Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп'ютених наук та інформаційних технологій

Кафедра систем штучного інтелекту

Практична робота №13

з курсу:

“ОБДЗ”

Виконала :

студентка групи КН-210

Бурцьо Ольга

Перевірив:

Мельникова Н.І.

Львів – 2020

Аналіз та оптимізація запитів

Мета роботи: Навчитися аналізувати роботу СУБД та оптимізовувати виконання складних запитів на вибірку даних. Виконати аналіз складних запитів за допомогою директиви EXPLAIN, модифікувати найповільніші запити з метою їх пришвидчення.

Хід роботи

1. Проаналізуємо виконання складного запиту попередньої лабораторної роботи

```
explain analyze
    select c.first_name, c.last_name, ad.country, cost, type_of_task from orders
        inner join apartment a on orders.id_apartment = a.id
        inner join client c on orders.id_clent = c.id
        inner join address ad on a.id_adress = ad.id
        where ad.country in ('Ukraine', 'Poland')
        order by cost;
```

```
QUERY PLAN
Sort  (cost=26.06..26.07 rows=1 width=143) (actual time=0.128..0.129 rows=16 loops=1)
  Sort Key: orders.cost
  Sort Method: quicksort  Memory: 26kB
  ->  Nested Loop  (cost=1.82..26.05 rows=1 width=143) (actual time=0.058..0.116 rows=16 loops=1)
        ->  Nested Loop  (cost=1.68..25.36 rows=1 width=135) (actual time=0.053..0.094 rows=16 loops=1)
              ->  Hash Join  (cost=1.54..18.98 rows=24 width=21) (actual time=0.043..0.052 rows=28 loops=1)
                    Hash Cond: (a.id = orders.id_apartment)
                    ->  Seq Scan on apartment a  (cost=0.00..14.80 rows=480 width=8) (actual time=0.015..0.016 rows=21 loops=1)
                    ->  Hash  (cost=1.24..1.24 rows=24 width=21) (actual time=0.022..0.022 rows=28 loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 10kB
                          ->  Seq Scan on orders  (cost=0.00..1.24 rows=24 width=21) (actual time=0.010..0.014 rows=28 loops=1)
              ->  Index Scan using address_pk on address ad  (cost=0.14..0.26 rows=1 width=122) (actual time=0.001..0.001 rows=1 loops=28)
                    Index Cond: (id = a.id_adress)
                    Filter: ((country)::text = ANY ('{Ukraine,Poland}'::text[]))
                    Rows Removed by Filter: 0
        ->  Index Scan using client_pk on client c  (cost=0.14..0.66 rows=1 width=16) (actual time=0.001..0.001 rows=1 loops=16)
              Index Cond: (id = orders.id_clent)
Planning Time: 0.382 ms
Execution Time: 0.177 ms
```

Час виконання - 0.177 мілісекунди

2. Створимо індекси для пришвидшення роботи та переглянемо їх

```sql
create index client_name_idx on client(id, last_name);

create unique index order_cl_idx on orders(id_client, id_apartment);

select indexname, indexdef from pg_indexes
where tablename = 'client';

select indexname, indexdef from pg_indexes
where tablename = 'orders';
```

	indexname	indexdef
1	client_pk	CREATE UNIQUE INDEX client_pk ON public.client USING btree (id)
2	client_name_idx	CREATE INDEX client_name_idx ON public.client USING btree (id, last_name)

	indexname	indexdef
1	order_pk	CREATE UNIQUE INDEX order_pk ON public.orders USING btree (id)
2	order_cl_idx	CREATE UNIQUE INDEX order_cl_idx ON public.orders USING btree (id_client, id_apartment)

3. Тепер перевіримо чи оптимізувалась робота

```
QUERY PLAN
Sort  (cost=25.25..25.26 rows=1 width=143) (actual time=0.109..0.110 rows=11 loops=1)
  Sort Key: orders.cost
  Sort Method: quicksort  Memory: 25kB
  ->  Nested Loop  (cost=1.75..25.24 rows=1 width=143) (actual time=0.056..0.100 rows=11 loops=1)
        ->  Nested Loop  (cost=1.61..24.47 rows=1 width=135) (actual time=0.052..0.084 rows=11 loops=1)
              ->  Hash Join  (cost=1.47..18.88 rows=21 width=21) (actual time=0.043..0.051 rows=21 loops=1)
                    Hash Cond: (a.id = orders.id_apartment)
                    ->  Seq Scan on apartment a  (cost=0.00..14.80 rows=480 width=8) (actual time=0.013..0.015 rows=21 loops=1)
                    ->  Hash  (cost=1.21..1.21 rows=21 width=21) (actual time=0.020..0.020 rows=21 loops=1)
                          Buckets: 1024  Batches: 1  Memory Usage: 10kB
                          ->  Seq Scan on orders  (cost=0.00..1.21 rows=21 width=21) (actual time=0.010..0.013 rows=21 loops=1)
              ->  Index Scan using address_pk on address ad  (cost=0.14..0.26 rows=1 width=122) (actual time=0.001..0.001 rows=1 loops=21)
                    Index Cond: (id = a.id_adress)
                    Filter: ((country)::text = ANY ('{Ukraine,Poland}'::text[]))
                    Rows Removed by Filter: 0
        ->  Index Scan using client_name_idx on client c  (cost=0.14..0.73 rows=1 width=16) (actual time=0.001..0.001 rows=1 loops=11)
              Index Cond: (id = orders.id_client)
Planning Time: 0.392 ms
Execution Time: 0.142 ms
```

Час виконання - 0.142.

Висновок: на даній лабораторній роботі я навчилась аналізувати роботу СУБД та оптимізувати виконання складних запитів на вибірку даних. Виконала аналіз складних запитів за допомогою директиви EXPLAIN, модифікувати найповільніші запити з метою їх пришвидшення. Оскільки дана бд несильно наповнена, то робота не сильно оптимізувалась.