

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

Групи КВ-84

Ніколайчук Данило

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*

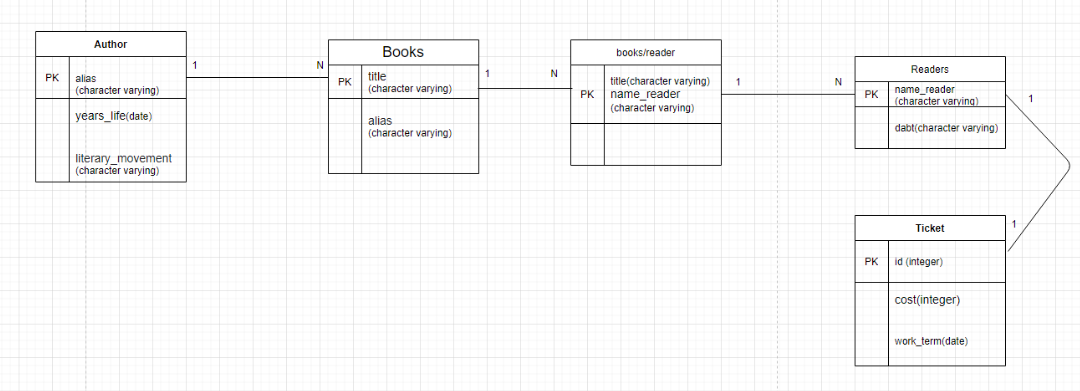
Відповідно до варіанту індексування продемонструвати на прикладах запитів SQL SELECT підвищення швидкодії їх виконання з використанням індексів, а також пояснити чому для деяких випадків індексування використовувати недоцільно. При цьому для наочного представлення слід використати функцію генерування 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).

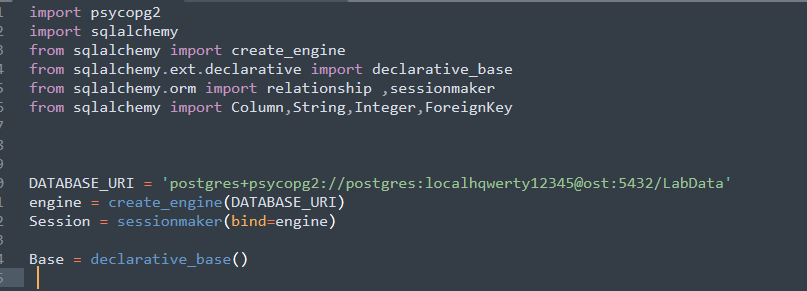
*Вимоги до пункту завдання №3*

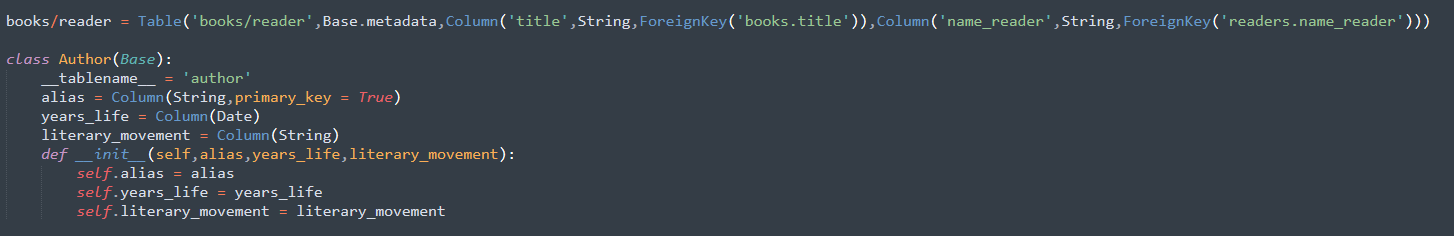
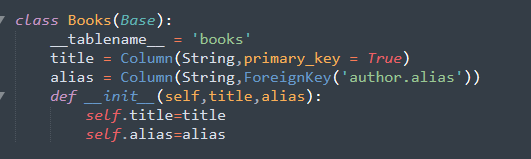
Створити тригер бази даних PostgreSQL відповідно до варіанта. Тригерна функція має включати обробку запису, що модифікується (вставляється або вилучається), умовні оператори, курсорні цикли та обробку

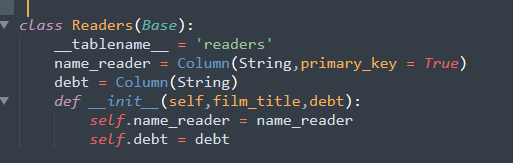
виключних ситуацій. Виконати відлагодження тригера при різних вхідних даних, навівши 2-3 приклади його використання.

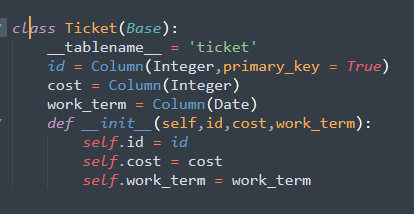


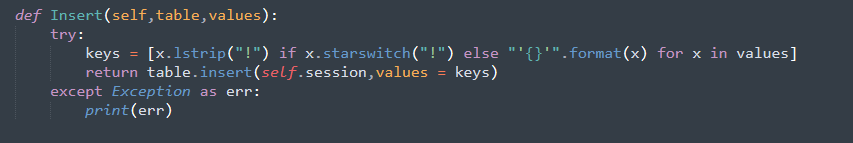


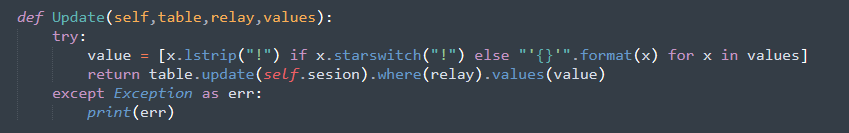
**Завдання 1  
  
  
**

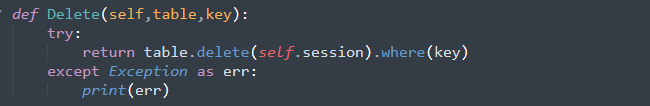
**Класи сутності  
  
  
**

****

****

**Функції Insert ,Update,Delete** 

****

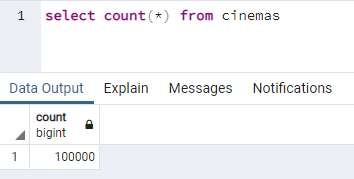
****

**Завдання 2**

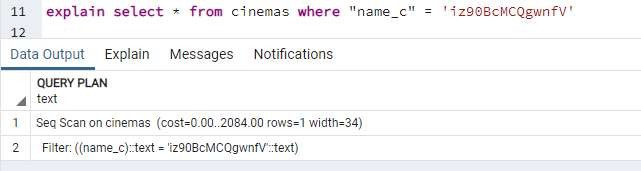
**Створення та аналіз індексів Btree та Hash**

**Btree**

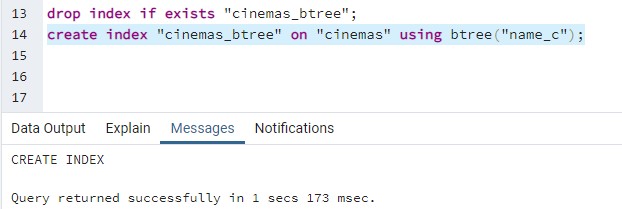
Створимо 100000 випадкових рядків в таблиці cinemas(id\_c, name\_c,street).



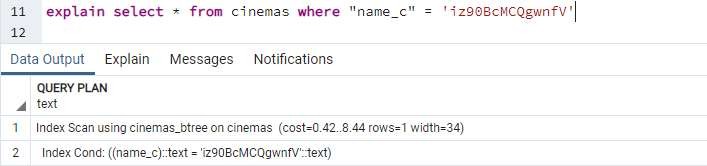
Виконаємо запит без індексу по стовпчику name\_c запису 'iz90BcMCQgwnfV'



Створимо індекс “cinemas\_btree” для таблиці “cinemas” по стовпчику “name\_c”:



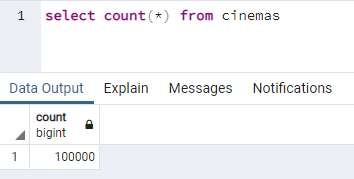
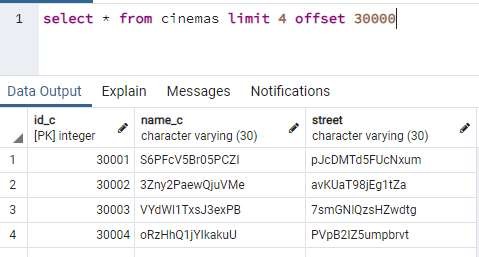
Виконаємо пошук 'iz90BcMCQgwnfV' знову.



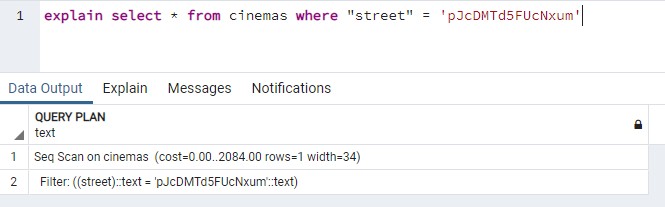
Бачимo, що пошук з індексом працює набагато швидше.

## Hash

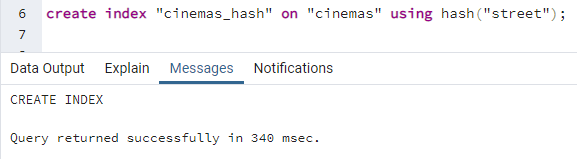
Для цього завдання знову використаємо таблицю cinemas(id\_c, name\_c,street). Слід зазначити, що індекс hash найкраще підходить для пошуку з використанням порівняння на “=”. Візьмем стовпчик street.



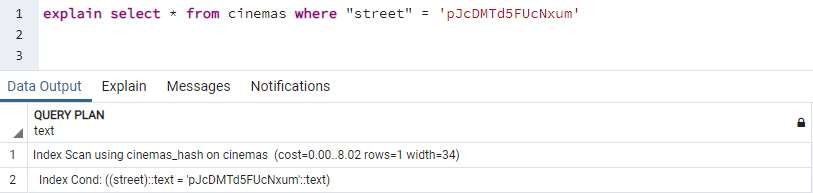
Виберемо одне з значень – "pJcDMTd5FUcNxum". Зробимо пошук по цьому імені.



Створимо індекс “cinemas\_hash” для таблиці “cinemas” по стовпчику “street”:



Виконаємо пошук "pJcDMTd5FUcNxum" знову.

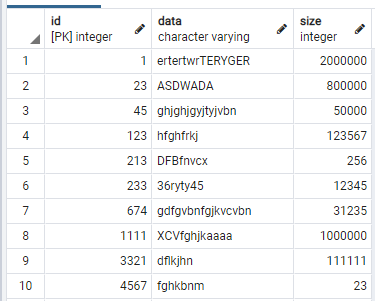


Подивившись на результати пошуку, можна сказати , що використання індексу дало досить значне підвищення швидкодії.

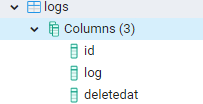
## Завдання 3

Необхідно створити тригери before delete та before update.

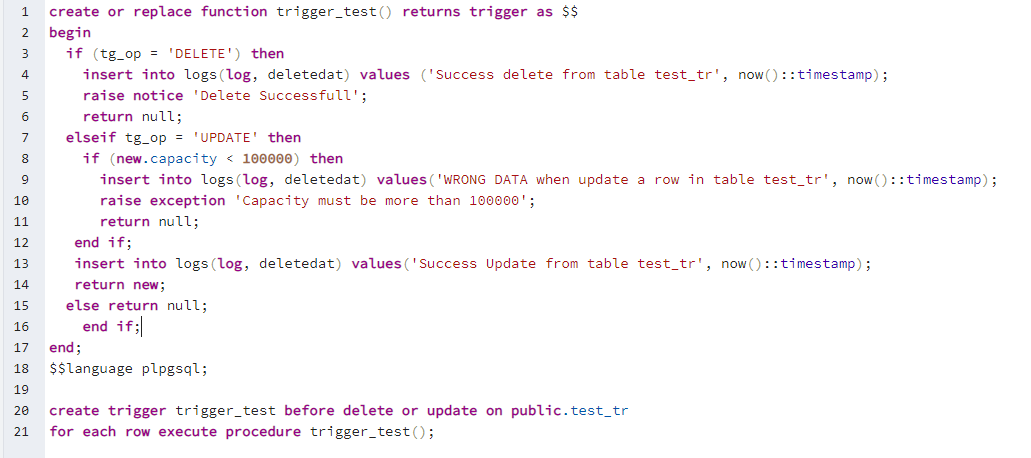
Для трегера створено таблицю test\_tr



Тригер при видаленні рядка записує інформацію в таблицю logs.



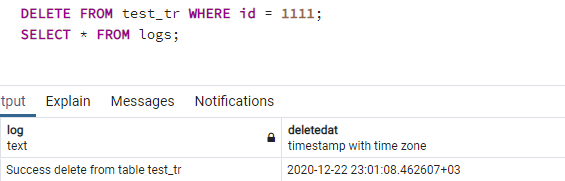
Trigger :



При роботі Delete в logs іде запис про дату видалення і час. При update виконується перевірка, щоб в оновленого рядка значення size було більшим за 100 тисяч, якщо воно менше то виникає помилка з повідомленням.

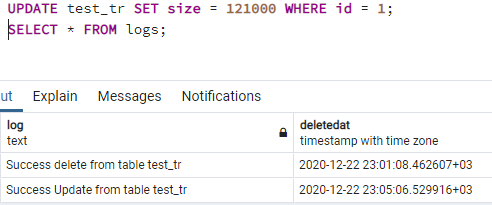
Випадки роботи тригера.

Для випадку DELETE:



При UPDATE:

Якщо size > 100000 :



Якщо size < 100000 (помилка):

