

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

Глеб Владислав

Київ – 2020

**Засоби оптимізації роботи СУБД PostgreSQL**

*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).

1. Створити та проаналізувати різні типи індексів у PostgreSQL.
2. Розробити тригер бази даних PostgreSQL.

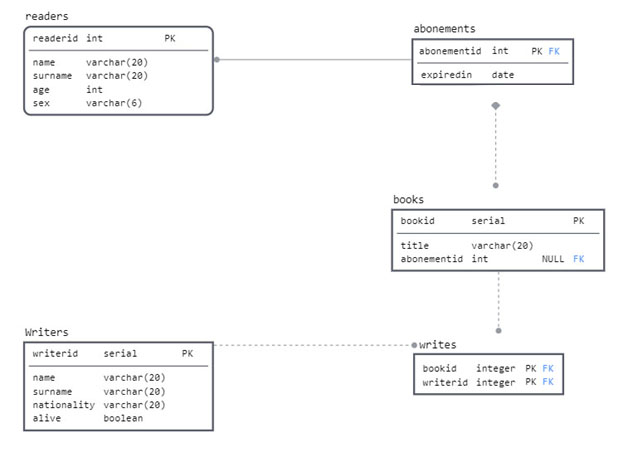
**Варіант 3**

У другому завданні проаналізувати індекси GIN,Hash.

Умова для тригера – insert

**Завдання 1**

Логічна схема бази даних “Інтернет магазин”



Для перетворення модулю “Model” програми, створеної в 2 лабораторній роботі, у вигляд об’єктно-реляційної моделі використовую бібліотеку SequelizeORM, яка є найпопулярнішою на мові JavaScript.

Зобразимо сутнісні класи програми. Продемонструємо код лише для одного класу Books:

class SeqBooks extends Model { }

SeqBooks.init({

    bookid: {

        type: DataTypes.INTEGER,

        autoIncrement: true,

        primaryKey: true

    },

    title: {

        type: DataTypes.INTEGER,

        allowNull: false

    },

    holder: {

        type: DataTypes.INTEGER,

        allowNull: true

    }

}, {

    sequelize,

    tableName: 'books',

    timestamps: false

})

SeqWrites.hasOne(SeqBooks,{foreignKey: 'bookid' ,targetKey: 'bookid'})

SeqAbonements.hasMany(SeqBooks,{foreignKey: 'holder', targetKey: 'abonementid'})

В класі Books, описується вміст таблиці і зв’язки з іншими таблицями

В цій ORM уже є готові базові CRUD (create, read, update, delete) методи. Метод для вставки у таблицю виглядає так

public static create<M extends Model>(

    this: ModelStatic<M>,

    values?: M['\_creationAttributes'],

    options?: CreateOptions<M['\_attributes']>

  ): Promise<M>;

Метод для модифікації даних

public static update<M extends Model>(

    this: ModelStatic<M>,

    values: Partial<M['\_attributes']>,

    options: UpdateOptions<M['\_attributes']>

  ): Promise<[number, M[]]>;

Та функція для видалення даних.

public static destroy<M extends Model>(

    this: ModelStatic<M>,

    options?: DestroyOptions<M['\_attributes']>

  ): Promise<number>;

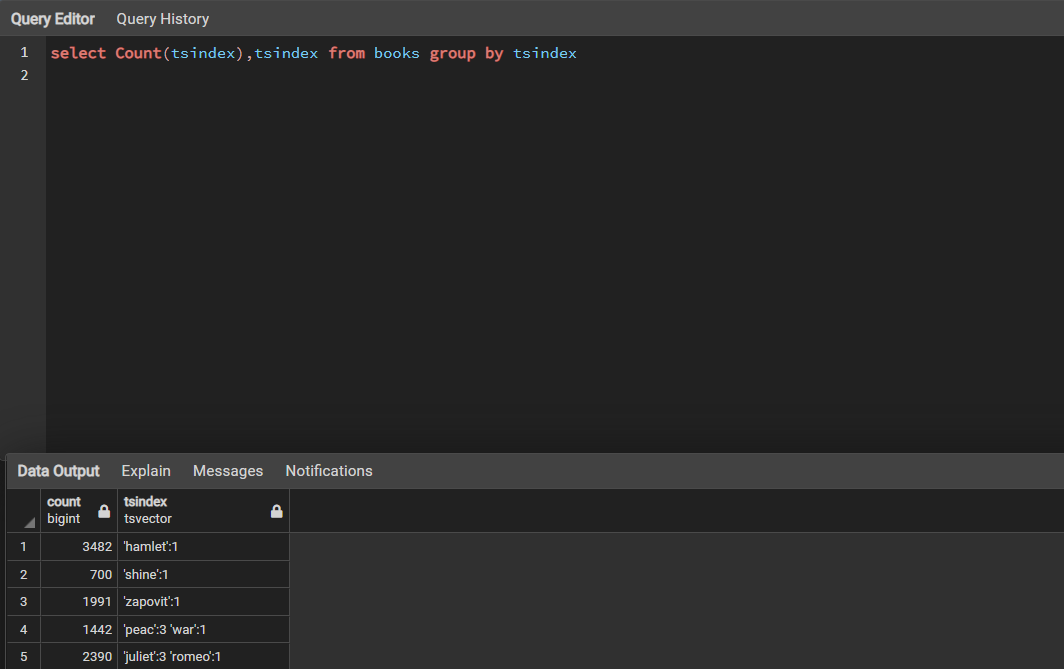
**Завдання 2**

**Створення та аналіз індекса GIN**

**GIN** - це Generalized Inverted Index, або обернений індекс. Його основною задачею є прискорення повнотекстового пошуку і тому вивчати даний індекс будемо на цьому прикладі.

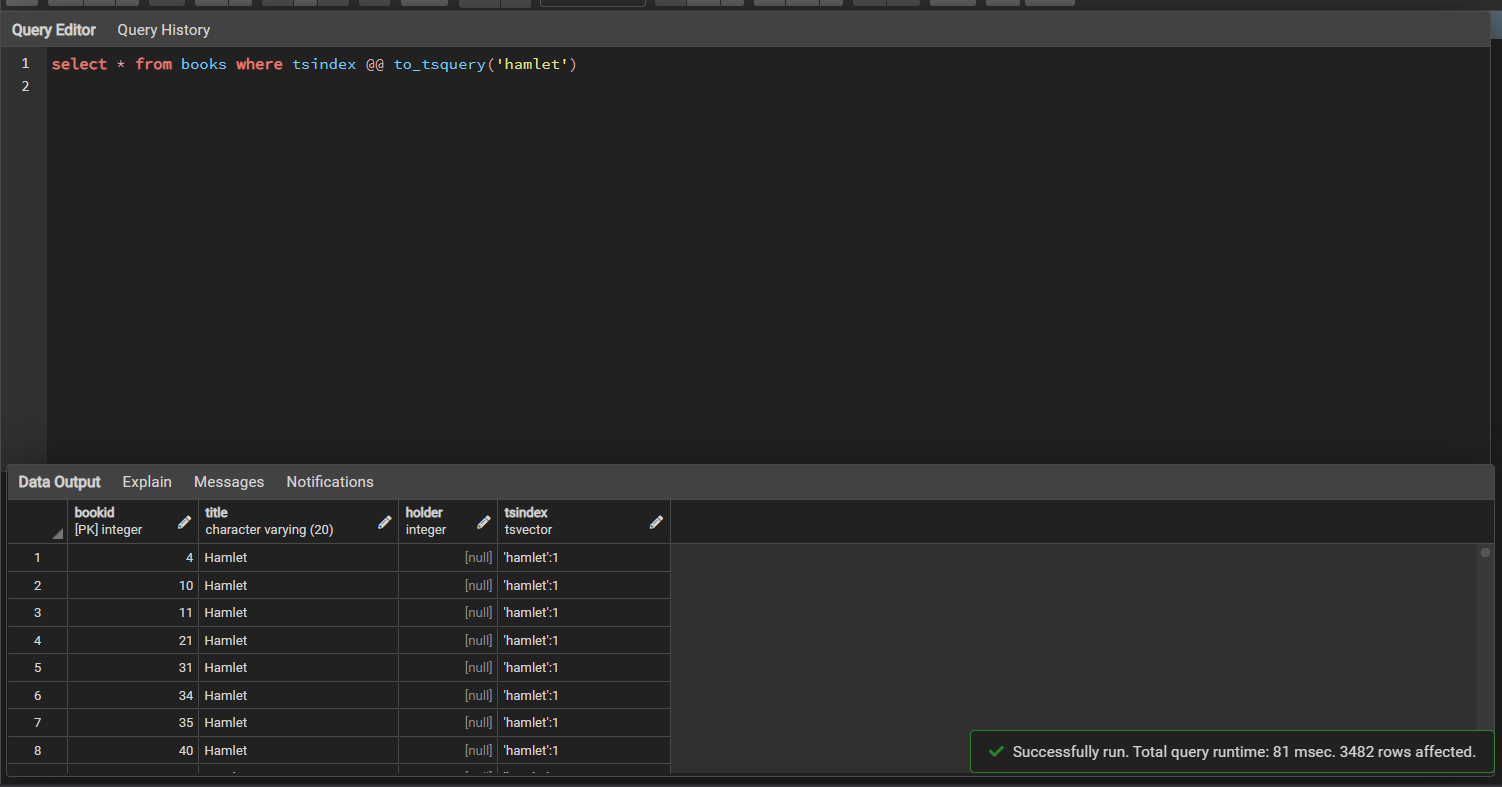
Додамо до таблиці Books колонку tsindex с типом tsvector та заповнимо його за допомогою функції to\_tsvector, який буде приймати як аргумент значення колонки RevText . Саме на колонці tsindex будемо створювати індекс, але спочатку подивимось на швидкість повнотекстового пошуку без індекса.

Для початку дізнаємось, які лексеми найчастіше зустрічаються, а які ні.

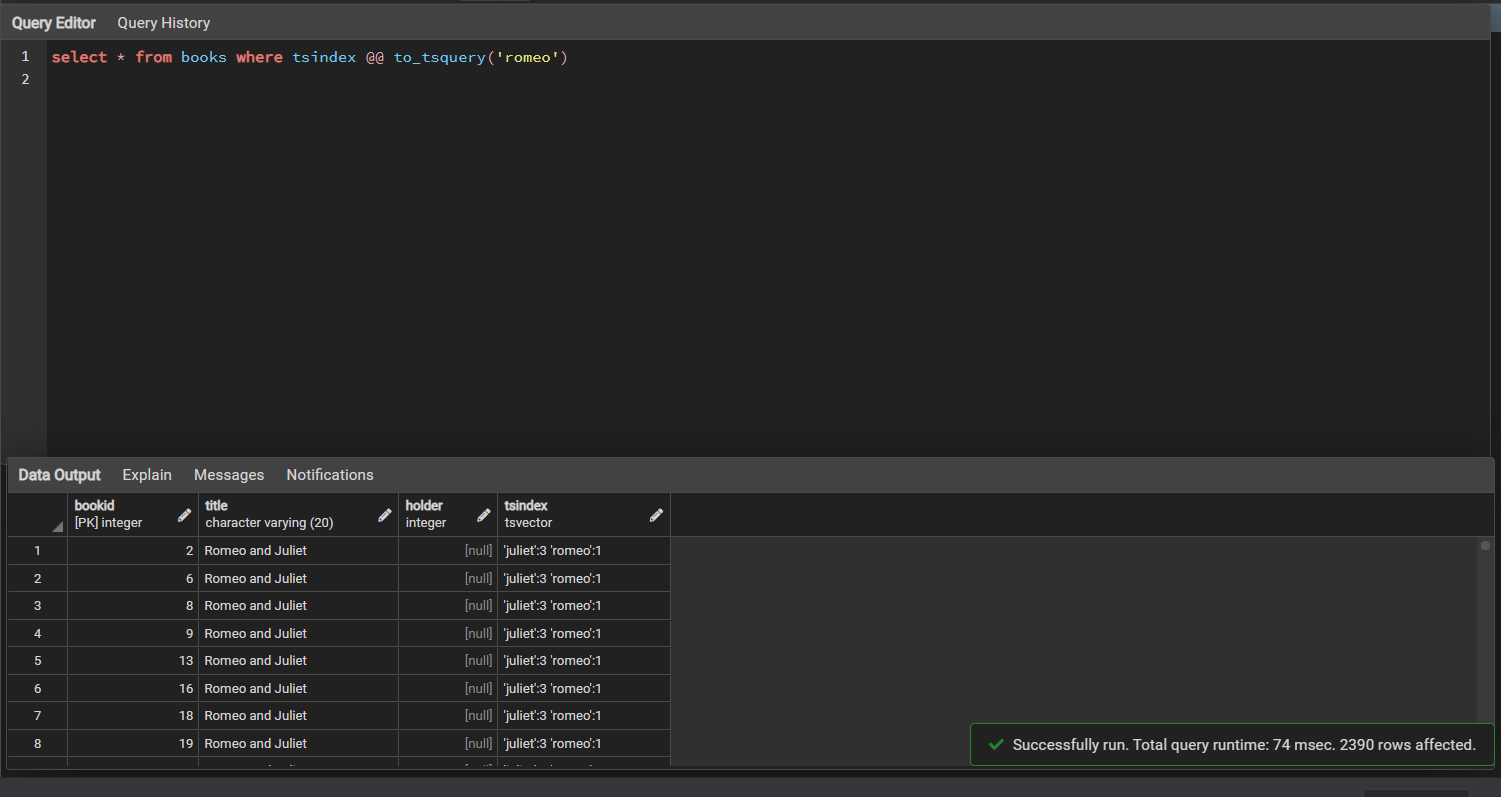


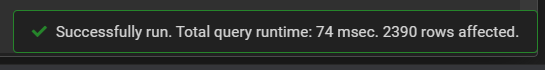
Бачимо, що найчастіше зустрічається “hamlet” та “juliet”, а рідше всього зустрічається “shine” та “war”.

Тепер виконаємо пошук по цих словах без індексу.



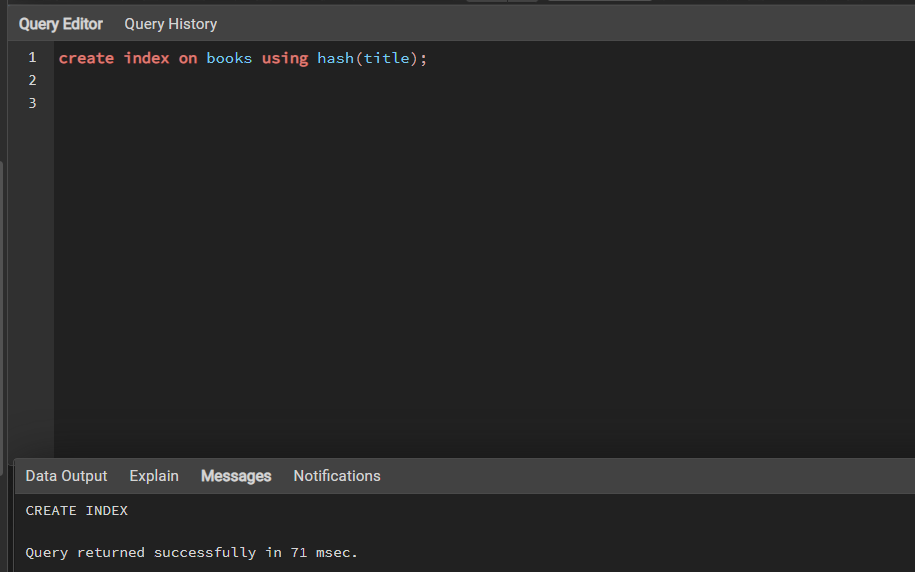






**Створення та аналіз індексу HASH**

Hash індекс застосовується для порівняння / побудови індексів малих і / або двійкових даних. Кожному значенню індексованого вираження зіставляється значення певної хеш функції відображає початкове значення на ціле число (іноді на рядок).



І проведемо такі самі виміри з пошуком “Hamlet” в таблиці.

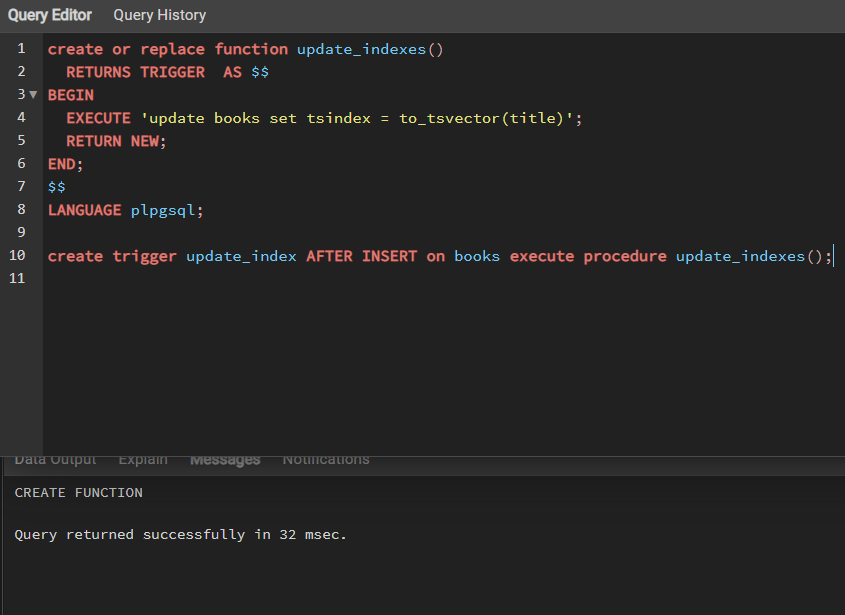
****

****

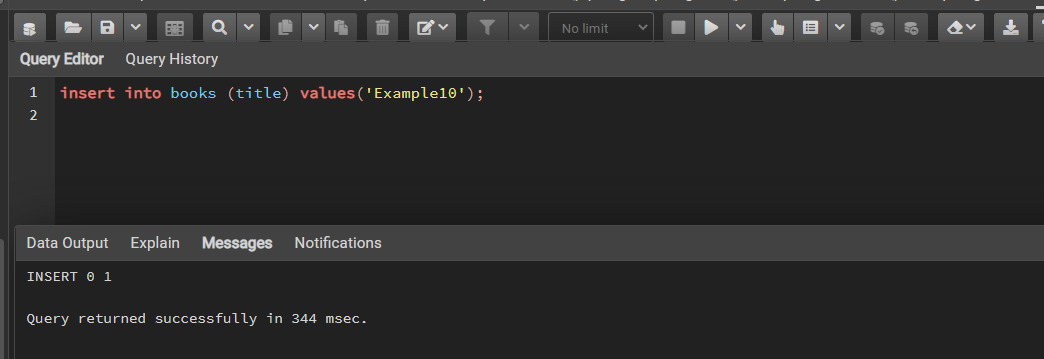
В результаті ми також отримали збільшення швидкодії. Оскільки в нас дуже багато однотипної інформації, то іде дуже багато часу на її виведення. Теоретично якби ми шукали 2-3 рядки то індексація могла б пришвидшити пошук в десятки разів

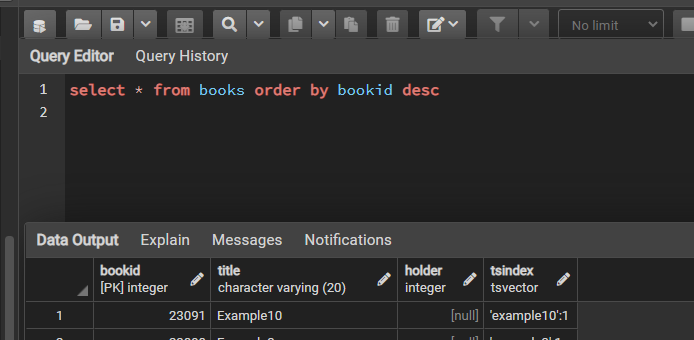
**Завдання 3**

Команда створення тригеру



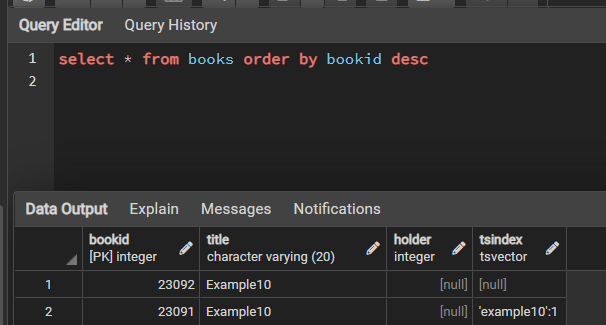
Даний тригер працює з операціями вставки після безпосередньої вставки. Коли ми записуємо щось в базу даних, то тригер автоматично добавляє gin індекс до елемента.





Якщо ми перевіримо нашу таблицю, то побачимо що у нас вставився запис разом з полем tsindex.

Давайте попробуємо видалити тригер і ввести якісь дані:



Видаливши тригер і повторивши вставку елемента Example10 ми отримали null значення в полі tsindex. Це ще раз доказує, що тригер повністю робочий.