

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»**

Виконав: студент ІII курсу

ФПМ групи КВ-83

Станіслав Непочатих

Київ – 2020

**Лабораторна робота № 3.**

Засоби оптимізації роботи СУБД PostgreSQL

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

Завдання роботи полягає у наступному:

Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).

Створити та проаналізувати різні типи індексів у PostgreSQL.

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.

Вимоги до інструментарію

1. Бібліотека для реалізації ORM - [SQLAlchemy для Python](https://www.sqlalchemy.org/) або інша з подібною функціональністю.
2. Середовище для відлагодження SQL-запитів до бази даних – pgAdmin 4.
3. СУБД - PostgreSQL 11-12.

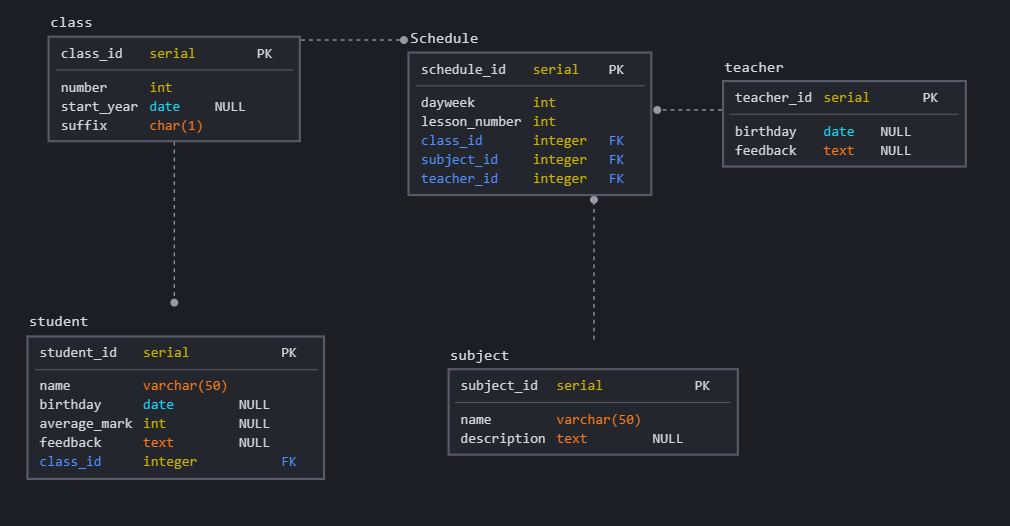
Варіант 9

Види індексів: *GIN, BRIN.*

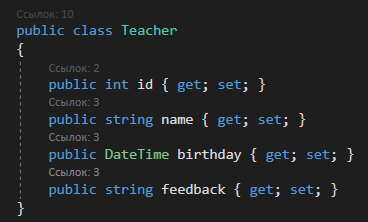
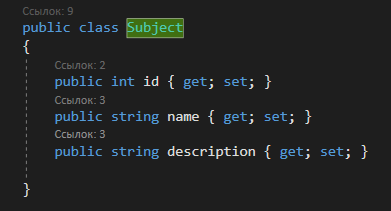
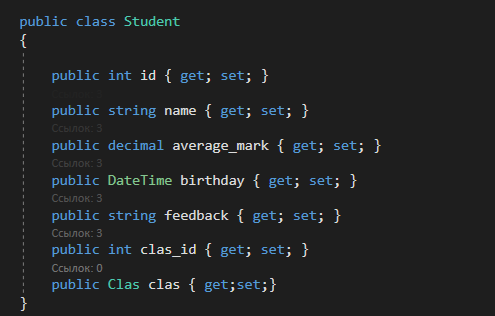
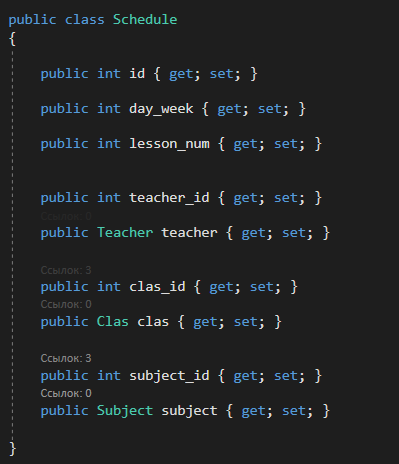
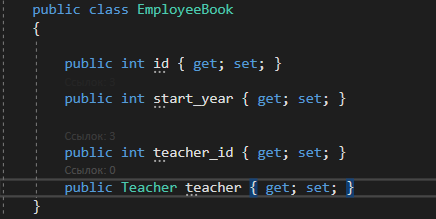
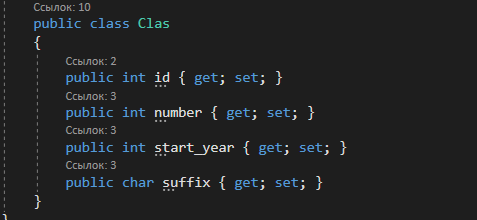
Умови для тригера: before update, delete.

**Завдання 1**

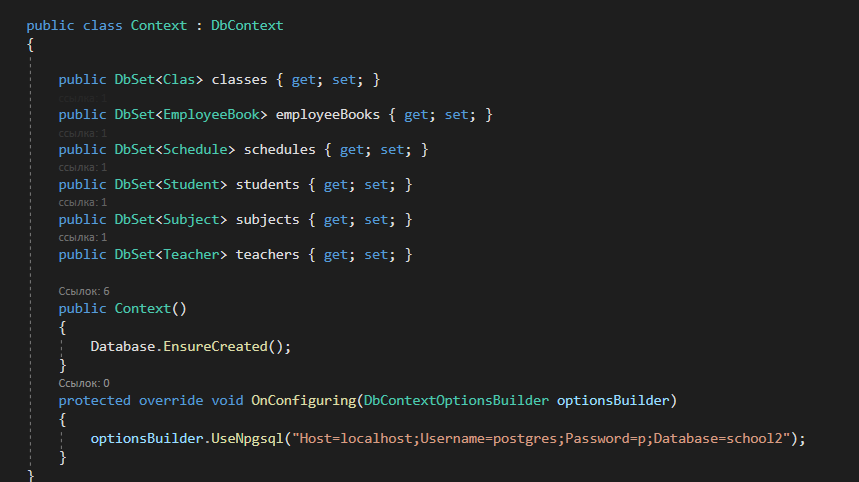
Логічна схема бази даних “Школа”



Моделі ORM

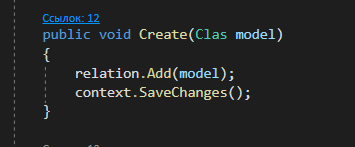


Клас контексту:

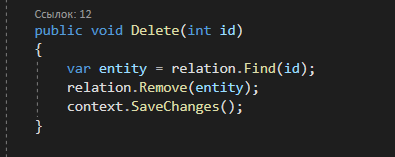


Приклади запитів у вигляді ORM

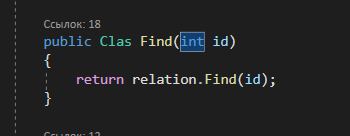
Запит додавання запису до таблиці:



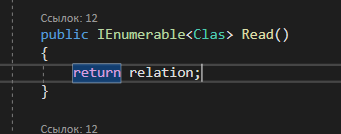
Запит видалення запису:

****

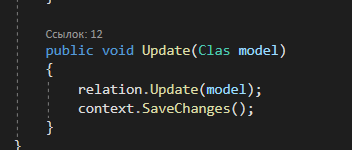
Запит пошуку:

****

Запит читання:

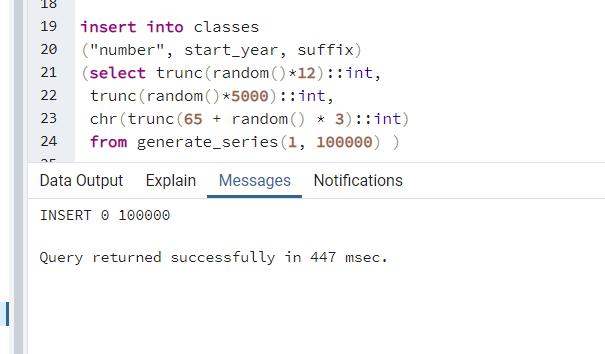


Запит оновлення:

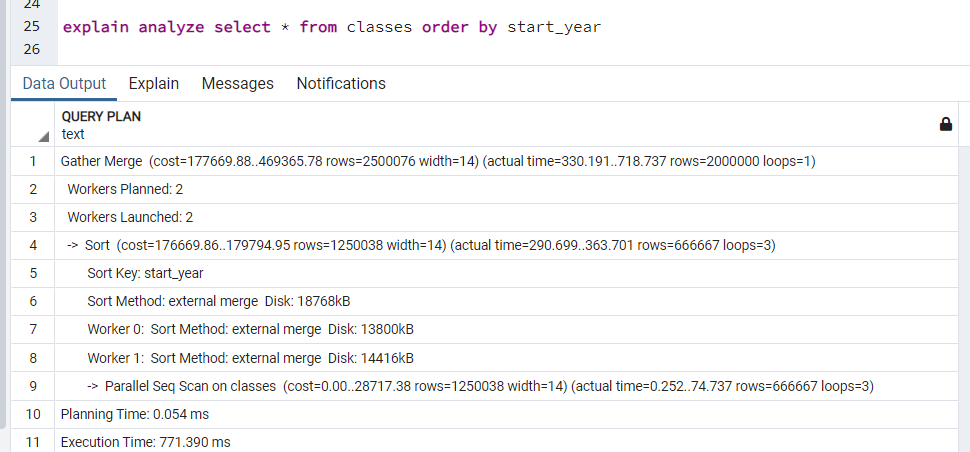


**Завдання 2**

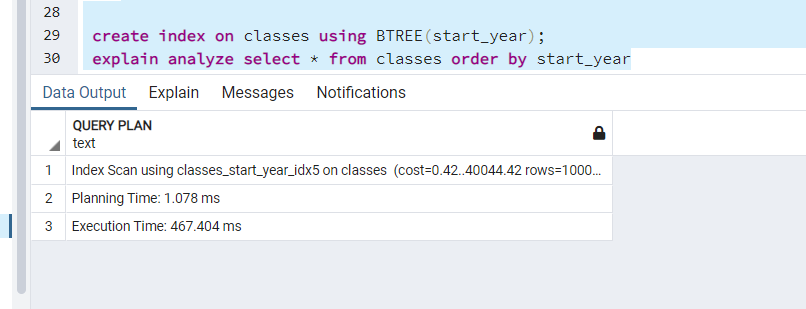
Для тестування використовуються 100000 рандомних даних, згенерованим запитом з другої лабораторної роботи.



Без використання індекса

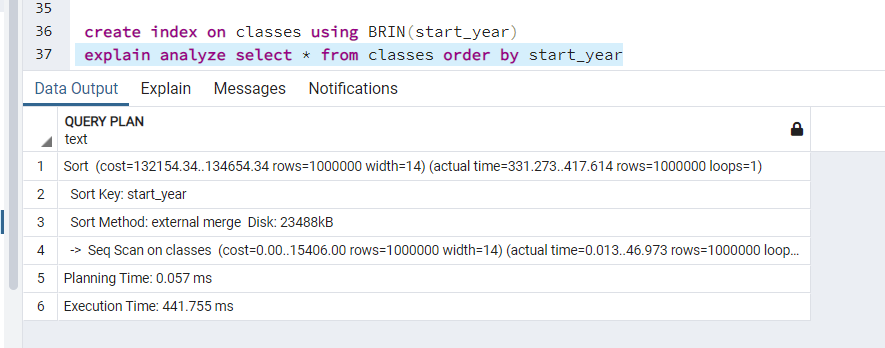


BTREE індекс



Як видно, з Btree індексом запит працює значно швидше. Це не дивно, адже цей індекс фактично зберігає в собі уже відсортовані дані, тому сортування займає значно менше часу. Для того щоб кожного разу знаходити найменший елемент в масиві тих, що ще не використовувались, достатньо просто рекурсивно проходити по дереву, кожного разу переходити до вузла з найменшим значенням. Коли вузол уже не ссилається ні на один інший вузол, повертаємо його значення і повертаємося на рівень вище.

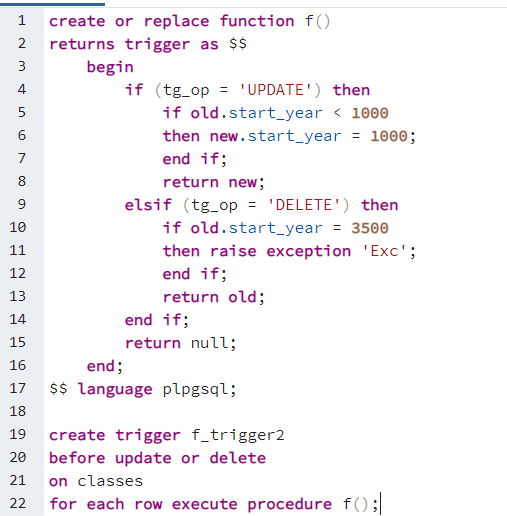
BRIN індекс



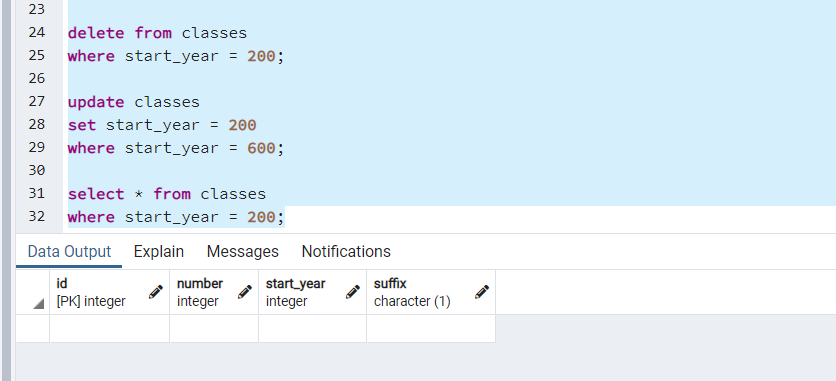
Як видно на зображенні, сортування з BRIN індексом працює швидше, ніж без нього. BRIN індекс працює так: всі дані діляться на секції, і кожного разу, коли ми шукаємо мінімальне число, ми дивимось на метадані кожної секції. Зазвичай там зберігається мінімальне і максимальне число секції, але може бути й по іншому. Це дозволяє не проглядати зайвий раз деякі ділянки пам’яті.

**Завдання 3**

Створимо функцію, яка буде викликатися перед видаленням чи оновленням даних і сам тригер:

****

Перевіримо роботу тригера. Спершу для оновлення. Для цього видалимо всі записи у яких start\_year = 200 і оновимо записи, в яких start\_year стане 200. За логікою нашого тригера, якщо запис, який оновлюється має нове значення start\_year < 1000, то він отримає start\_year = 1000. Отже, після виконання цих дій у таблиці не має бути жодного запису з start\_year = 200. Перевіримо:

****

При видаленні запису зі start\_year = 3500 має викликатися помилка:

