

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

Алексеєв Михайло

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 “Модель” зробленої 2-ої лабораторної роботи з шаблону MVC у ORM.
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

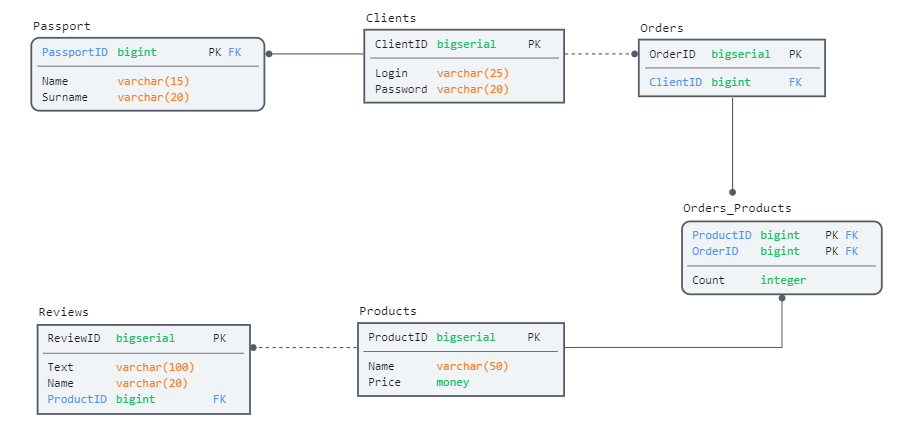
**Варіант 2 (номер залікової 8302)**

У другому завданні проаналізувати індекси Hash, BRIN.

Умова для тригера – after insert, update

**Завдання 1**

Логічна схема бази даних “Інтернет магазин”



Для того щоб перетворити модулю “Model”, який був розроблений у минулій лабораторній роботі, з MVC у ORM(об’єктно-реляційної моделі) використовується JavaScript бібліотеку sequelize, яка була створена для цієї задачі.

Ініціалізація моделі для усіх сутностей є однаковою, тому розберемо її на прикладі Сlients:

const { Sequelize, Model, DataTypes } = require('sequelize');

const sequelize = new Sequelize('sqlite::memory:');

class Clients extends Model {}

Clients.init(

    {

        clientid: { //Ініціалізація PK сутності

            type: DataTypes.BIGSERIAL,

            primaryKey: true,

            autoIncrement: true

        },

        login: DataTypes.STRING,

        password: DataTypes.STRING

    },

    {

        sequelize,

        modelName: 'clients',

        primaryKey: false,

        freezeTableName: true,

        timestamps: false

    }

);

Orders.hasOne(Books, { foreignKey: 'clientid', targetKey: 'clientid' }); //Опис FK

module.exports = Clients; //Експорт моделі для майбутнього використання у модулі “Сontroller”

Спочатку ініціалізується клас-сутність Clients, потім описується усі поля, які належать таблиці, а після – зовнішні ключі з іншими таблицями.

В цій ORM уже є готові базові методи для додавання, видалення та виводу даних. Метод вставки у бібліотеці реалізовано таким чином. Так як мова JavaScript написана на мові С++, її реалізація має її схожий синтаксис.

public static create<M extends Model>(

    this: ModelStatic<M>,

    values?: M['\_creationAttributes'],

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

  ): Promise<M>;

Метод видалення даних.

public static destroy<M extends Model>(

    this: ModelStatic<M>,

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

  ): Promise<number>;

Метод змінни даних

public static update<M extends Model>(

    this: ModelStatic<M>,

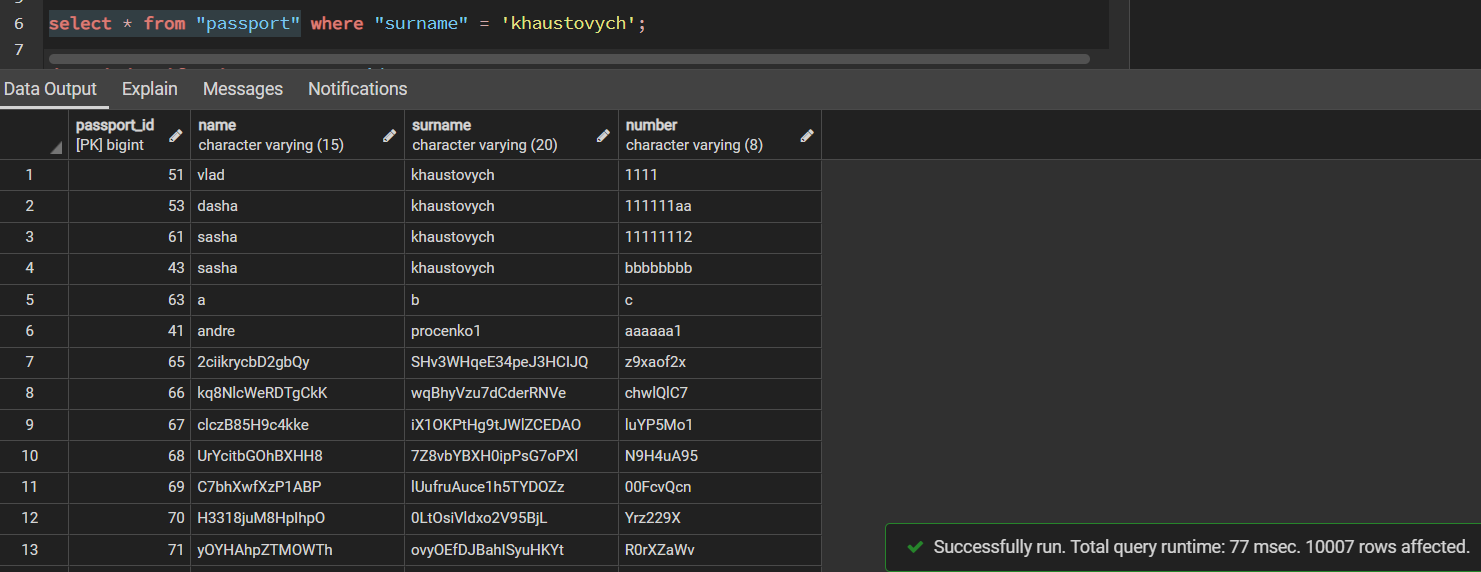
    values: Partial<M['\_attributes']>,

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

  ): Promise<[number, M[]]>;

**Завдання 2**

Для цього завдання створимо 100000 випадкових рядків в таблиці passport(name, surname, number).

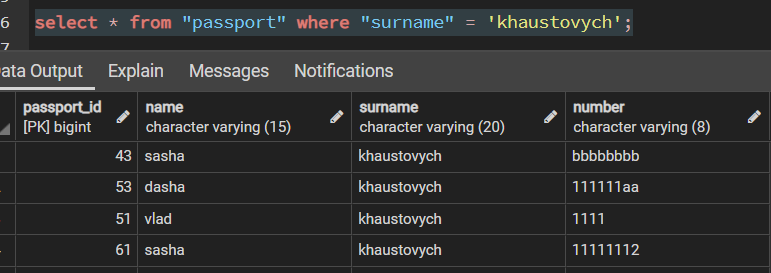


**Створення та аналіз індексу HASH**

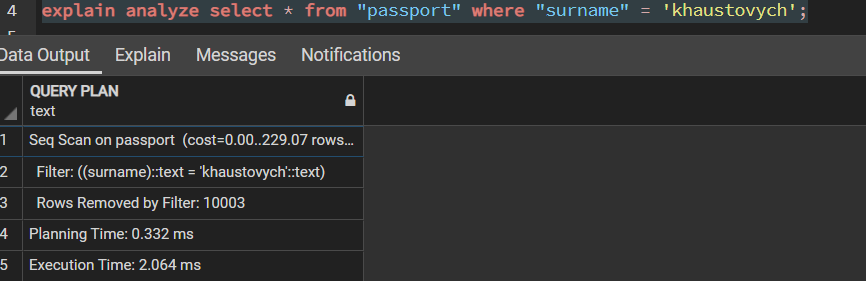
**Hash-індекси** були запропоновані Артуром Фуллером, і припускають зберігання не самих значень, а їх хеш, завдяки чому зменшується розмір (а, відповідно, і збільшується швидкість їх обробки) індексів з великих полів. Таким чином, при запитах з використанням HASH-індексів, порівнюватися будуть не шукане зі значення поля, а хеш від шуканого значення з хешамі полів.

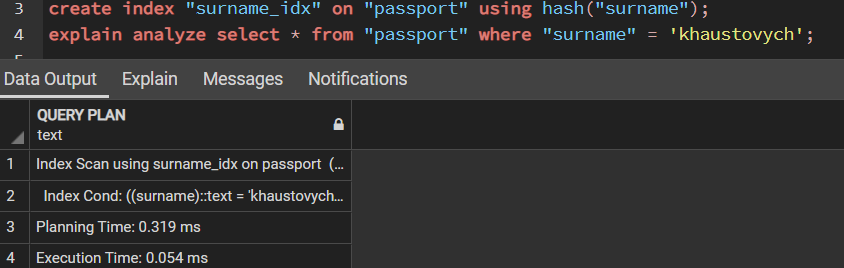
Через нелінейнойсті хеш-функцій даний індекс не можна сортувати за значенням, що призводить до неможливості використання в порівняннях більше / менше і «is null». Крім того, так як хеші не унікальні, то для співпадаючих хеш застосовуються методи вирішення колізій.

Проведемо пошук surname=“ khaustovych ” в таблиці passport

****



Проведемо аналіз використання запиту без індексу  


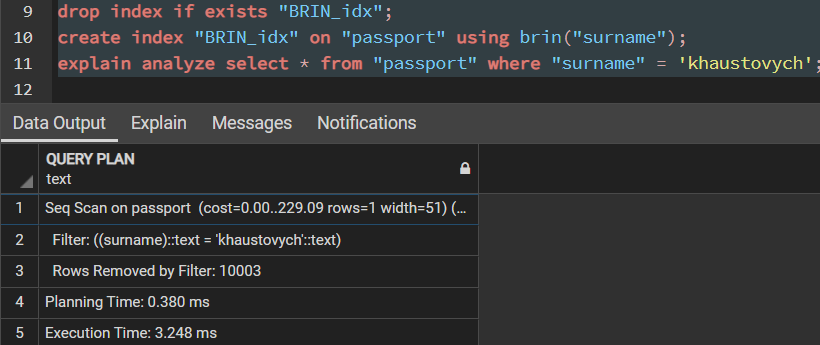
Проведемо аналіз використання запиту з індексом Hash  


Бачимо, що після створення Hash-індексу, видно вже покращення результату. Тобто застосування індексу HASH покращило роботу повнотекстового пошуку.

**Створення та аналіз індексу BRIN**

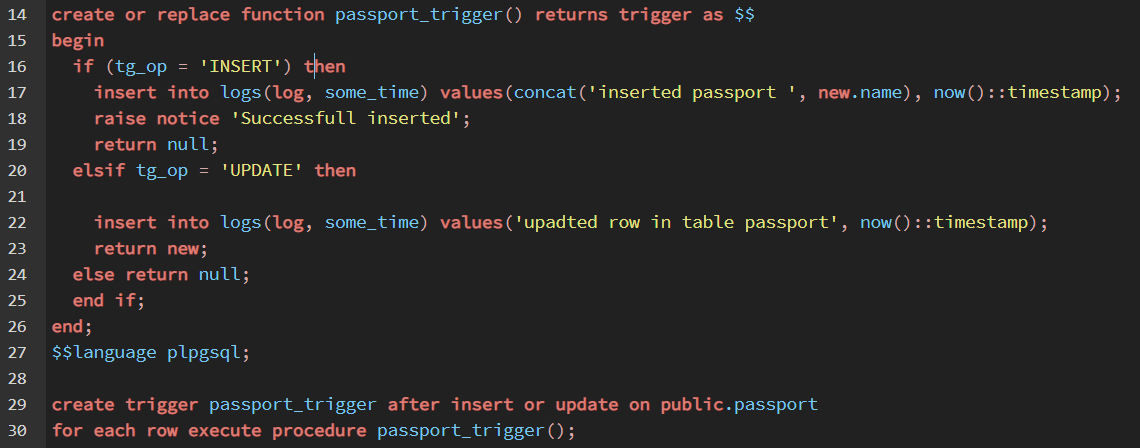
BRIN - це  Block Range Index.  Призначена для обробки дуже великих таблиць, в яких певні стовпці мають деяку природну кореляцію з їх фізичним розташуванням в таблиці, головна концепція якого не знаходження необхідного значення, а уникнення перегляду свідомо непотрібних. Він працює добре для тих стовпчиків, де значення корелюють із їх фізичним положенням в таблиці. Тобто, якщо запит без ORDER BY видає значення стовпчика практично в порядку зростання чи спадання.

Аналогічно створюємо індекс та робимо запит для surname=“ khaustovych “.



**Завдання 3**

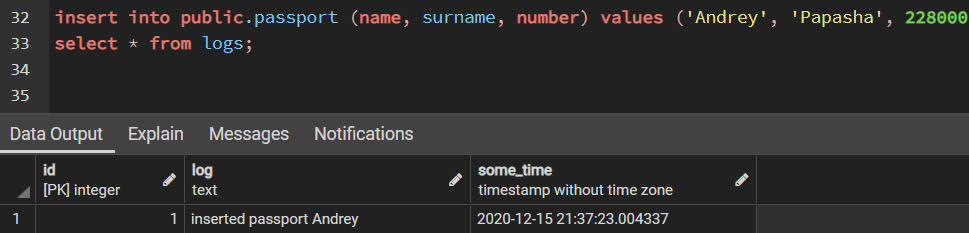
Умова для тригера – after insert, update. Для цього вибрав таблицю passport. Тригер, при видаленні рядка записує інформацію в таблицю logs.



Після того, як буде виконано INSERT-запит в таблицю logs записується повідомлення про те, який запис додали до таблиці та час коли це було зроблено.

При виконані UPDATE-запит тригера у таблицю logs, також, додається запис з відповідним повідомленням.

Розглянемо процес додавання запису.



А також зміни вже існуючих даних.

