НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних**

**систем**

**Лабораторна робота №2**

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.

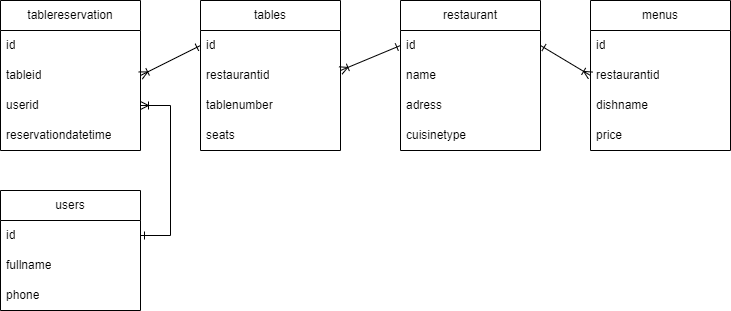
Варіант 3

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)

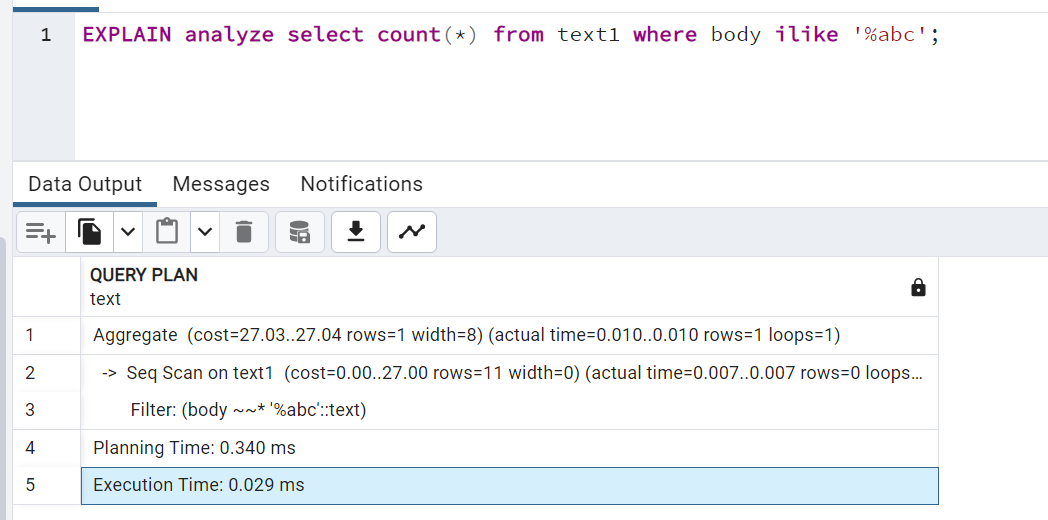
from(select \* from generate\_series(1, 100000) as id) as x;



GIN

Час виконання без індексу:

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

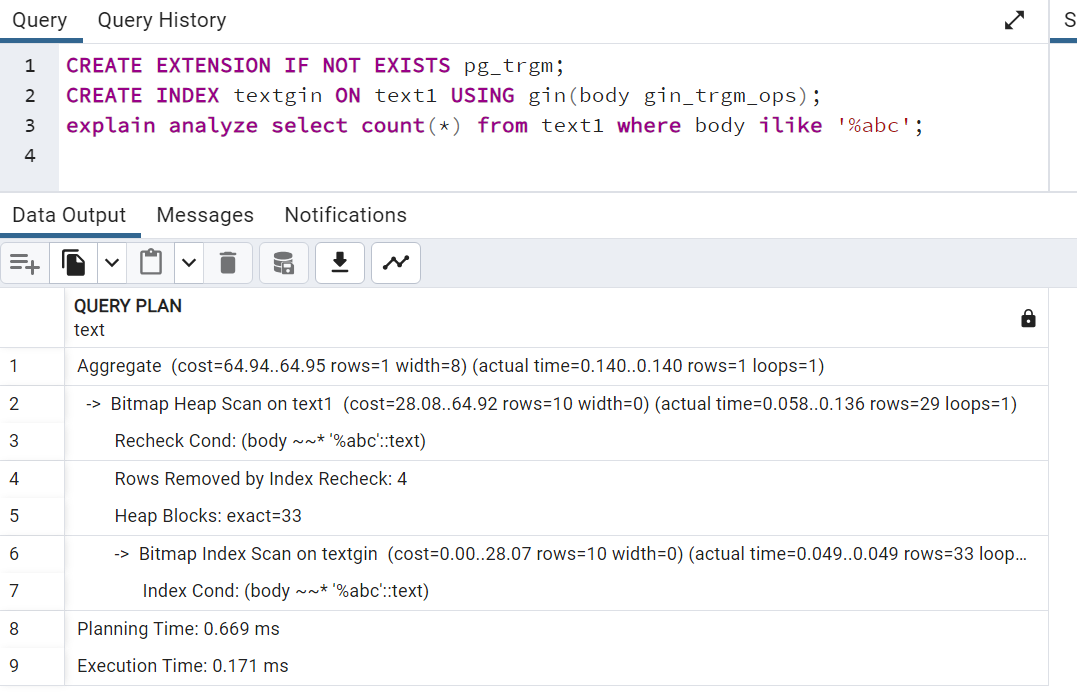


Час виконання з використанням індексу:

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';

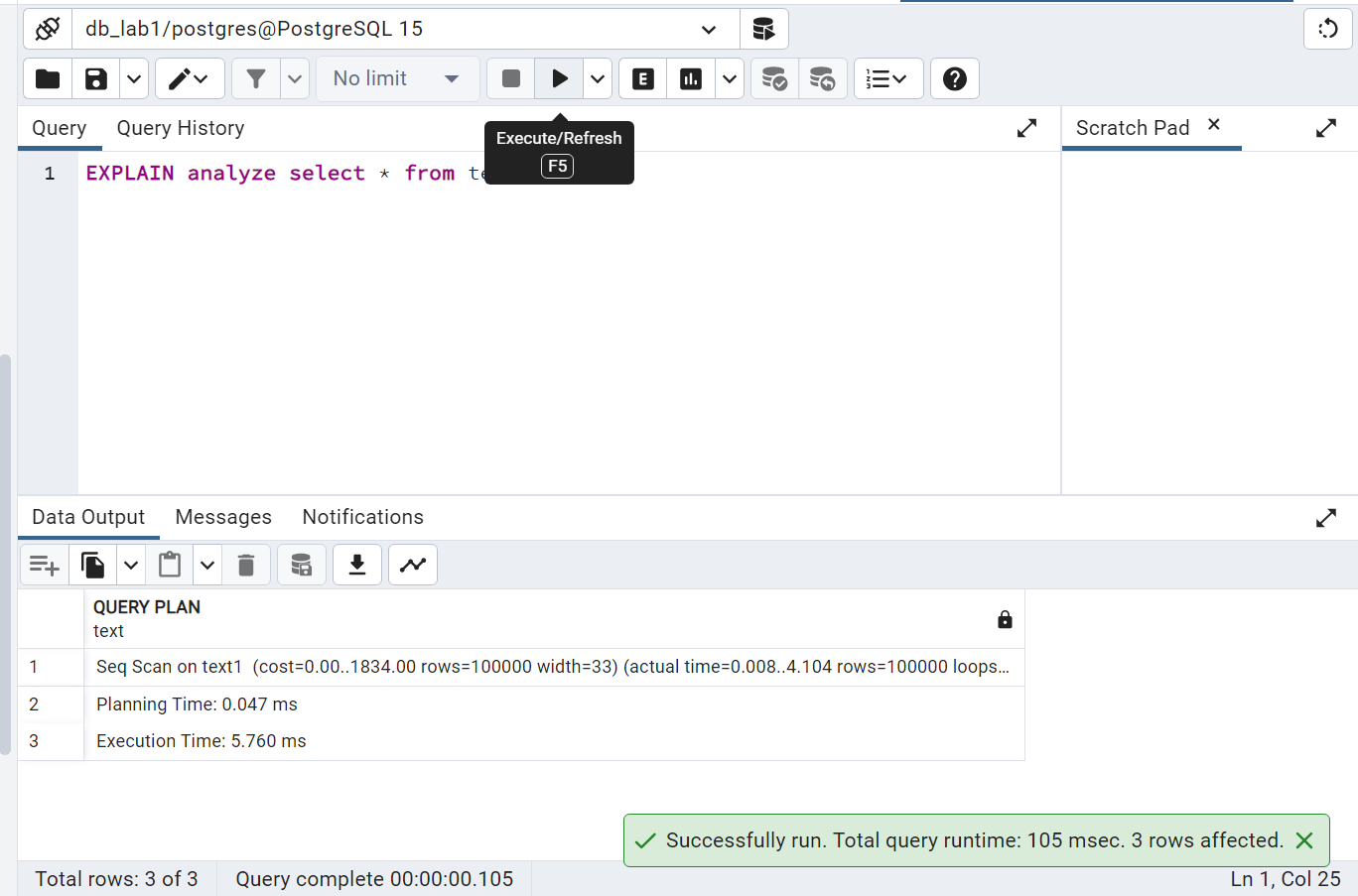


Індекс GIN призначений для швидкого пошуку, але його ефективність може бути обмеженою при великому об'ємі однотипних даних або запитах, які не використовують індексовані стовпці. Неправильна конфігурація запитів або часті зміни даних можуть призвести до швидкого застаріння індексу, що зменшить його ефективність.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

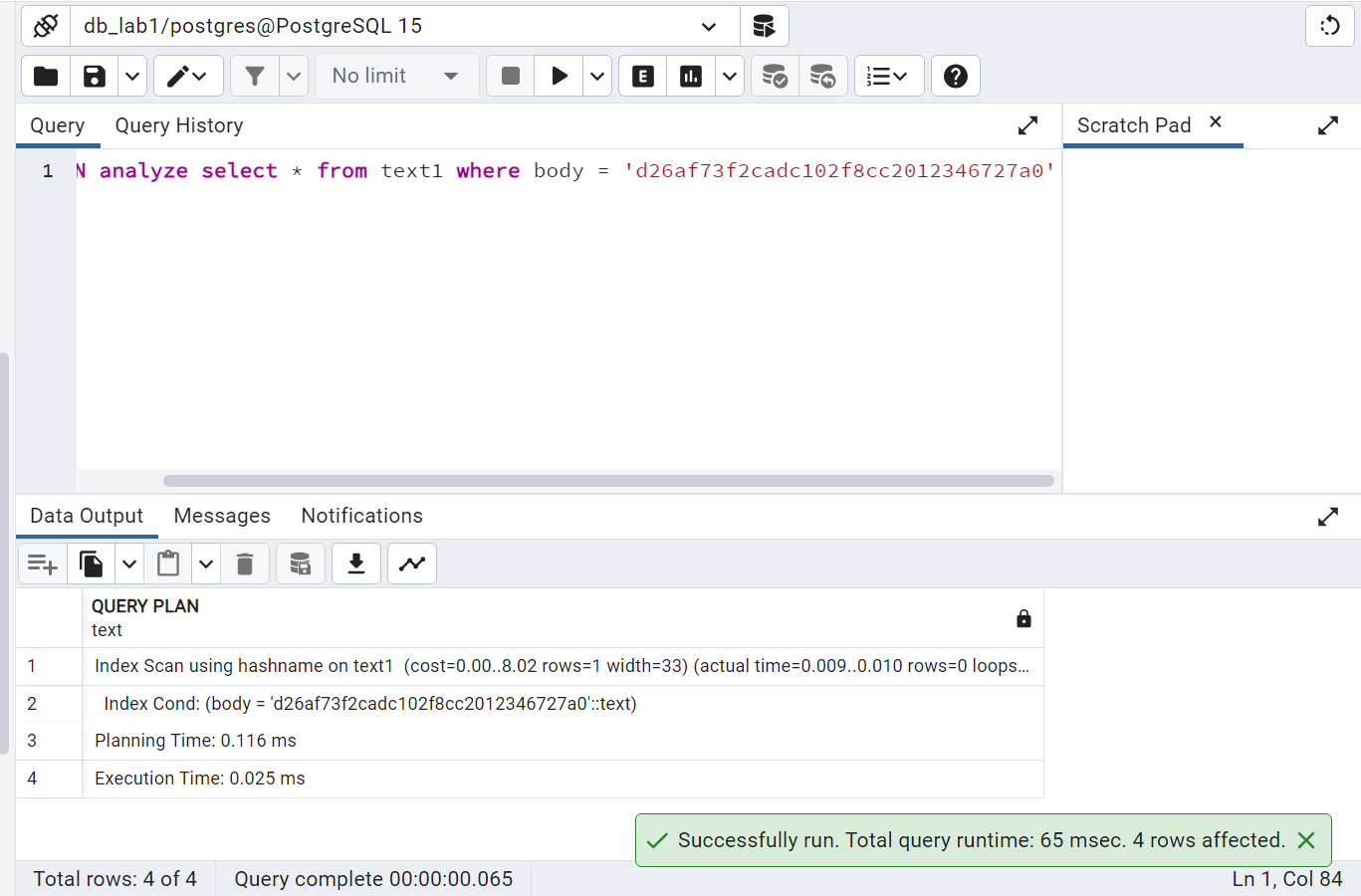
Hash

Час виконання без індексу:

EXPLAIN analyze select \* from text1



Час виконання з використанням індексу:

EXPLAIN analyze select \* from text1 where body = 'd26af73f2cadc102f8cc2012346727a0'

В даному випадку результат *без індексу* (5.760 мс) можна пояснити тим, що база даних повинна пройти через всі записи у таблиці для знаходження необхідної інформації. Це може бути часозатратно, особливо коли таблиця містить велику кількість записів, оскільки потрібно прочитати кожен запис для виконання запиту.

А от при виконанні *з індексом* 0.025 мс індекс дозволяє базі даних швидко знаходити відповідні записи, не проходячи повністю через всі рядки таблиці. Це досягається завдяки структурі даних індексу, що дозволяє ефективно шукати потрібну інформацію за ключем, скорочуючи час доступу до записів.

**Завдання 3**

*before delete*

**Код тригеру:**

CREATE OR REPLACE FUNCTION updf2()

RETURNS TRIGGER AS $$

declare

cursor\_log cursor for select \* from "restaurants";

row\_log "restaurants"%rowtype;

BEGIN

if old."id" % 2 = 0 then

for row\_log in cursor\_log loop

update "restaurants" set "cuisinetype" = 'empty after deleting';

end loop;

end if;

return new;

END;

$$ LANGUAGE plpgsql;

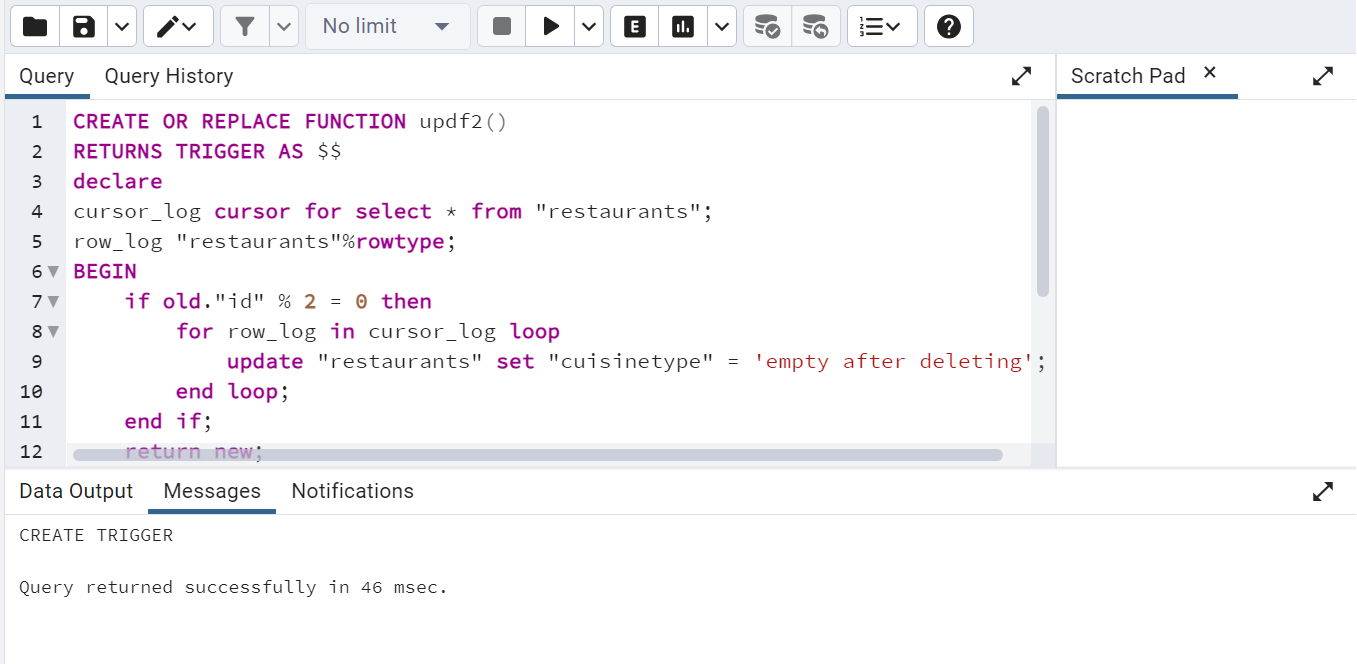
CREATE OR REPLACE TRIGGER updtrigger1

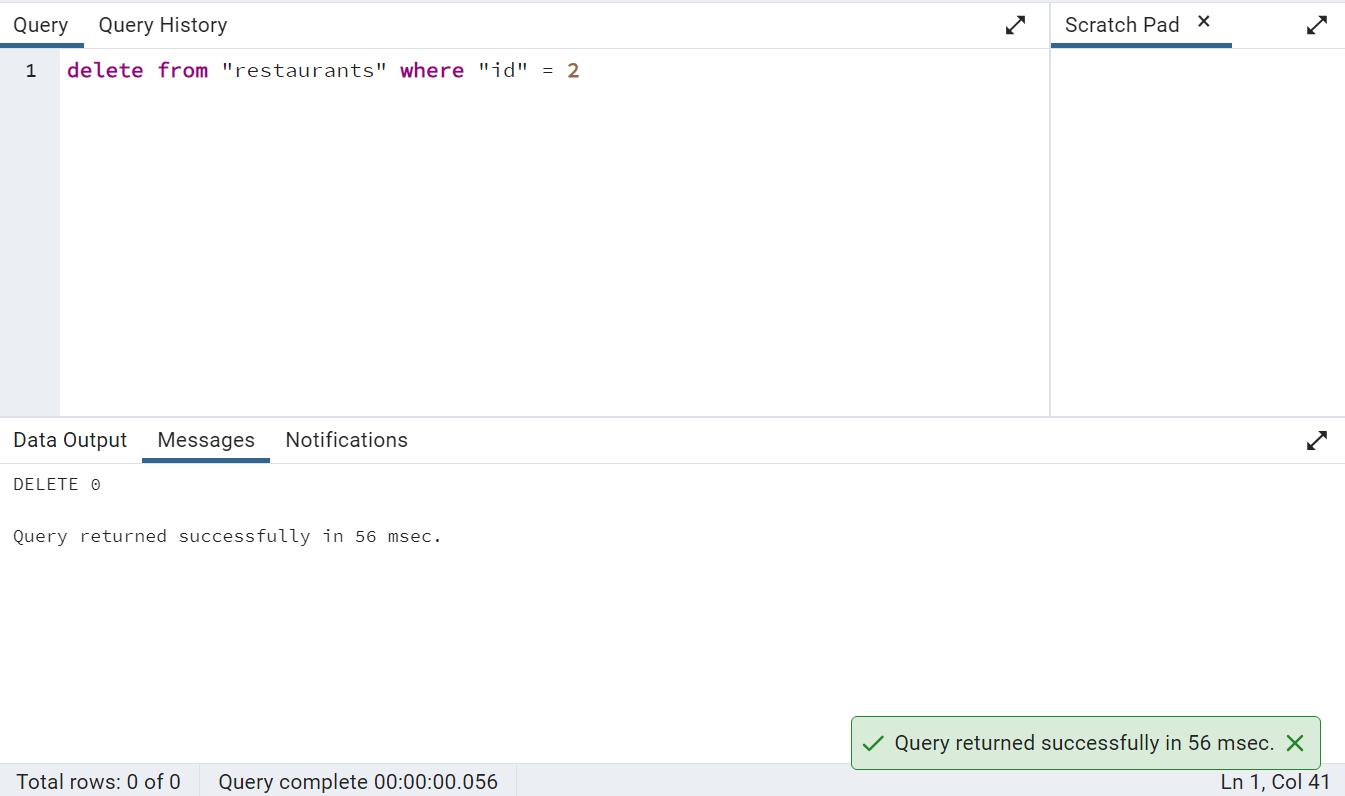
BEFORE DELETE ON "restaurants"

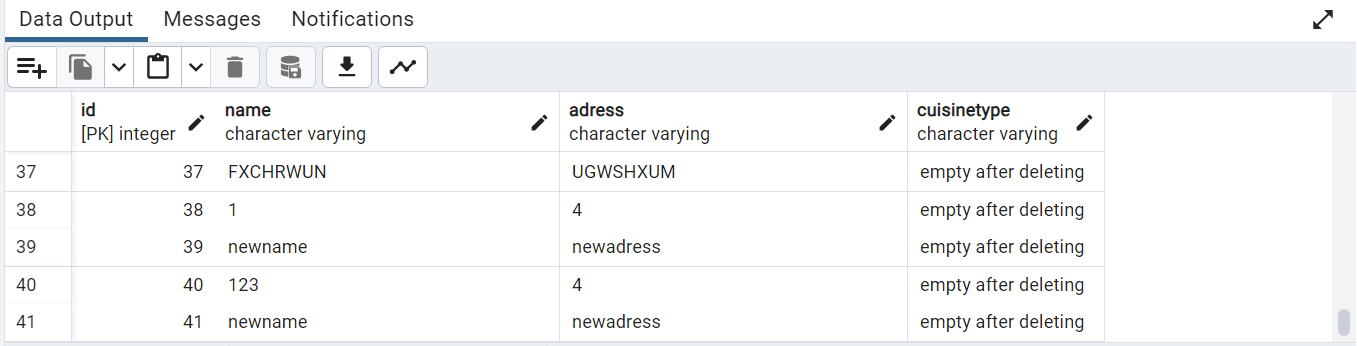
FOR EACH ROW

EXECUTE FUNCTION updf2();

**Тестування роботи тригеру:**







*before update*

**Код тригеру:**

CREATE OR REPLACE FUNCTION updf()

RETURNS TRIGGER AS $$

BEGIN

if old."id" < 1 then

raise notice 'ID is not valid';

return new;

end if;

if old."name" = '' then

UPDATE "new" SET "name" = '123';

end if;

if old."adress" = '' then

UPDATE "new" SET "adress" = '123';

end if;

if old."cuisinetype" = '' then

UPDATE "new" SET "cuisinetype" = '123';

end if;

return new;

END;

$$ LANGUAGE plpgsql;

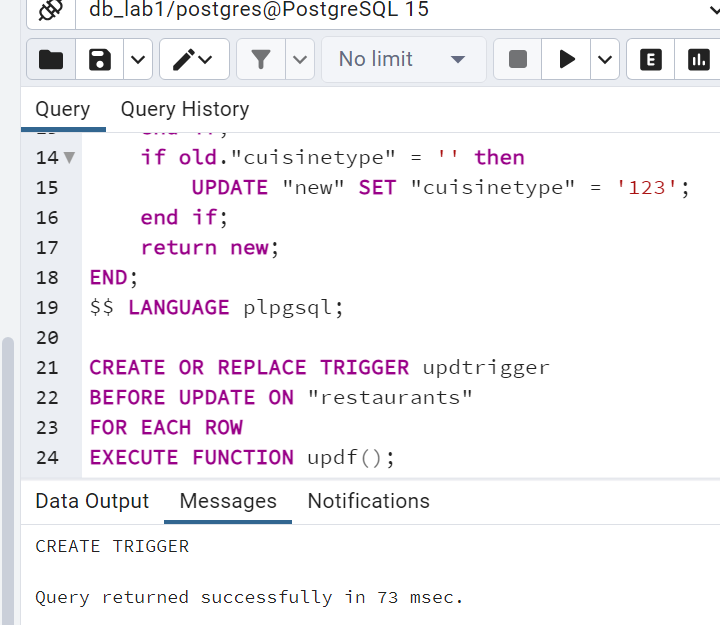
CREATE OR REPLACE TRIGGER updtrigger

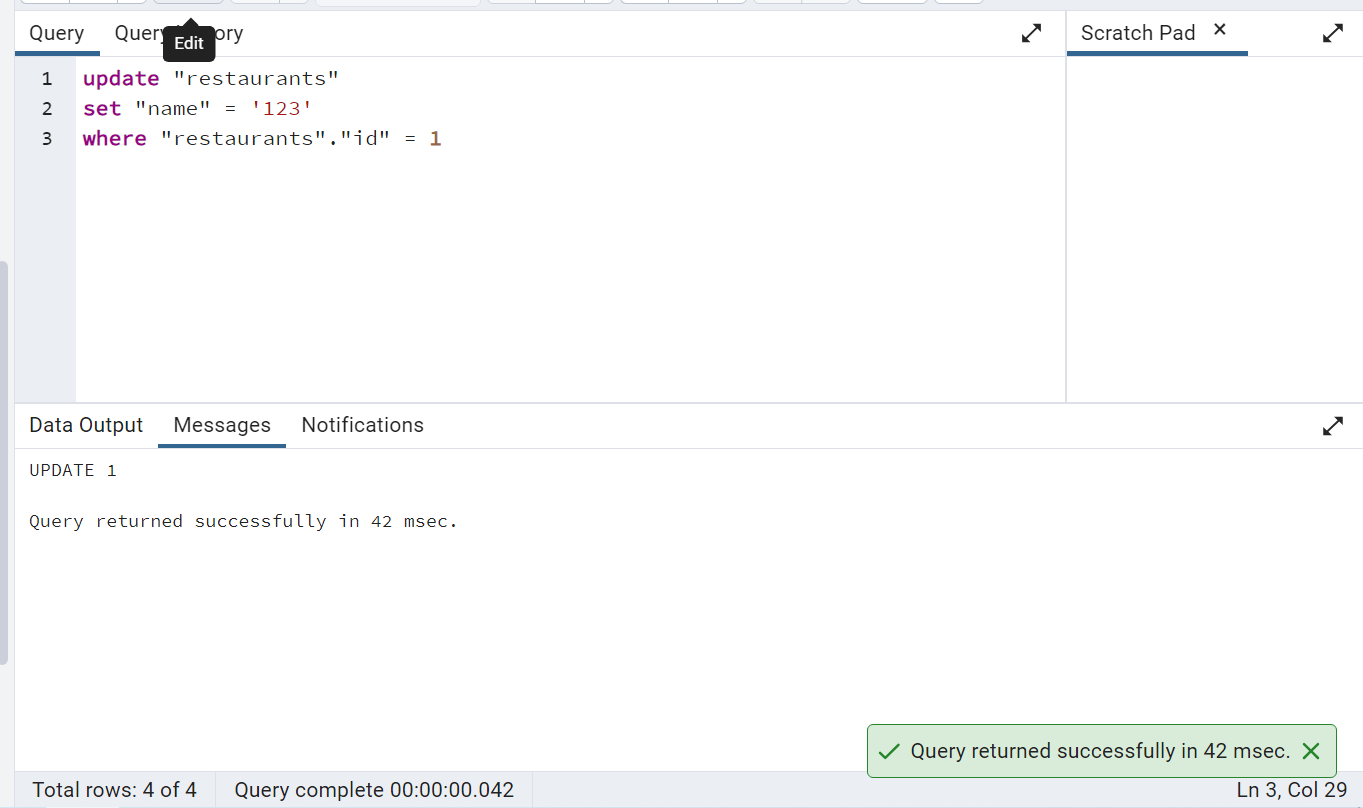
BEFORE UPDATE ON "restaurants"

FOR EACH ROW

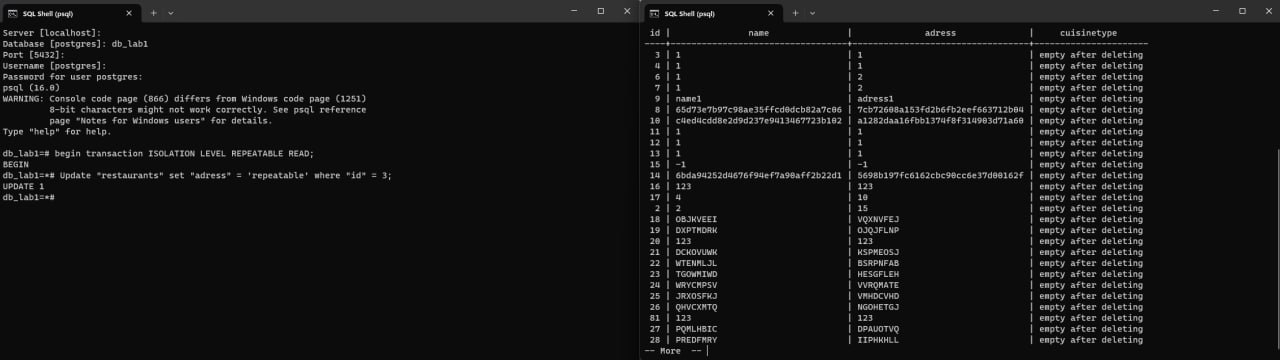
EXECUTE FUNCTION updf();

**Тестування роботи тригеру:**

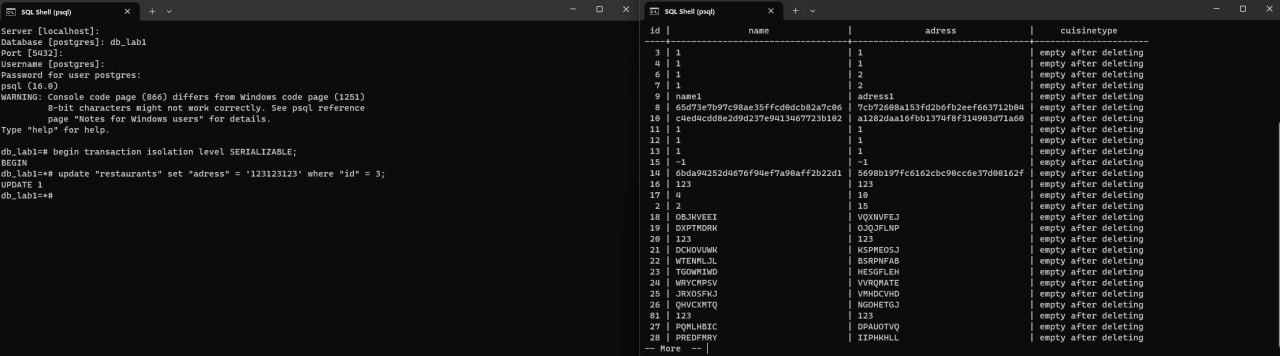




**Завдання 4**

REPEATABLE READ 

SERIALIZABLE



READ COMMITED

