

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ "КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"

Факультет прикладної математики Кафедра програмного забезпечення комп'ютерних систем

Лабораторна робота №3

з дисципліни "Бази даних"

"Засоби оптимізації роботи СУБД PostgreSQL"

Виконав	Зарахована
студент III курсу	""""20p.
групи КП-83	викладачем
Симонюк Володимир Павлович	Радченко К. О.
варіант № 18	

Мета роботи:

Здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL шляхом впровадження ORM (об'єктно-орієнтованої моделі).

Завдання роботи полягає у наступному:

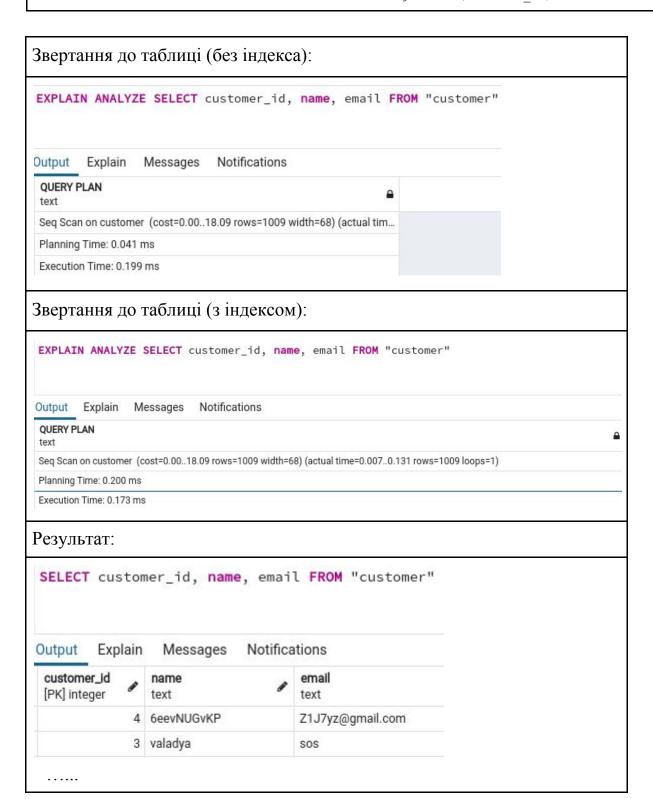
- 1. Перетворити модуль "Модель" з шаблону MVC лабораторної роботи №2 у вигляд об'єктно-реляційної проекції (ORM).
- **2.** Створити та проаналізувати різні типи індексів у PostgreSQL. Індекси за варіантом **BTree**, **GIN**.
- **3.** Розробити тригер бази даних PostgreSQL. Тригери за варіантом **after update**, **insert**.

Хід роботи:

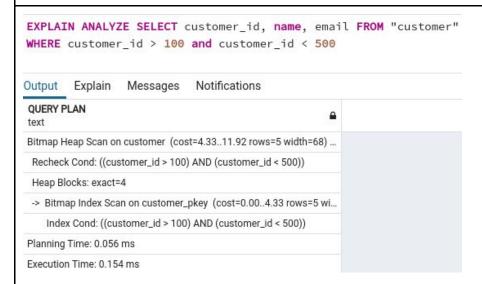
Завдання №2:

BTree

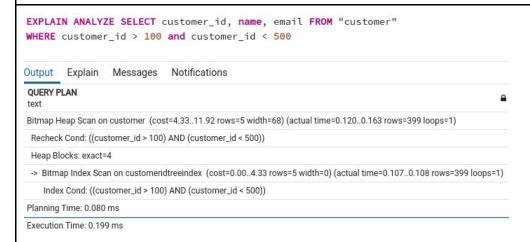
CREATE INDEX customerIdTreeIndex ON "customer" using btree (customer id);



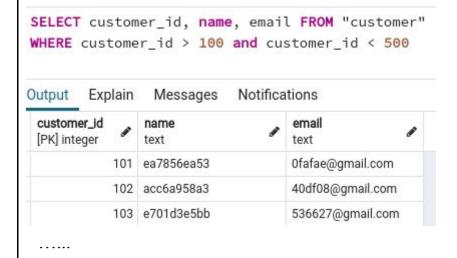
Звертання до таблиці з використанням фільтра (без індекса):

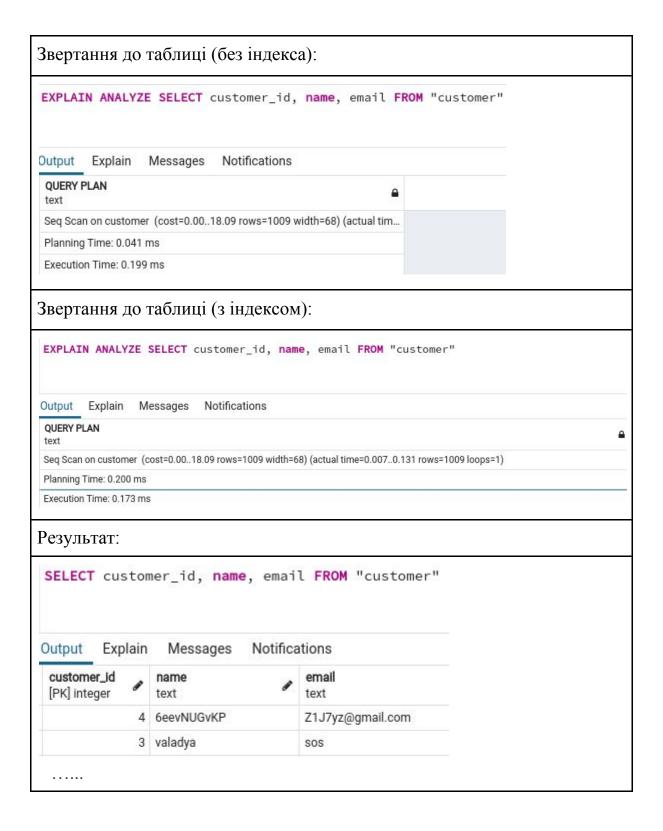


Звертання до таблиці з використанням фільтра (з індексом):

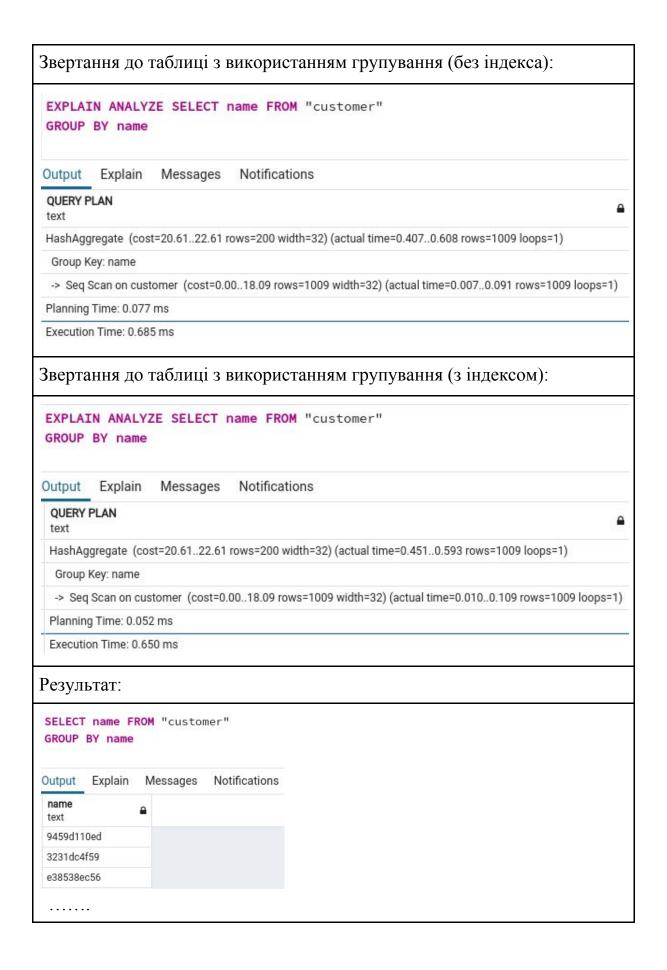


Результат:









Як і очікувалось, доданий ВТree індекс на колонку customer_id, що ϵ PrimaryKey даної сутності, ніяк не вплива ϵ на продуктивність запитів, адже саме цей індекс використову ϵ PrimaryKey 'під капотом'.

GIN

```
ALTER TABLE "product" ADD COLUMN ts_pname tsvector;

UPDATE "product" SET ts_pname = to_tsvector (name) WHERE true;

CREATE INDEX ginProductNameIndex ON "product" USING gin(ts_pname);
```

Після додавання даного індексу ми повинні спостерігати прискорення пошуку по імені конкретного продукту. Впровадження даного індексу ϵ доречним в силу використання приладів, де покупці за назвою продукту генерують цінники.

Практика показала відповідність теорії, при збільшенні часу планування вдвічі, швидкість пошуку об'єкта з фіксованим значенням проіндексованої колонки, зменшилась втричі!

Звертання до таблиці з використанням фільтра (без індекса): EXPLAIN ANALYZE SELECT product_id, name, cost, description FROM "product" WHERE name LIKE 'tomato' Output Explain Messages Notifications QUERY PLAN Seq Scan on product (cost=0.00..17.88 rows=3 width=100) (actual time=0.114..0.115 rows=1 loops=1) Filter: (name ~~ 'tomato'::text) Rows Removed by Filter: 999 Planning Time: 0.042 ms Execution Time: 0.131 ms Звертання до таблиці з використанням фільтра (з індексом): EXPLAIN ANALYZE SELECT product_id, name, cost, description FROM "product" WHERE to_tsquery('tomato') @@ ts_pname Output Explain Messages Notifications **QUERY PLAN** Bitmap Heap Scan on product (cost=8.29..22.45 rows=5 width=100) (actual time=0.022..0.023 rows=1 loops=1) Filter: (to_tsquery('tomato'::text) @@ ts_pname) Heap Blocks: exact=1 -> Bitmap Index Scan on ginproductnameindex (cost=0.00..8.29 rows=5 width=0) (actual time=0.013..0.013 rows=1 loops=1) Index Cond: (ts_pname @@ to_tsquery('tomato'::text)) Planning Time: 0.087 ms Execution Time: 0.043 ms Результат: SELECT product_id, name, cost, description FROM "product" WHERE name LIKE 'tomato' Output Explain Messages Notifications name 🎤 product_id description cost [PK] integer text numeric text 8 tomato 113 6b4a7e89d919...

Звертання до таблиці з використанням агрегатної функції (без індекса): EXPLAIN ANALYZE SELECT COUNT(product_id) FROM "product" WHERE name LIKE 'tomato' Output Explain Messages Notifications **QUERY PLAN** Aggregate (cost=17.88..17.89 rows=1 width=8) (actual time=0.119..0.119 rows=1 loops=1) -> Seq Scan on product (cost=0.00..17.88 rows=3 width=4) (actual time=0.115..0.116 rows=1 loops=1) Filter: (name ~~ 'tomato'::text) Rows Removed by Filter: 999 Planning Time: 0.049 ms Execution Time: 0.139 ms Звертання до таблиці з використанням агрегатної функції (з індексом): EXPLAIN ANALYZE SELECT COUNT(product_id) FROM "product" WHERE to_tsquery('tomato') @@ ts_pname Output Explain Messages Notifications QUERY PLAN text Aggregate (cost=22.46..22.47 rows=1 width=8) (actual time=0.033..0.034 rows=1 loops=1) -> Bitmap Heap Scan on product (cost=8.29..22.45 rows=5 width=4) (actual time=0.029..0.030 rows=1 loops=1) Filter: (to_tsquery('tomato'::text) @@ ts_pname) Heap Blocks: exact=1 -> Bitmap Index Scan on ginproductnameindex (cost=0.00..8.29 rows=5 width=0) (actual time=0.021..0.021 rows=1 loops=1 Index Cond: (ts_pname @@ to_tsquery('tomato'::text)) Planning Time: 0.090 ms Execution Time: 0.058 ms Результат: SELECT COUNT(product_id) FROM "product" WHERE name LIKE 'tomato' Output Explain Messages count bigint 1

Завдання №3:

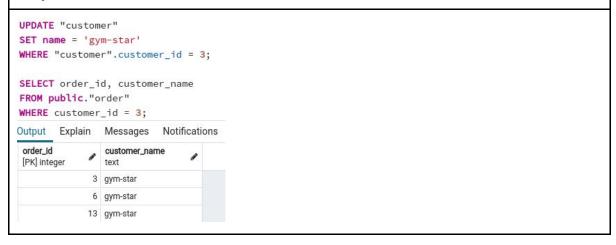
After update. При зміні ім'я клієнта, автоматично змінюємо значення колонки імені на його замовленнях:

```
CREATE OR REPLACE FUNCTION afterUpdateCustomer()
       RETURNS TRIGGER
       LANGUAGE plpgsql
       AS $$
DECLARE
    orders CURSOR IS
       SELECT *
       FROM "order"
       WHERE customer_id = NEW.customer_id;
BEGIN
   FOR order_ IN orders_ LOOP
UPDATE "order"
        SET customer name = NEW.name
        WHERE order_id = order_.order_id;
    end loop;
    return NEW;
END;
$$;
-- declaring func on trigger
{\tt CREATE\ TRIGGER\ after\_update\_customer}
AFTER UPDATE
ON "customer"
FOR EACH ROW
EXECUTE PROCEDURE after update customer();
```

Результат до оновлення:



Результат після оновлення:



After insert. Якщо ім'я нового користувача - 'lucker' - для нього створюється купон зі значенням 25

```
CREATE OR REPLACE FUNCTION after_insert_customer()
    RETURNS TRIGGER
    LANGUAGE plpgsql
    AS $$

BEGIN

IF NEW.name = 'lucker' THEN
    INSERT INTO "coupon" (value)
    VALUES 25;

return NEW;

END;

$$;

CREATE TRIGGER after_insert_customer
    AFTER INSERT
    ON public."customer"
    FOR EACH ROW
    EXECUTE PROCEDURE after_insert_customer();
```