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 III курсу

ФПМ групи КВ-13

Гармаш Д.О.

Перевірив:

Київ – 2023

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

*Завдання* роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC РГР у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.
4. Навести приклади та проаналізувати рівні ізоляції транзакцій у PostgreSQL.

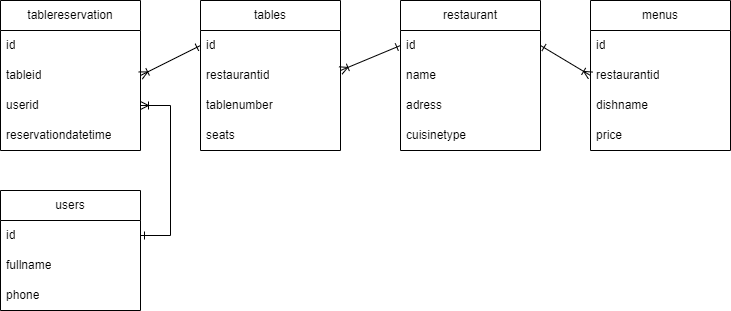
Варіант 3

У другому завданні проаналізувати індекси GIN, Hash.

Умова для тригера – before delete, update.

**Завдання 1**

Модель бази даних:



Для перетворення модулів моделей програми, створених в розрахунковій роботі, у вигляд об’єктно-реляційної моделі було використано Entity Framework Core.

public class Menu

{

public int id { get; set; }

public int? restaurantid { get; set; }

[ForeignKey("restaurantid")]

public Restaurant Restaurant { get; set; }

public string dishname { get; set; }

public int price { get; set; }

}

public class Restaurant

{

public int id { get; set; }

public string name { get; set; }

public string adress { get; set; }

public string cuisinetype { get; set; }

}

public class Table

{

public int id { get; set; }

public int? restaurantid { get; set; }

[ForeignKey("restaurantid")]

public Restaurant Restaurant { get; set; }

public int tablenumber { get; set; }

public int seats { get; set; }

}

public class TableReservation

{

public int id { get; set; }

public int? tableid { get; set; }

[ForeignKey("tableid")]

public Table Table { get; set; }

public int? userid { get; set; }

[ForeignKey("userid")]

public User User { get; set; }

public DateTime reservationdatetime { get; set; }

}

public class User

{

public int id { get; set; }

public string fullname { get; set; }

public string phone { get; set; }

}

Налаштуння зв’язку з таблицями:

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {

optionsBuilder.UseNpgsql(ConnectionString);

}

protected override void OnModelCreating(ModelBuilder modelBuilder) {

modelBuilder.Entity<Menu>().Navigation(o => o.Restaurant).AutoInclude();

modelBuilder.Entity<TableReservation>().Navigation(o => o.User).AutoInclude();

modelBuilder.Entity<TableReservation>().Navigation(o => o.Table).AutoInclude();

modelBuilder.Entity<Table>().Navigation(o => o.Restaurant).AutoInclude();

}

Забезпечення доступу до сутностей:

DbSet<Menu> menus { get; set; }

DbSet<Table> tables { get; set; }

DbSet<Restaurant> restaurants { get; set; }

DbSet<User> users { get; set; }

DbSet<TableReservation> tablereservations { get; set; }

Реалізація вставки:

public void AddRestaurant(Restaurant r) {

restaurants.Add(r);

SaveChanges();

}

Реалізація редагування:

public void EditRestaurant(int id, Restaurant r) {

// Оновлення існуючого ресторану в базі даних за ID

Restaurant er = restaurants.Single(o => o.id == id);

er.adress = r.adress;

er.cuisinetype = r.cuisinetype;

er.name = r.name;

SaveChanges();

}

Реалізація видалення:

public void DeleteTableReservation(int id) {

// Видалення бронювання столика з бази даних за ID

TableReservation tr = tablereservations.Single(o => o.id == id);

tablereservations.Remove(tr);

SaveChanges();

}

**Завдання 2**

Створення таблиці:

create table text1(

body text

);

Вставка та генерація даних:

insert into text1

Select

md5(random()::text)

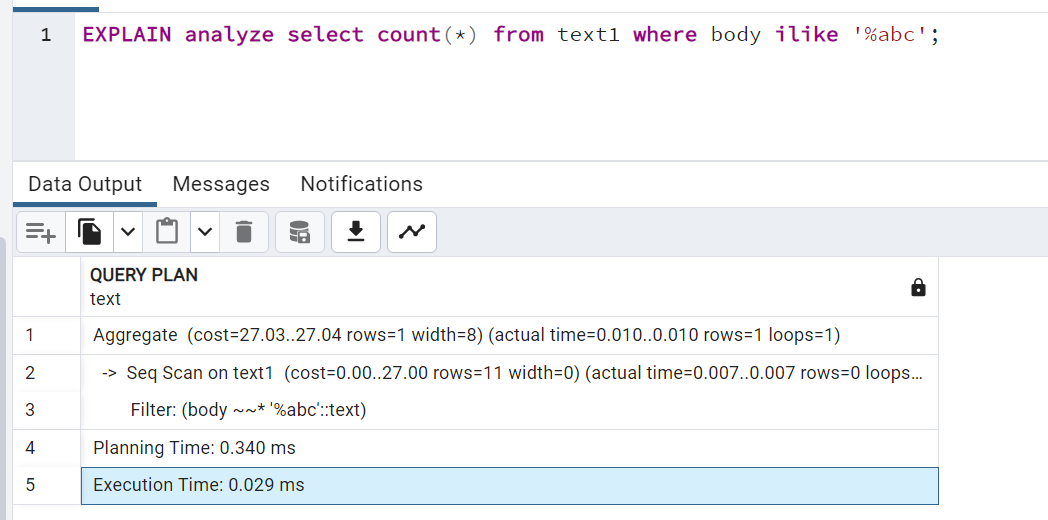
from(select \* from generate\_series(1, 100000) as id) as x;



GIN

Час виконання без індексу:

EXPLAIN analyze select count(\*) from text1 where body ilike '%abc';

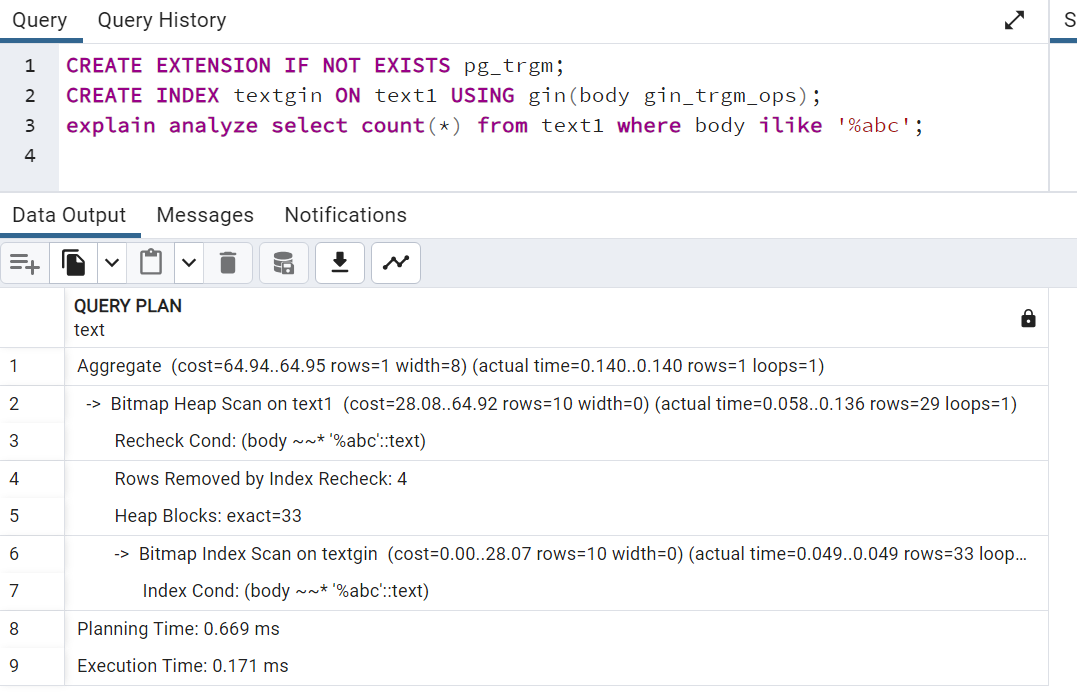


Час виконання з використанням індексу:

CREATE EXTENSION IF NOT EXISTS pg\_trgm;

CREATE INDEX textgin ON text1 USING gin(body gin\_trgm\_ops);

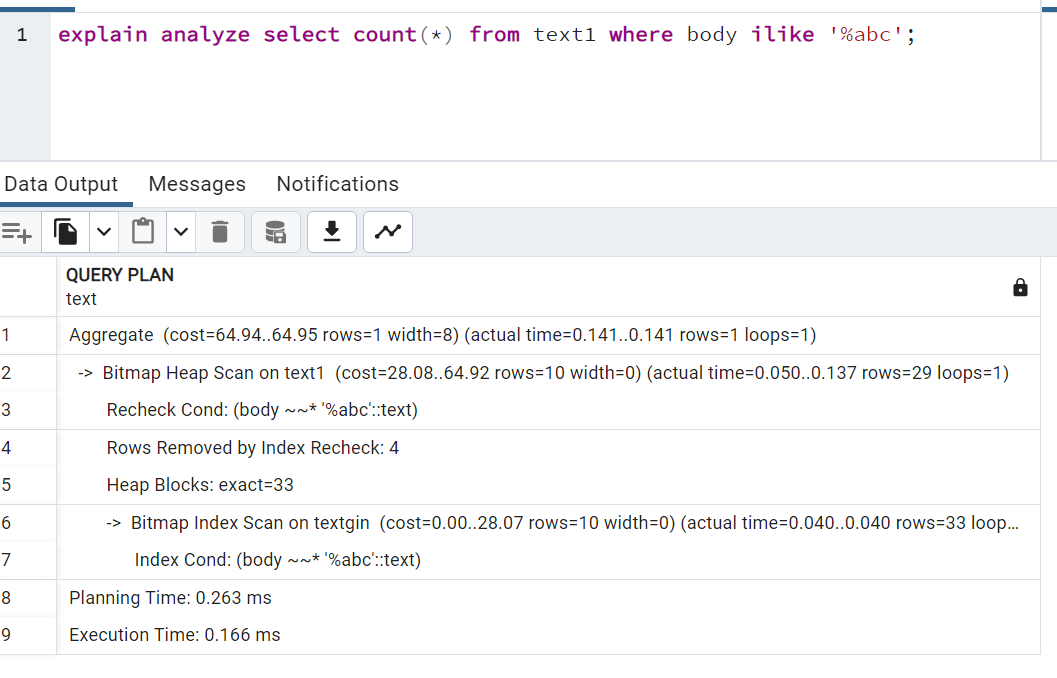
explain analyze select count(\*) from text1 where body ilike '%abc';



Індекс GIN призначений для швидкого пошуку, але його ефективність може бути обмеженою при великому об'ємі однотипних даних або запитах, які не використовують індексовані стовпці. Неправильна конфігурація запитів або часті зміни даних можуть призвести до швидкого застаріння індексу, що зменшить його ефективність. Для оптимізації використання індексів слід ретельно переглянути структуру даних та оптимізувати запити для використання індексованих полів.

Hash

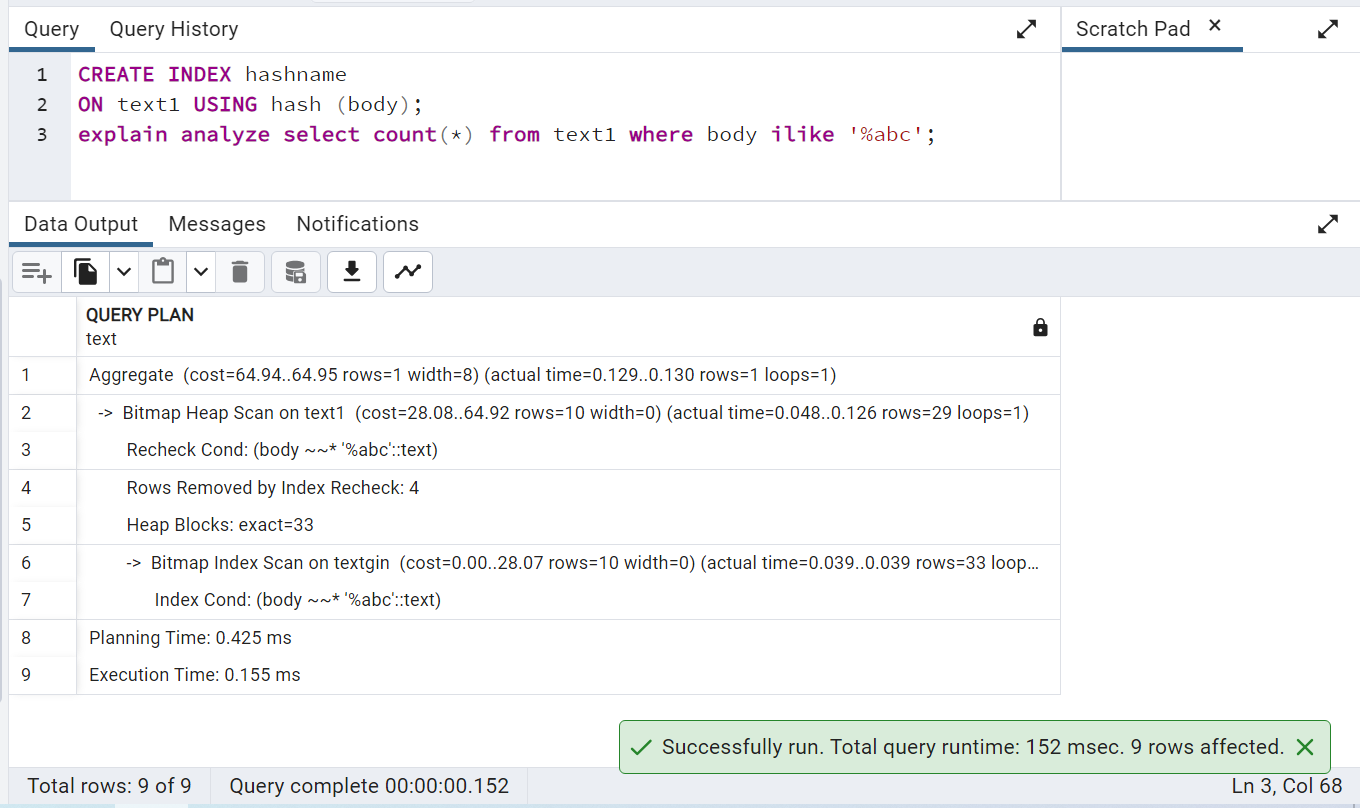
Час виконання без індексу:



Час виконання з використанням індексу:

ON text1 USING hash (body);

explain analyze select count(\*) from text1 where body ilike '%abc';



У даному випадку не спостерігається прискорення завдяки використанню індексу HASH, бо мала різниця у часі виконання (0.166 ms без індексу та 0.155 ms з індексом) може бути нечуттєвою для вимірювання. Також незначна різниця у часі виконання може бути пов'язана з регулярним оновленням індексу HASH, бо зазвичай якщо дані не великі або якщо, наприклад, обсяг оперативної пам'яті або конфігурація сервера також впливають на швидкість виконання, то використання індексу HASH може не мати помітного впливу на продуктивність запитів.

**Завдання 3**

*before delete*

**Код тригеру:**

CREATE OR REPLACE FUNCTION updf2()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO "restaurants"("name", "adress", "cuisinetype")

values('newname', 'newadress', 'newcuisinetype');

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

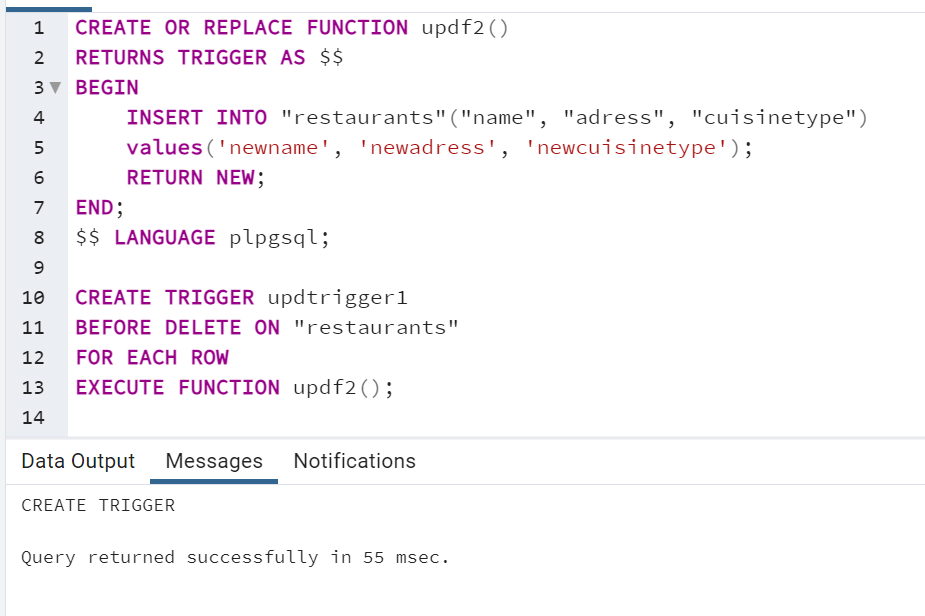
CREATE TRIGGER updtrigger1

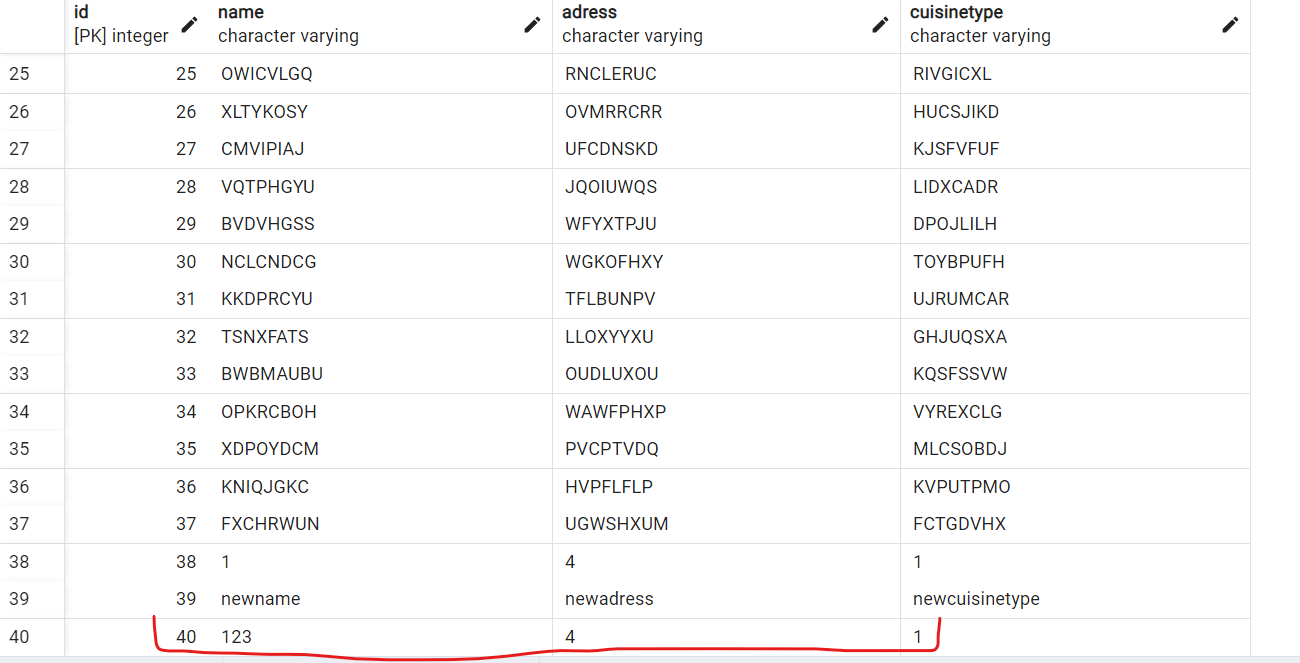
BEFORE DELETE ON "restaurants"

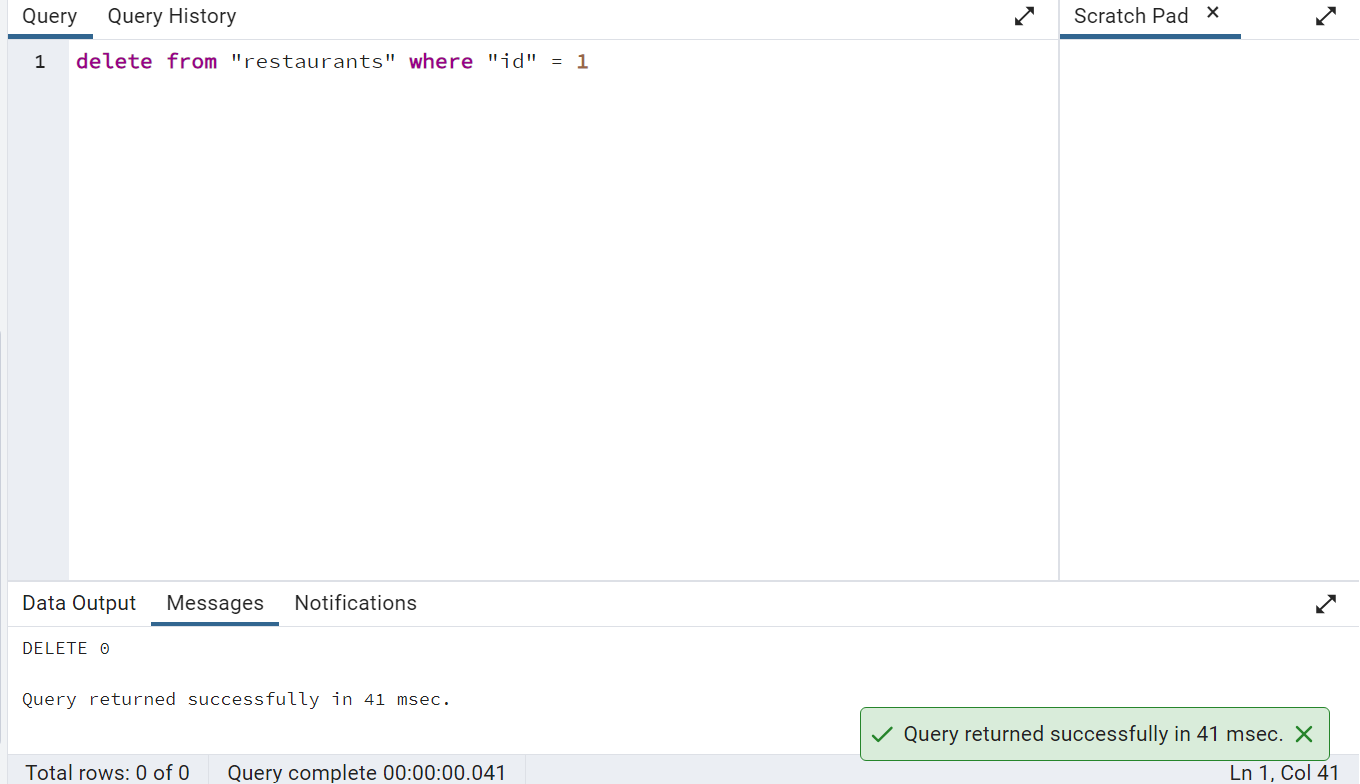
FOR EACH ROW

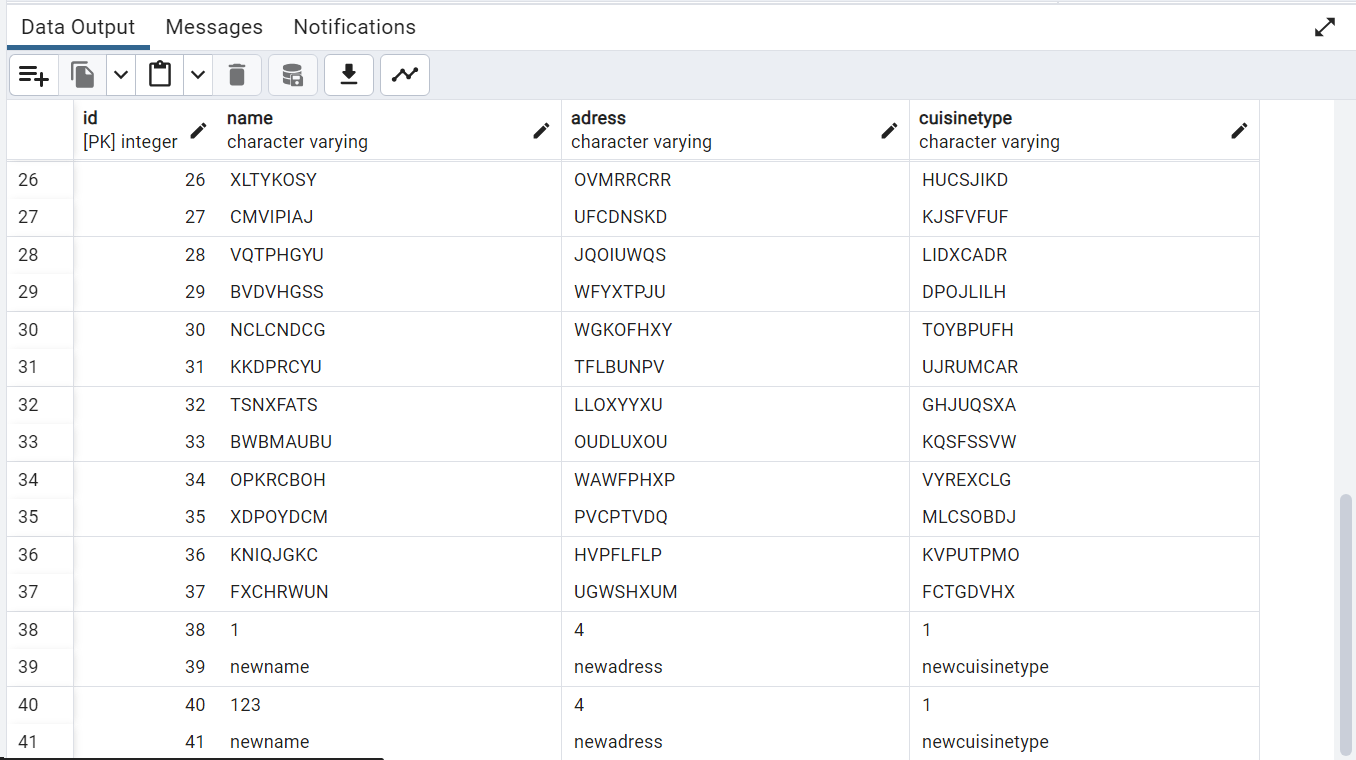
EXECUTE FUNCTION updf2();

**Тестування роботи тригеру:**









*before update*

**Код тригеру:**

CREATE OR REPLACE FUNCTION updf()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO "restaurants"("name", "adress", "cuisinetype")

values(OLD."name", OLD."adress", OLD."cuisinetype");

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

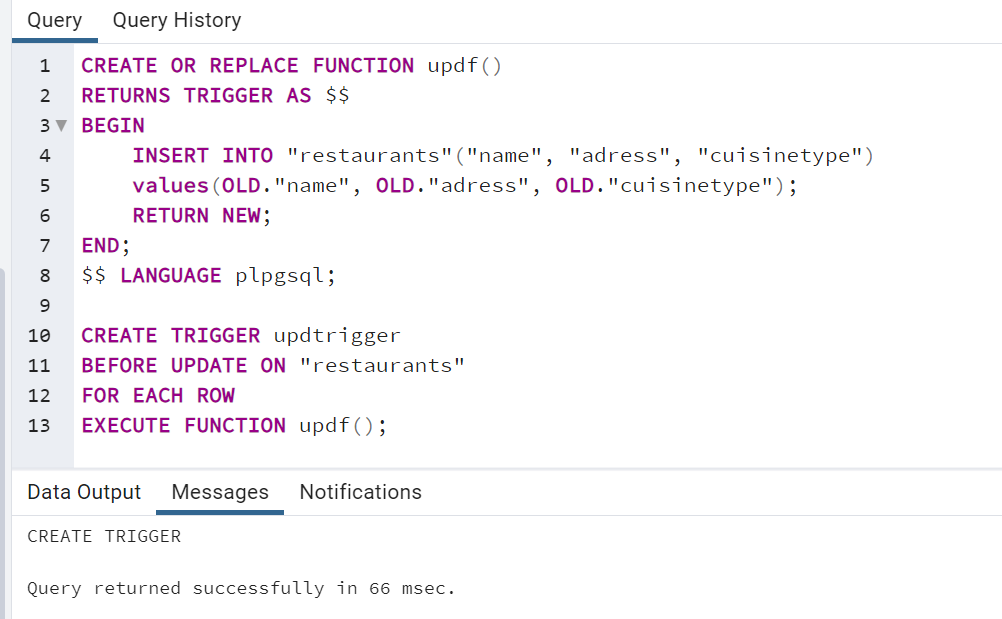
CREATE TRIGGER updtrigger

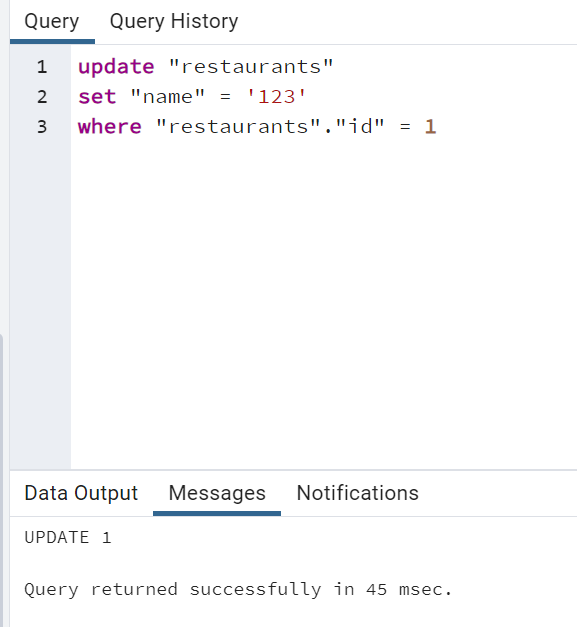
BEFORE UPDATE ON "restaurants"

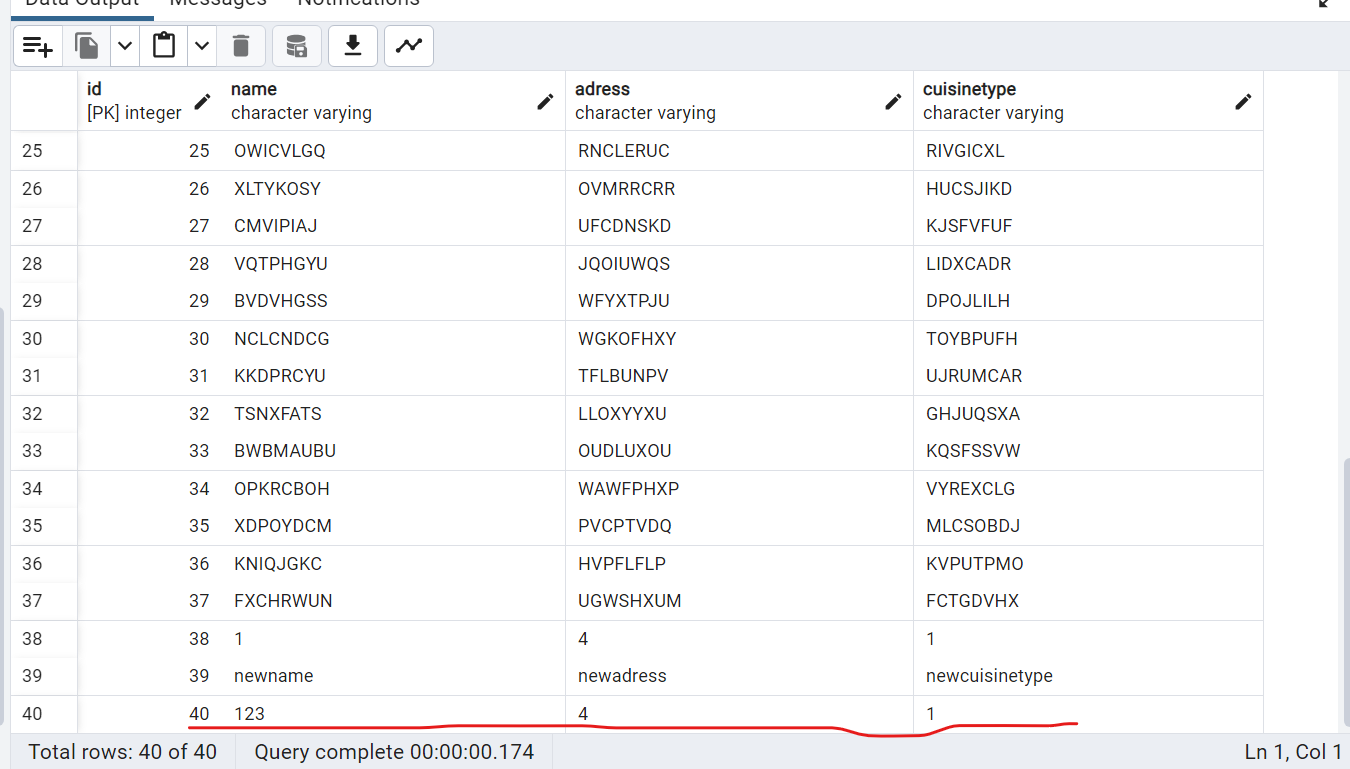
FOR EACH ROW

EXECUTE FUNCTION updf();

**Тестування роботи тригеру:**



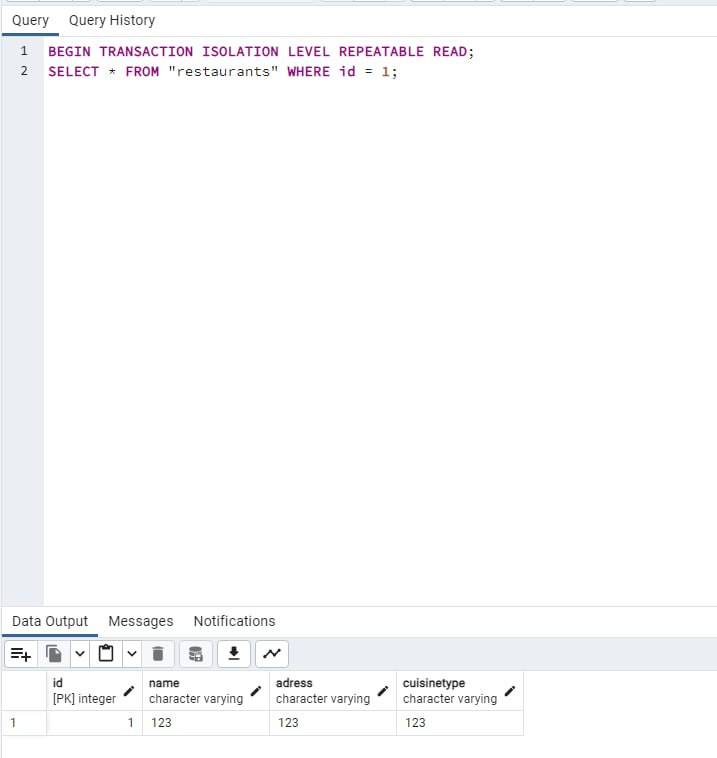


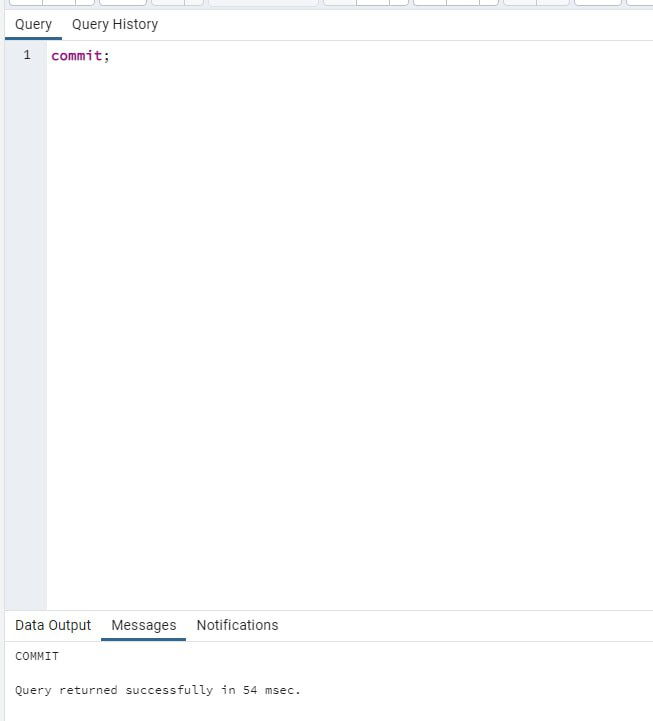


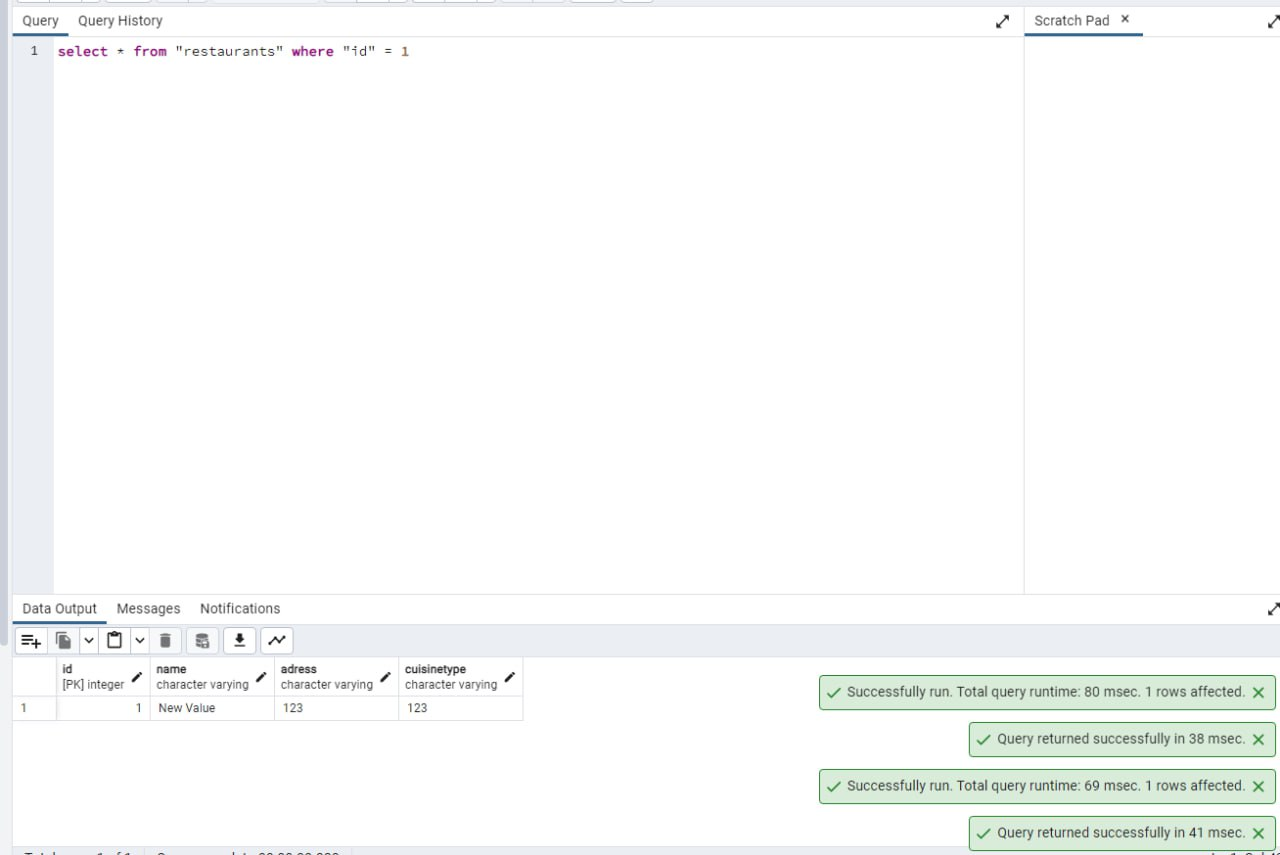
**Завдання 4**

Read Commited vs Repeatable read









REPEATABLE READ vs SERIALIZABLE

