

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 ІІI курсу

ФПМ групи КВ-81

Прокопчук М.О.

Перевірив:

Київ – 2020

**Завдання**

*Загальне 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.

*Деталізоване 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.

1. 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).
2. 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.

Варіант №11

****

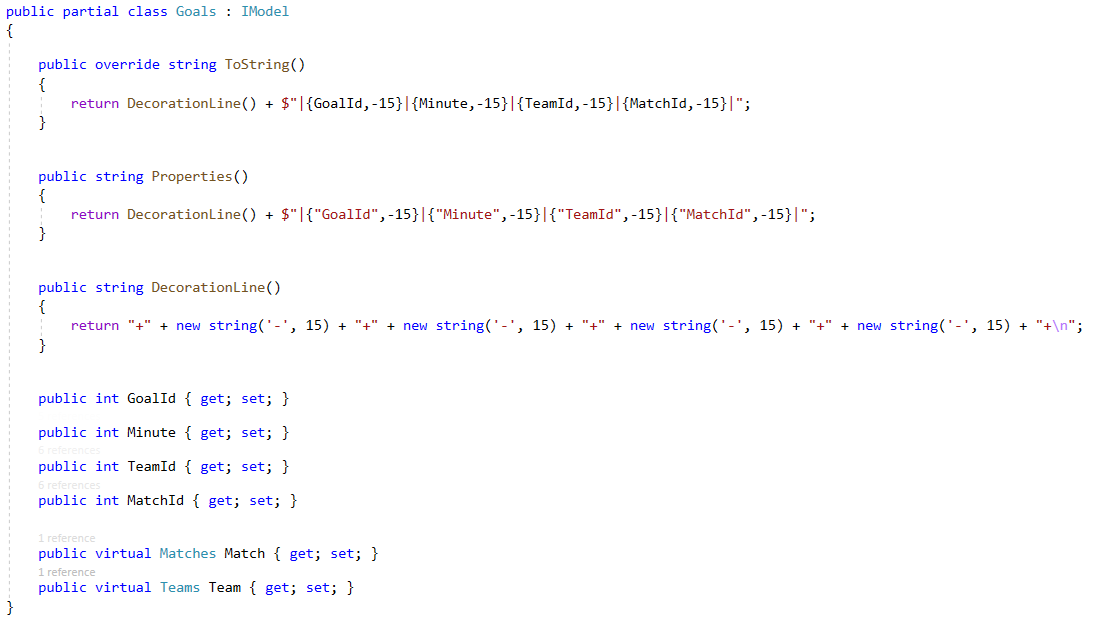
**Пункт №1**

Для реалізації першого завдання було використано entity framework (database first approach). Було створено такі класи:

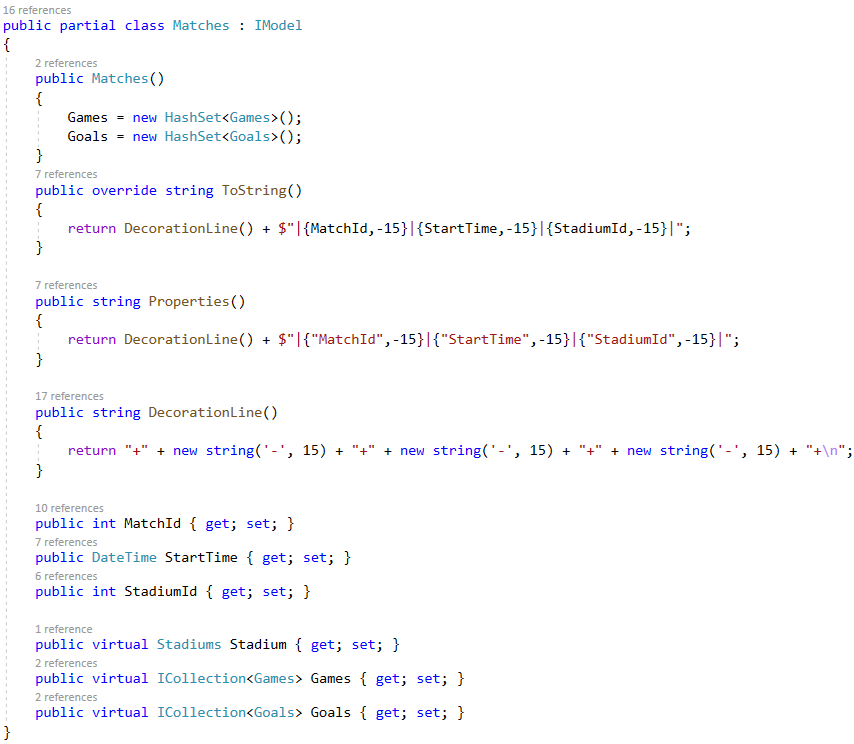
Games:



Goals:



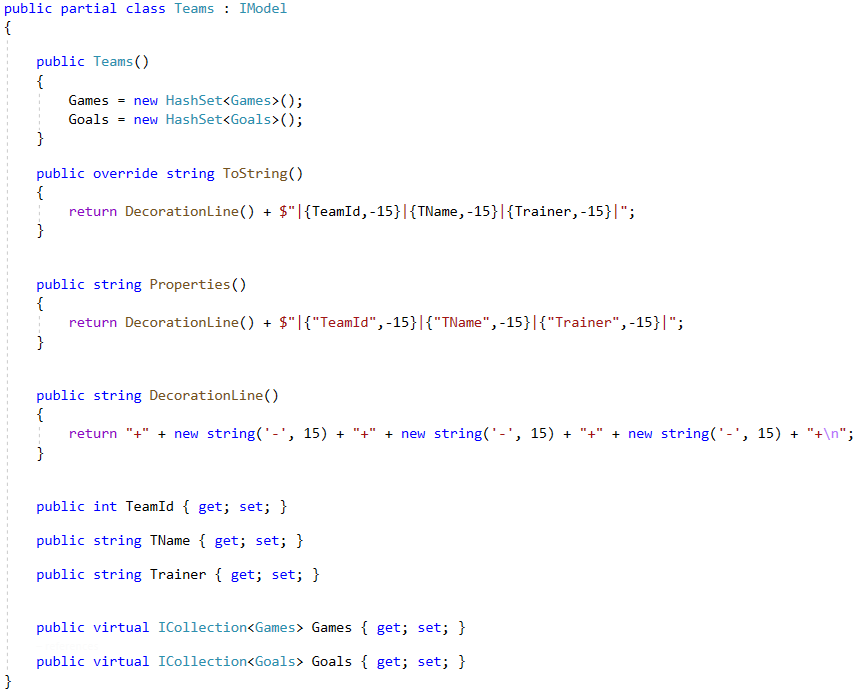
Matches:



Stadiums:

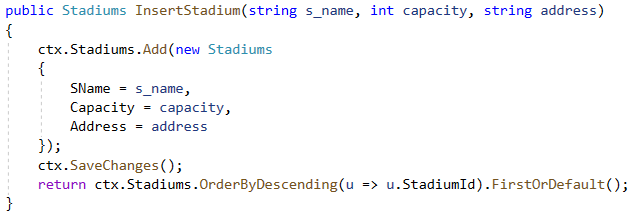


Teams:



Тепер запити CRUD виглядають таким чином (приклади будуть надані для таблиці Stadiums):

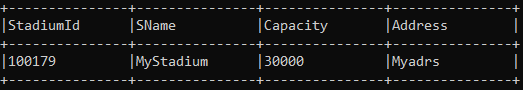
Insert:



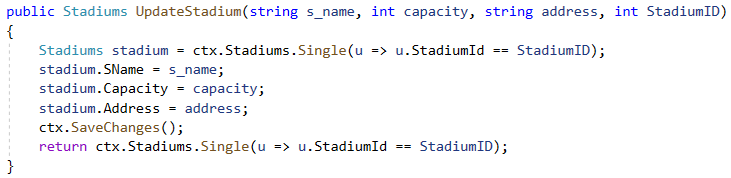
Виклик Insert:



Результат Insert:



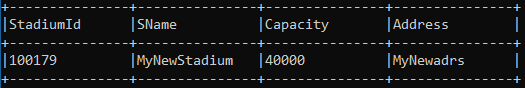
Update:



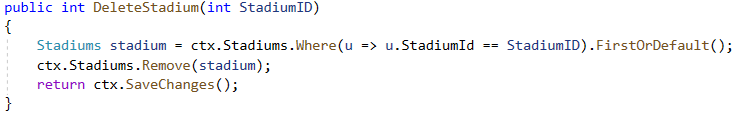
Виклик Update:



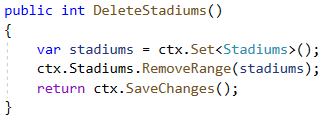
Результат Update:



Delete:



Або



Виклик Delete:

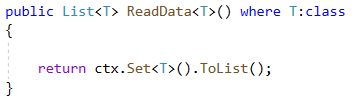


Або



У результаті повертає число видалених рядків.

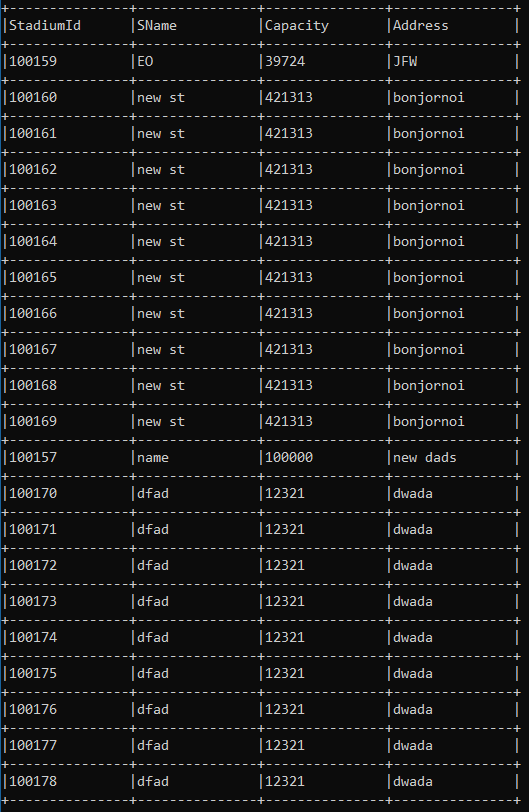
Read:



Виклик Read:

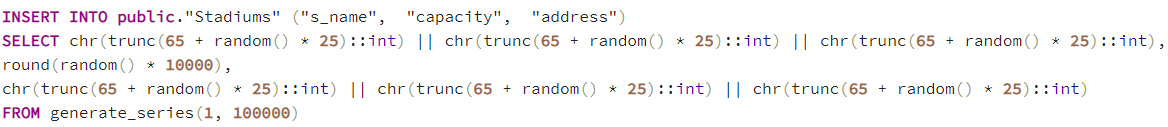


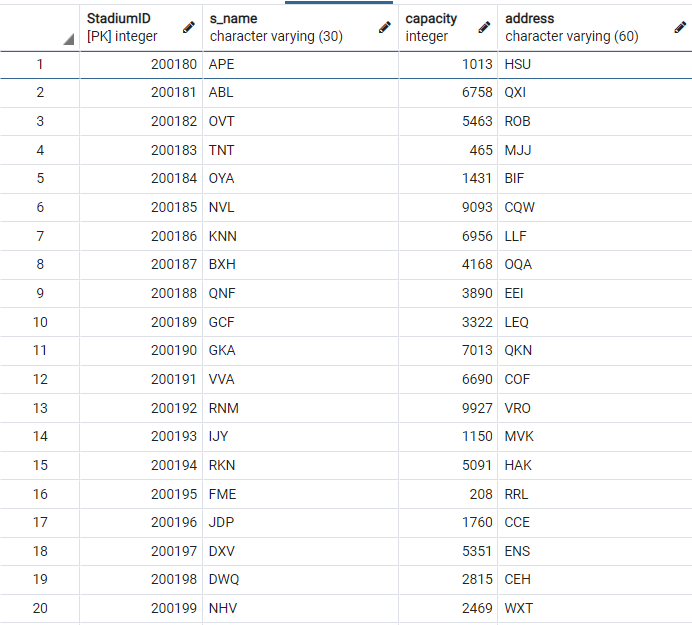
Результат Read:



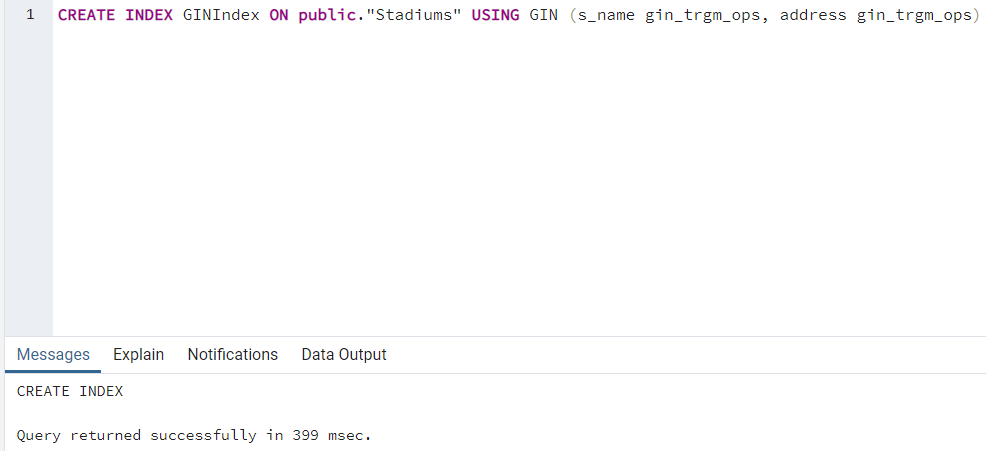
**Пункт №2**

Згенеруємо 100 000 рядків за допомогою SQL запиту:

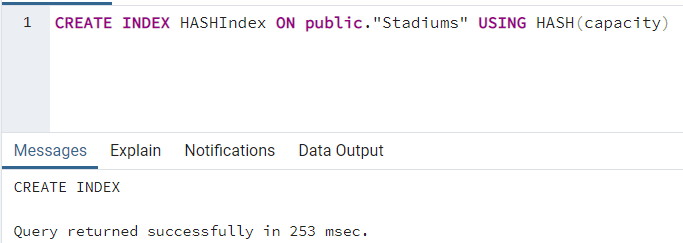
****

****

Створимо GIN індекс:

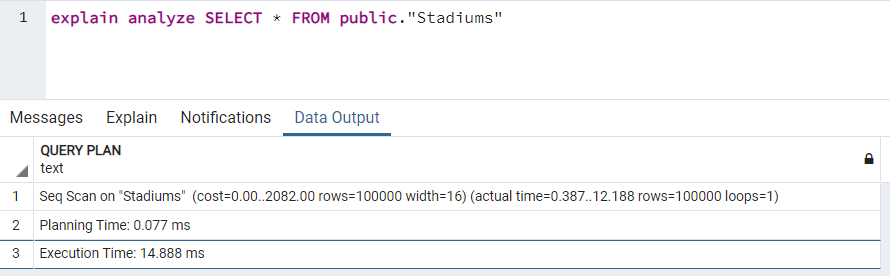


Створимо HASH індекс:

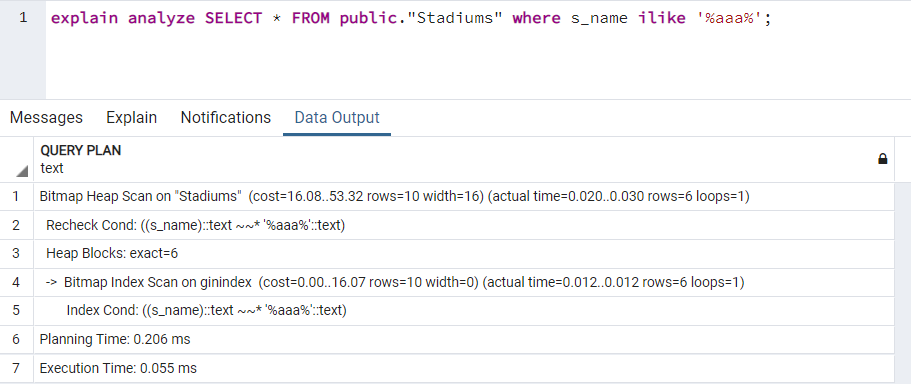


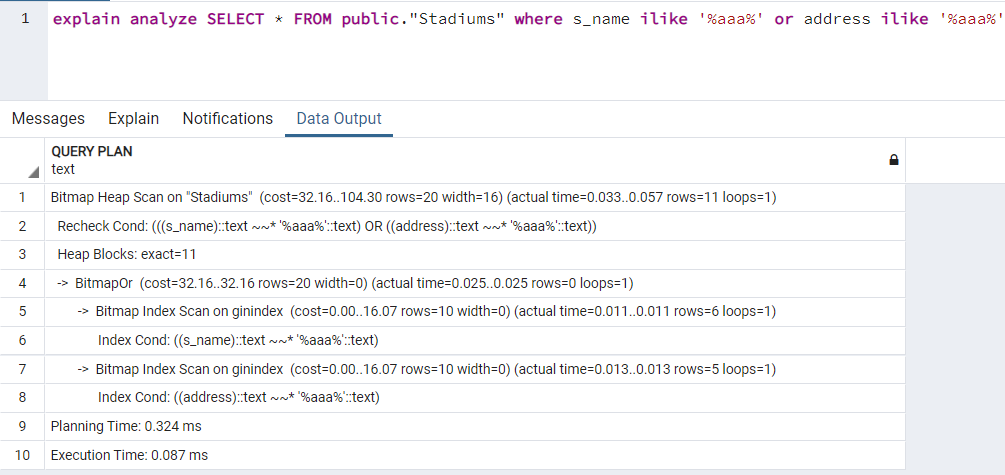
Проаналізуємо запити:

Вибірка всієї таблиці:

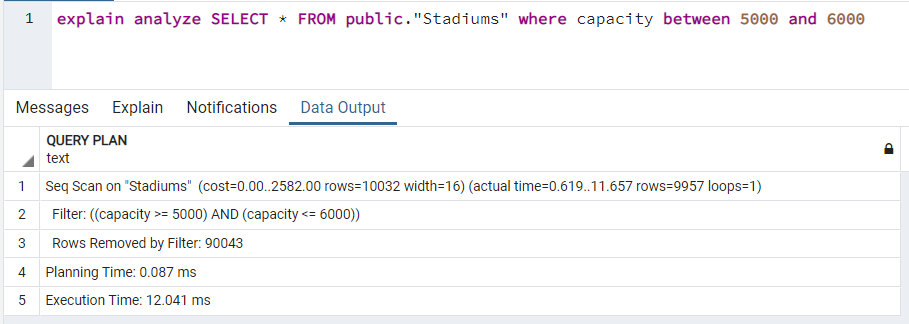


Для вибірки всієї таблиці індексування не використовується, так як вся інформація знаходиться послідовно в пам’яті.

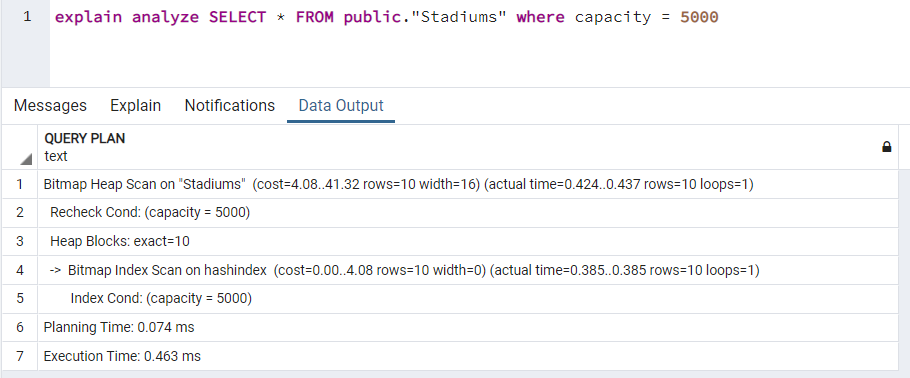
Фільтрація: 



Видно, що у цих запитах було використано GIN. Це пов’язано з тим що знайти (вибрати частину таблиці) якійсь конкретні значення набагато простіше за допомогою індексування, ніж проходити всю таблицю порядково.

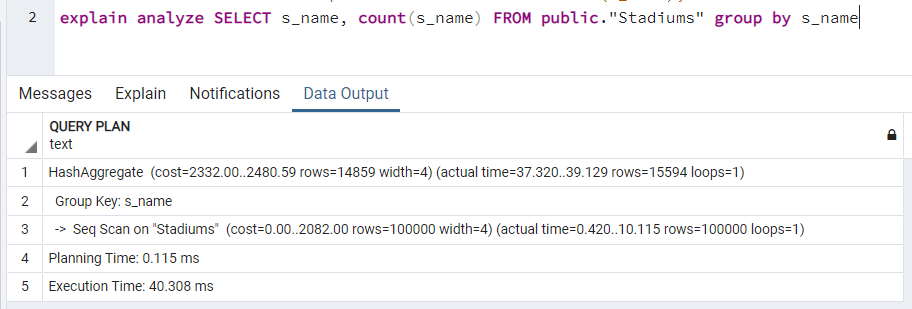


HASH індекс не може використовуватися для порівняння більше/менше, тому у цьому запиті використовуватися не може (більш детально у висновках).



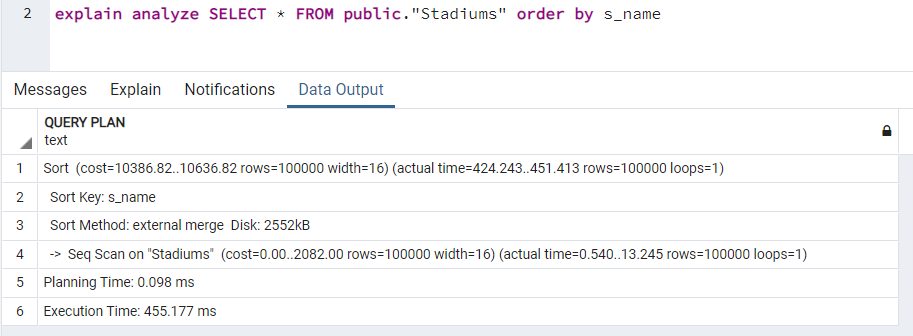
При знаходженні конкретного значення планувальник використовує HASH індексацію. HASH індекс дозволяє швидко виконати запит на рівність.

Агрегатні функції, групування:



У цьому випадку планувальник використовує тимчасову хеш-таблицю для групування записів. Операція HashAggregate не вимагає попередньо упорядкованого набору даних, натомість вона використовує великий обсяг пам'яті для матеріалізації проміжного результату.

Cортування:

****

Postgres не використовує індексацію для сортування, так як для проведення сортування всієї таблиці потрібно повністю її просканувати. Послідовне сканування у цьому випадку набагато швидше, ніж індексне сканування.

Отже, HASH індекс зберігає не значення, а їхні хеші. Такий спосіб індексування зменшує розмір і збільшує швидкість на обробку полів. Запит з використанням індексів хешу буде порівнюватися не зі значенням поля, а із хеш-значенням потрібних хеш-полів.

Оскільки хеш-функції нелінійні, такий індекс неможливо сортувати. Це спричиняє неможливість використовувати порівняння більше/менше та “IS NULL” з цим індексом.

Великою перевагою є швидкість роботи (О(1)), а також те що, при додаванні нових значень в таблицю індекс не треба перебудовувати. Недоліком є чутливість до колізій і їх використовують тільки для порівняння.

GIN корисні, коли індекс повинен відображати багато значень в один рядок. GIN добре підходять для індексації значень масивів, а також для здійснення повнотекстового пошуку.

Перевагою є те, що:

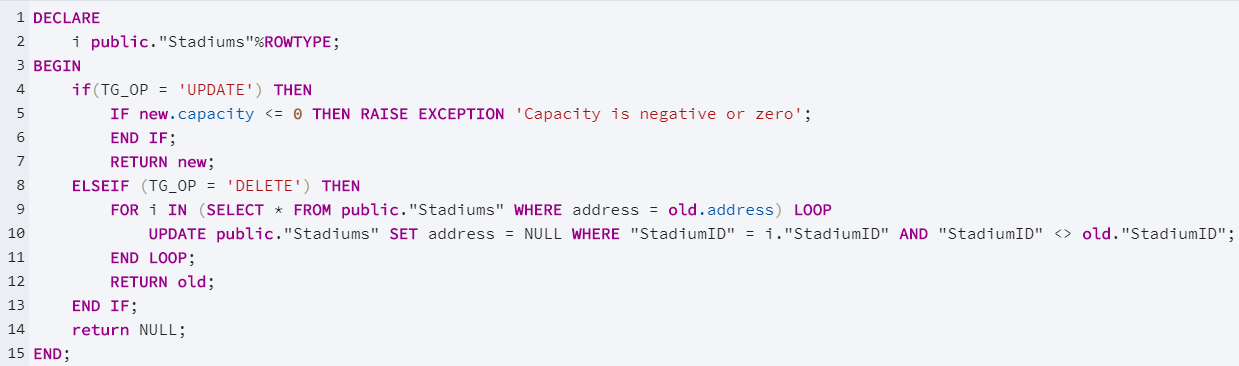
добре підходить для повнотекстового пошуку;

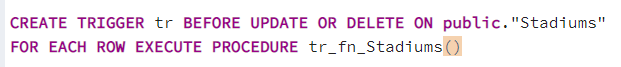
добре підходить для пошуку напівструктурованих даних;

добре працює для частого повторення елементів (і тому ідеально підходить для повнотекстового пошуку).

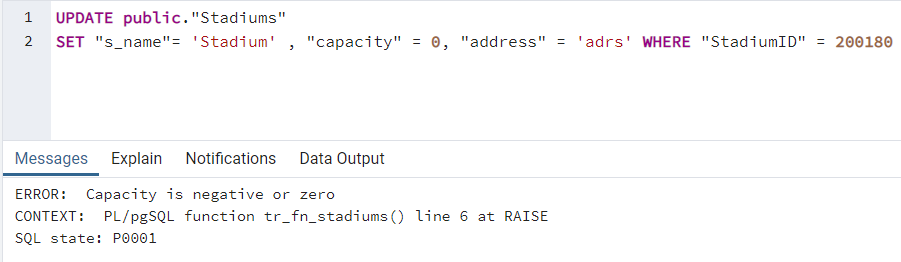
**Пункт №3**

Створимо тригер:

****

****

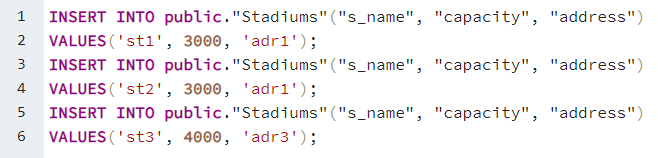
Перевірка роботи тригера для Update:

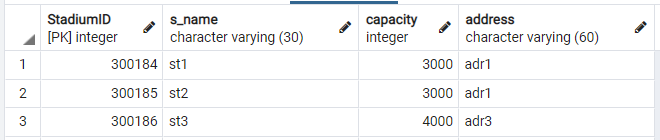


При становленні значення кількості місць нуль або менше нуля видаємо помилку.

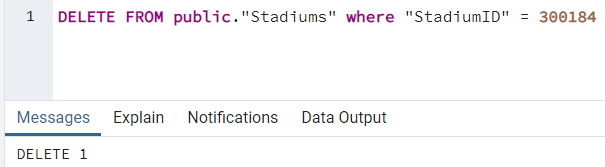
Перевірка роботи тригера для DELETE:

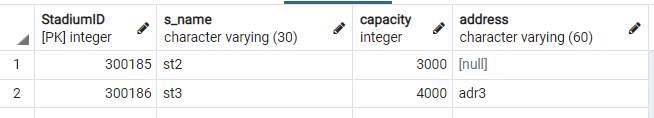
Початкова таблиця має такий вигляд:

****

****

Після видалення рядку отримуємо:

****

****

Тригер знаходить в таблиці стадіони які мають такий самий адрес і замінює ці адреси на NULL.

**Пункт №4**

Код програми можна знайти за посиланням [тут](https://github.com/FaceandControl/DataBaseLabs) (репозиторій github).