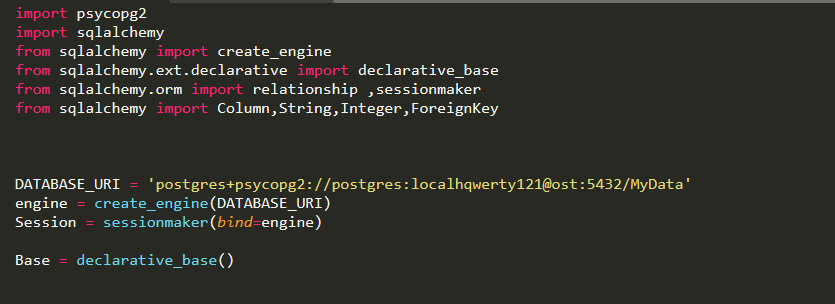
Створити тригер бази даних PostgreSQL відповідно до варіанта. Тригерна функція має включати обробку запису, що модифікується (вставляється або вилучається), умовні оператори, курсорні цикли та обробку

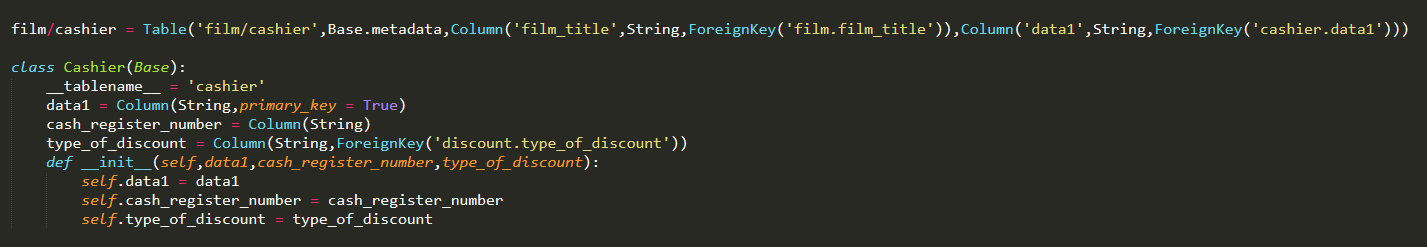
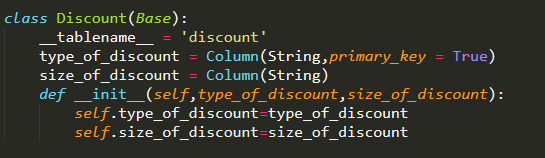
виключних ситуацій. Виконати відлагодження тригера при різних вхідних даних, навівши 2-3 приклади його використання.

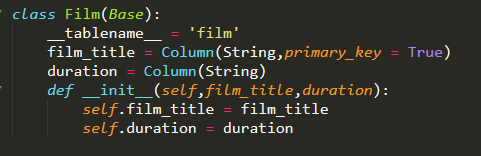


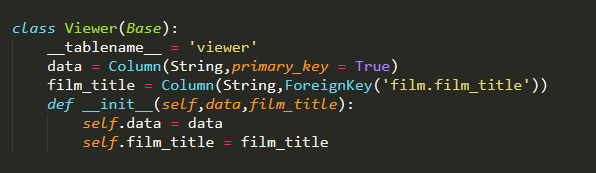
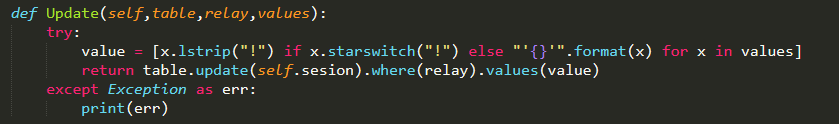
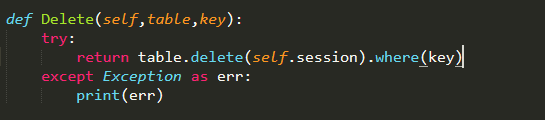
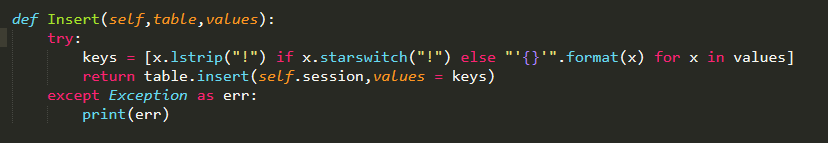


# Модель бази даних

**Завдання 1  
  
  
**

**Класи сутності  
  
  
**

****

**  
Функції Update,Delete, Insert  
** **** 

**Завдання 2**

**Створення та аналіз індексів GIN та Hash**

**GIN** - це Generalized Inverted Index, або обернений індекс. Його основною задачею є прискорення повнотекстового пошуку.

**HASH** – це режим індексу, який автоматично застосовує хеш-функцію, до даних індексу. Хоча хешування і займає додатковий час, при великої кількості даних, це може значно прискорити виконання запитів, через те, що час доступу хеш-таблиці менший ніж у звичайних колекцій.

Для створення індексів для текстового поля використовувалися такі команди:

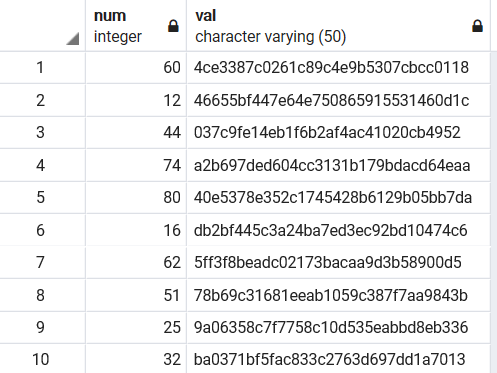
CREATE INDEX txt\_idx ON "Test" USING GIN (val gin\_trgm\_ops);

CREATE INDEX txt\_idx ON "Test" USING Hash (val);

Для створення індексів по числовому полю для Hash можливо скористатися командою:

CREATE INDEX num\_idx ON "Test" USING Hash (num);

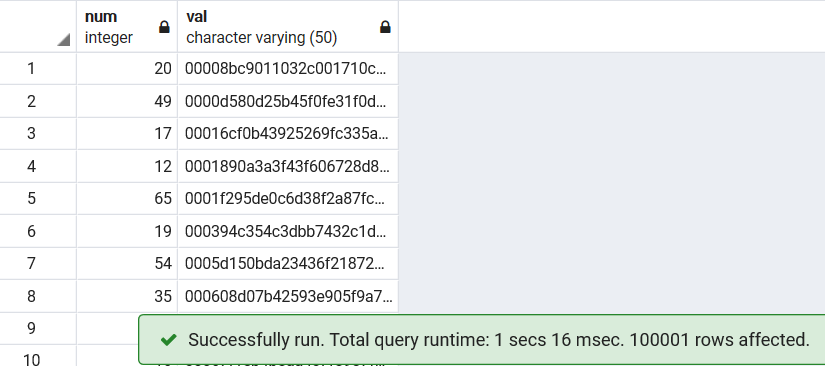
Результат запиту SELECT \* FROM public."Test"



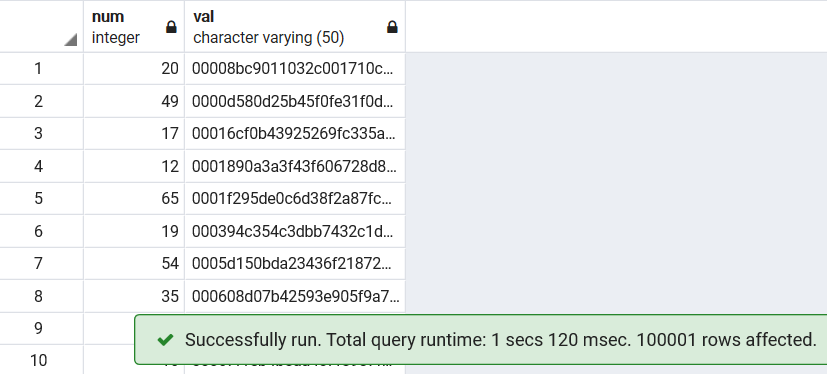
**Запит №1**

SELECT num, val FROM "Test" ORDER BY val;

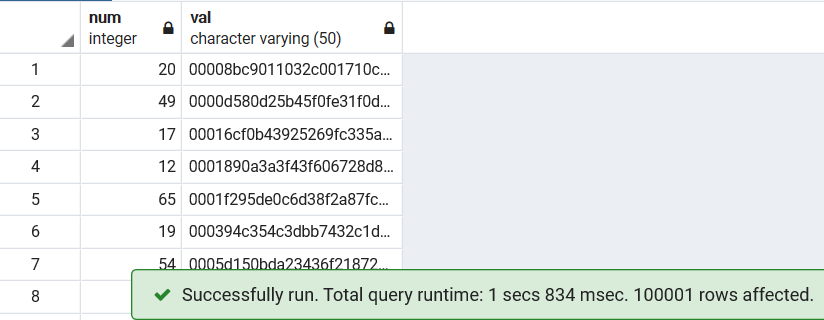
Без індексування.



Hash



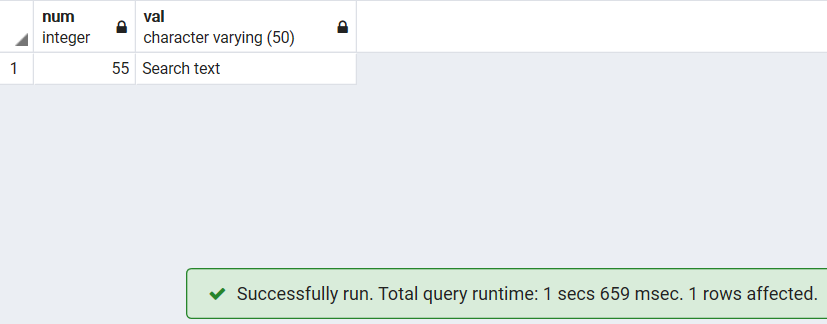
GIN



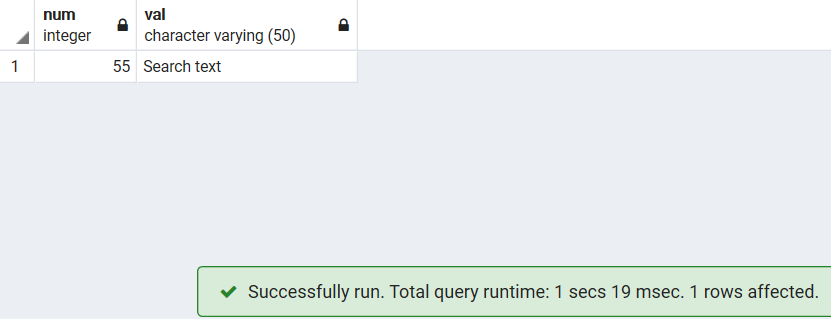
**Запит №2**

SELECT \* FROM "Test" WHERE val = 'Search text';

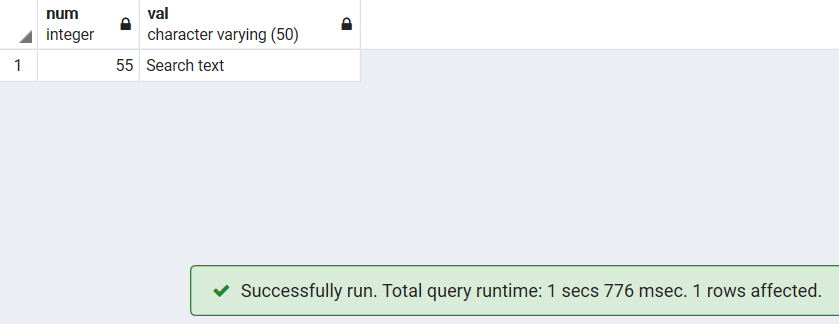
Без індексування



Hash



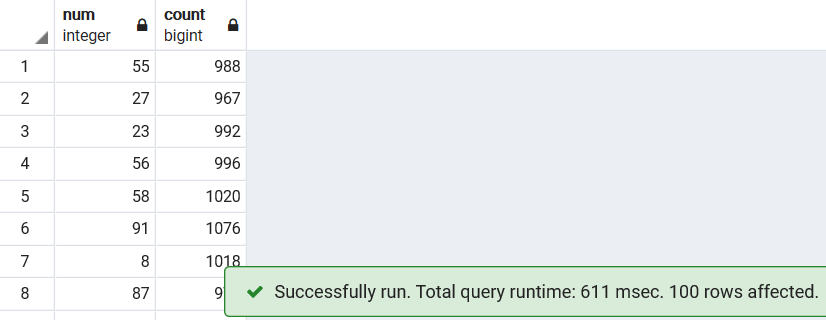
GIN



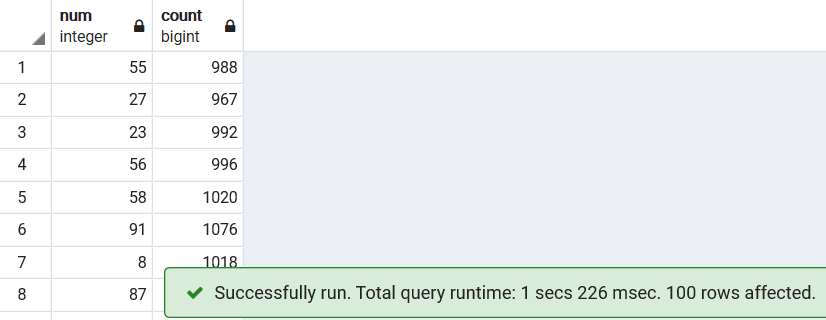
**Запит №3**

SELECT num, count(num) FROM "Test" GROUP BY num

Без індексування



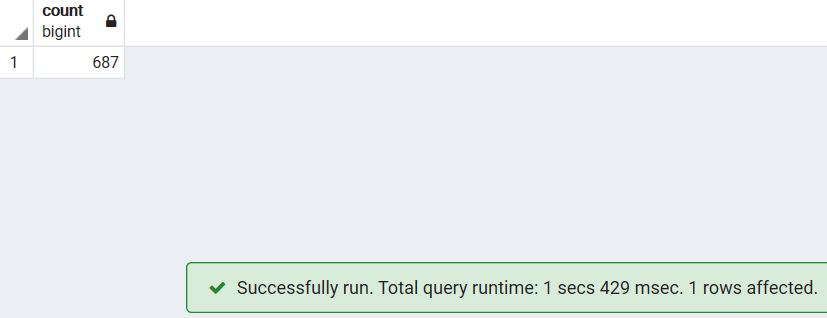
Hash



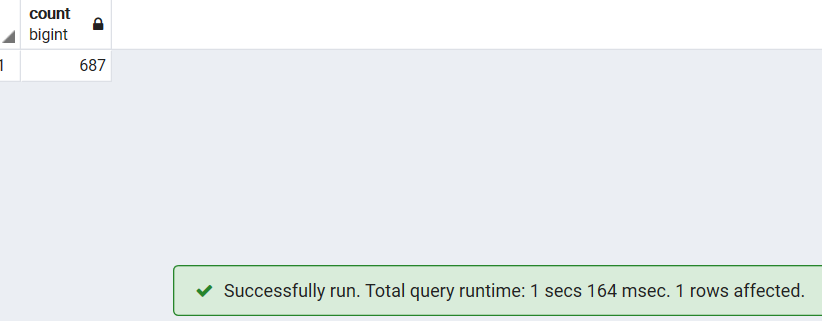
**Запит №4**

SELECT count(\*) FROM "Test" WHERE val ILIKE '%aaa%';

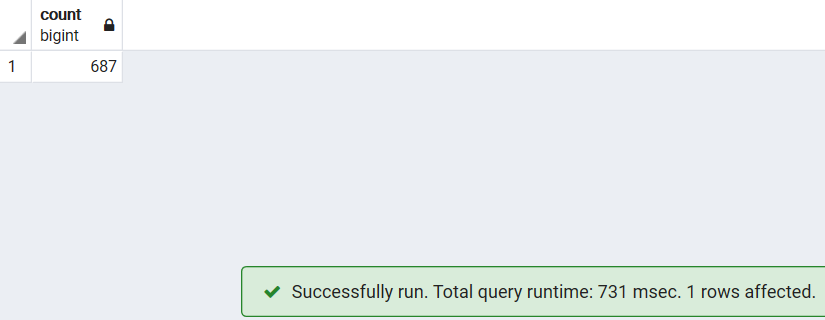
Без індексування



Hash



GIN



GIN зручно використовувати коли необхідний пошук символів у тексті. Основними недоліками є неможливість використання наприклад числових типів данних і довге створення, переіндексація.

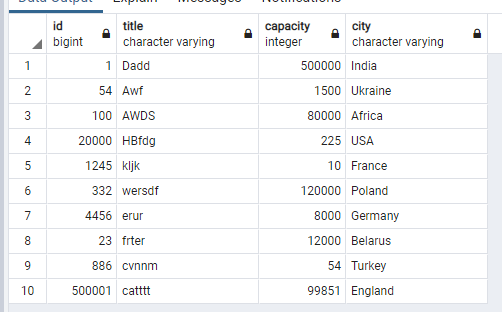
GIN добре підходить для даних, які не часто оновлюються і коли в запиті присутні такі операції як LIKE.

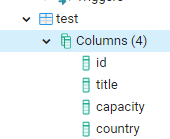
Індекс Hash зручно використовувати коли необхідно прискорити пошук і також важливим є розмір індексу. Xешування індексу ефективне, для виконання сортування, групування або пошуку даних ), проте воно неефективне, для прямої роботи з даними ,також, хешування займає значний час, тому цей спосіб буде ефективний, лише при великій кількості даних. Hash краще працює з числами ніж з рядками символів.

# Завдання 3

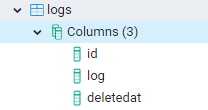
Необхідно створити тригери before delete та before update.

Для трегера створено таблицю test

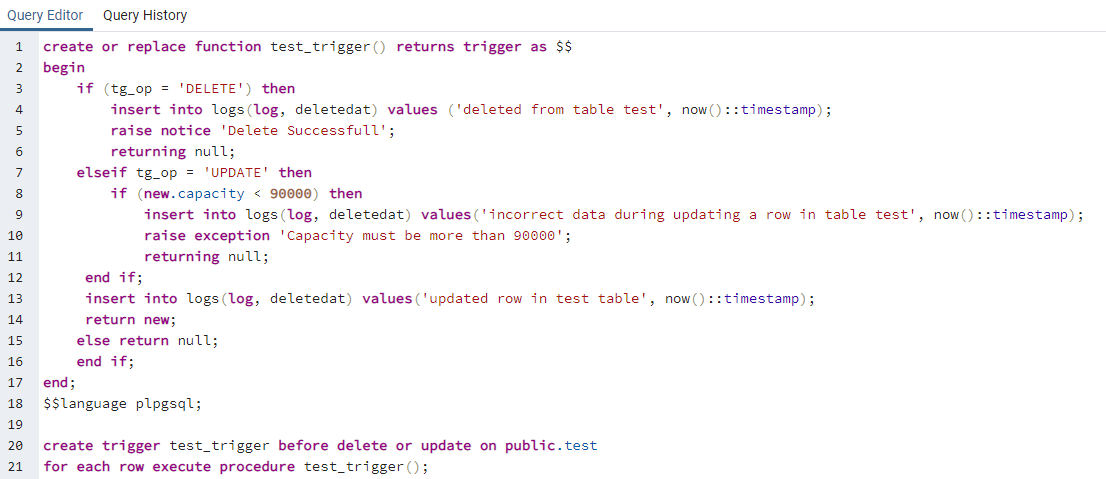




Тригер при видаленні рядка записує інформацію в таблицю logs.



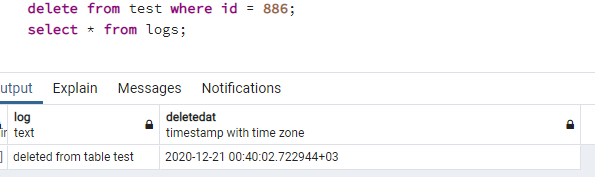
Тригер



Якщо спрацював тригер delete ми записуємо в таблицю logs повідомлення та дату видалення. В випадку update тригера, виконується перевірка, щоб в оновленого рядка значення capacity було більшим за 90 тисяч, якщо воно менше то виникає помилка з відповідним повідомленням.

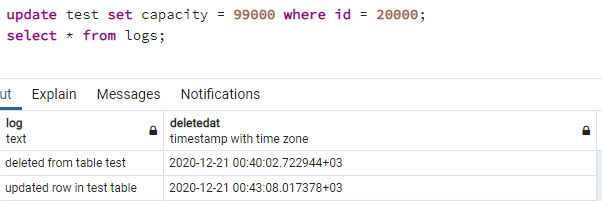
Розглянемо випадки роботи тригера.

Для випадку видалення:



При випадку оновлення рядку:

Коли дані коректні:



Коли capacity вводиться менше 90000

