데이터베이스 #assignment6

수업		Cose371-02		교수		정연돈교수	
대학	정보대학	학과	컴퓨터과	학번	2018320211	이름	진지과

Equivalent SQL Statements Exercise

```
select distinct unsorted from table1 where unsorted between 967 and 969;

select distinct unsorted from table1 where unsorted in (967,968,969);

select distinct unsorted from table1 where unsorted=967 or unsorted=968 or unsorted=969;

select unsorted from table1 where unsorted=967
union select unsorted from table1 where unsorted=968
union select unsorted from table1 where unsorted=969;
```

Q1. EXPLAIN ANALYZE your SQL statements and discuss the results of the queries

a. between

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted between 967 and 969;
                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Unique  (cost=253093.64..253093.72 rows=16 width=4) (actual time=953.575..953.588 rows=3 loops=1)
   ->  Sort  (cost=253093.64..253093.68 rows=16 width=4) (actual time=953.574..953.576 rows=18 loops=1)
         Sort Key: unsorted
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on table1  (cost=0.00..253093.32 rows=16 width=4) (actual time=76.664..953.522 rows=18 loops=1)
               Filter: ((unsorted >= 967) AND (unsorted <= 969))
               Rows Removed by Filter: 9999982
 Planning Time: 0.044 ms
 JIT:
   Functions: 5
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 0.795 ms, Inlining 0.000 ms, Optimization 0.321 ms, Emission 3.798 ms, Total 4.914 ms
 Execution Time: 954.425 ms
(13 rows)
```

b. in

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted in (967,968,969);
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Unique  (cost=240593.64..240593.72 rows=17 width=4) (actual time=1077.820..1077.835 rows=3 loops=1)
   ->  Sort  (cost=240593.64..240593.68 rows=17 width=4) (actual time=1077.818..1077.820 rows=18 loops=1)
         Sort Key: unsorted
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on table1  (cost=0.00..240593.29 rows=17 width=4) (actual time=89.560..1077.766 rows=18 loops=1)
               Filter: (unsorted = ANY ('{967,968,969}'::integer[]))
               Rows Removed by Filter: 9999982
 Planning Time: 0.055 ms
 JIT:
   Functions: 5
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 1.001 ms, Inlining 0.000 ms, Optimization 0.322 ms, Emission 3.913 ms, Total 5.236 ms
 Execution Time: 1078.888 ms
(13 rows)
```

c. or

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted=967 or unsorted=968 or unsorted=969;
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Unique  (cost=278093.71..278093.80 rows=17 width=4) (actual time=999.373..999.386 rows=3 loops=1)
   ->  Sort  (cost=278093.71..278093.76 rows=17 width=4) (actual time=999.370..999.371 rows=18 loops=1)
         Sort Key: unsorted
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on table1  (cost=0.00..278093.37 rows=17 width=4) (actual time=83.952..999.313 rows=18 loops=1)
               Filter: ((unsorted = 967) OR (unsorted = 968) OR (unsorted = 969))
               Rows Removed by Filter: 9999982
 Planning Time: 0.114 ms
 JIT:
   Functions: 5
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 1.042 ms, Inlining 0.000 ms, Optimization 0.406 ms, Emission 5.361 ms, Total 6.809 ms
 Execution Time: 1000.482 ms
(13 rows)
```

d. union

```
hw6=# explain analyze select unsorted from table1 where unsorted=967
hw6-# union select unsorted from table1 where unsorted=968
hw6-# union select unsorted from table1 where unsorted=969;
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=684280.10..684280.28 rows=18 width=4) (actual time=2799.666..2799.666 rows=3 loops=1)
   Group Key: table1.unsorted
   ->  Append  (cost=0.00..684280.06 rows=18 width=4) (actual time=1010.936..2799.568 rows=18 loops=1)
         ->  Seq Scan on table1  (cost=0.00..228093.26 rows=6 width=4) (actual time=1010.934..1327.125 rows=1 loops=1)
               Filter: (unsorted = 967)
               Rows Removed by Filter: 9999999
         ->  Seq Scan on table1 table1_1  (cost=0.00..228093.26 rows=6 width=4) (actual time=54.112..751.105 rows=5 loops=1)
               Filter: (unsorted = 968)
               Rows Removed by Filter: 9999995
         ->  Seq Scan on table1 table1_2  (cost=0.00..228093.26 rows=6 width=4) (actual time=78.843..721.318 rows=12 loops=1)
               Filter: (unsorted = 969)
               Rows Removed by Filter: 9999988
 Planning Time: 0.059 ms
 JIT:
   Functions: 17
   Options: Inlining true, Optimization true, Expressions true, Deforming true
   Timing: Generation 2.378 ms, Inlining 129.626 ms, Optimization 304.309 ms, Emission 180.039 ms, Total 616.352 ms
 Execution Time: 2802.155 ms
(18 rows)
```

모두 unsorted이기 때문에 sort후 seq scan을 실행합니다.

(d)에서는 'select unsorted from table where unsorted=X' 을 3번씩 실행하고 union을 하기때문에, 수행 시간이 다른 것 보다 더 길게 나타납니다.

Q2. Create an Btree index on “unsorted” column and repeat Q1.

```
create index on table1 using btree(unsorted);
```

a. between

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted between 967 and 969;
                                                          QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------
----
 Unique  (cost=0.43..68.79 rows=16 width=4) (actual time=0.021..0.103 rows=3 loops=1)
   ->  Index Only Scan using table1_unsorted_idx on table1  (cost=0.43..68.75 rows=16 width=4) (actual time=0.018..0.089 rows=18 loops
=1)
         Index Cond: ((unsorted >= 967) AND (unsorted <= 969))
         Heap Fetches: 18
 Planning Time: 0.227 ms
 Execution Time: 0.131 ms
(6 rows)
```

b. in

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted in (967,968,969);
                                                          QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=0.43..81.65 rows=17 width=4) (actual time=0.029..0.049 rows=3 loops=1)
   ->  Index Only Scan using table1_unsorted_idx on table1  (cost=0.43..81.61 rows=17 width=4) (actual time=0.029..0.045 rows=18 loops=1)
         Index Cond: (unsorted = ANY ('{967,968,969}'::integer[]))
         Heap Fetches: 18
 Planning Time: 0.066 ms
 Execution Time: 0.063 ms
(6 rows)
```

c. or

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted=967 or unsorted=968 or unsorted=969;
                                                     QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=278093.35..278093.43 rows=17 width=4) (actual time=1080.959..1080.973 rows=3 loops=1)
   ->  Sort  (cost=278093.35..278093.39 rows=17 width=4) (actual time=1080.958..1080.959 rows=18 loops=1)
         Sort Key: unsorted
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on table1  (cost=0.00..278093.00 rows=17 width=4) (actual time=95.515..1080.640 rows=18 loops=1)
               Filter: ((unsorted = 967) OR (unsorted = 968) OR (unsorted = 969))
               Rows Removed by Filter: 9999982
 Planning Time: 0.109 ms
 JIT:
   Functions: 5
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 2.362 ms, Inlining 0.000 ms, Optimization 0.683 ms, Emission 5.884 ms, Total 8.929 ms
 Execution Time: 1083.405 ms
(13 rows)
```

d. union

```
hw6-# union select unsorted from table1 where unsorted=969;
                                                          QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=85.93..86.11 rows=18 width=4) (actual time=0.055..0.056 rows=3 loops=1)
   Group Key: table1.unsorted
   ->  Append  (cost=0.43..85.89 rows=18 width=4) (actual time=0.015..0.047 rows=18 loops=1)
         ->  Index Only Scan using table1_unsorted_idx on table1  (cost=0.43..28.54 rows=6 width=4) (actual time=0.015..0.017 rows=1 loops=1)
               Index Cond: (unsorted = 967)
               Heap Fetches: 1
         ->  Index Only Scan using table1_unsorted_idx on table1 table1_1  (cost=0.43..28.54 rows=6 width=4) (actual time=0.004..0.009 rows=5 loops=1)
               Index Cond: (unsorted = 968)
               Heap Fetches: 5
         ->  Index Only Scan using table1_unsorted_idx on table1 table1_2  (cost=0.43..28.54 rows=6 width=4) (actual time=0.012..0.020 rows=12 loops=1)
               Index Cond: (unsorted = 969)
               Heap Fetches: 12
 Planning Time: 0.087 ms
 Execution Time: 0.082 ms
(14 rows)
```

(a)(b)(d)는 모두 퀵소트후 index only scan을 사용하고, 각 경우 간 수행시간이 크게 차이가 나지 않습니다.

or연산을 사용하는 (c)는 sort를 먼저 실행하고 seq scan을 하는 것을 볼 수 있습니다. 시간 또한 다른 경우보다 월등하게 많이 걸리는 것을 볼 수 있습니다.

Q3. Create an hash index on “unsorted” column and repeat Q1.

```
drop index table1_unsorted_index;
create index on table1 using hash(unsorted);
```

a. between

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted between 967 and 969;
                                                              QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=166595.05..166595.13 rows=17 width=4) (actual time=1426.965..1426.984 rows=3 loops=1)
   ->  Sort  (cost=166595.05..166595.09 rows=17 width=4) (actual time=1426.964..1426.969 rows=18 loops=1)
         Sort Key: unsorted
         Sort Method: quicksort  Memory: 25kB
         ->  Gather  (cost=1000.00..166594.70 rows=17 width=4) (actual time=723.744..1431.012 rows=18 loops=1)
               Workers Planned: 2
               Workers Launched: 2
               ->  Parallel Seq Scan on table1  (cost=0.00..165593.00 rows=7 width=4) (actual time=445.609..1357.631 rows=6 loops=3)
                     Filter: ((unsorted >= 967) AND (unsorted <= 969))
                     Rows Removed by Filter: 3333327
 Planning Time: 0.259 ms
 JIT:
   Functions: 13
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 6.281 ms, Inlining 0.000 ms, Optimization 1.660 ms, Emission 16.170 ms, Total 24.111 ms
 Execution Time: 1433.315 ms
(16 rows)
```

b. in

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted in (967,968,969);
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=84.05..84.14 rows=18 width=4) (actual time=0.080..0.086 rows=3 loops=1)
   ->  Sort  (cost=84.05..84.09 rows=18 width=4) (actual time=0.078..0.079 rows=18 loops=1)
         Sort Key: unsorted
         Sort Method: quicksort  Memory: 25kB
         ->  Bitmap Heap Scan on table1  (cost=12.14..83.67 rows=18 width=4) (actual time=0.041..0.069 rows=18 loops=1)
               Recheck Cond: (unsorted = ANY ('{967,968,969}'::integer[]))
               Heap Blocks: exact=18
               ->  Bitmap Index Scan on table1_unsorted_idx  (cost=0.00..12.13 rows=18 width=0) (actual time=0.031..0.031 rows=18 loops=1)
                     Index Cond: (unsorted = ANY ('{967,968,969}'::integer[]))
 Planning Time: 0.109 ms
 Execution Time: 0.114 ms
(11 rows)
```

c. or

```
hw6=# explain analyze select distinct unsorted from table1 where unsorted=967 or unsorted=968 or unsorted=969;
                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=84.13..84.22 rows=18 width=4) (actual time=0.075..0.084 rows=3 loops=1)
   ->  Sort  (cost=84.13..84.17 rows=18 width=4) (actual time=0.075..0.079 rows=18 loops=1)
         Sort Key: unsorted
         Sort Method: quicksort  Memory: 25kB
         ->  Bitmap Heap Scan on table1  (cost=12.15..83.75 rows=18 width=4) (actual time=0.034..0.060 rows=18 loops=1)
               Recheck Cond: ((unsorted = 967) OR (unsorted = 968) OR (unsorted = 969))
               Heap Blocks: exact=18
               ->  BitmapOr  (cost=12.15..12.15 rows=18 width=0) (actual time=0.019..0.020 rows=0 loops=1)
                     ->  Bitmap Index Scan on table1_unsorted_idx  (cost=0.00..4.04 rows=6 width=0) (actual time=0.008..0.008 rows=1 loops=1)
                           Index Cond: (unsorted = 967)
                     ->  Bitmap Index Scan on table1_unsorted_idx  (cost=0.00..4.04 rows=6 width=0) (actual time=0.006..0.006 rows=5 loops=1)
                           Index Cond: (unsorted = 968)
                     ->  Bitmap Index Scan on table1_unsorted_idx  (cost=0.00..4.04 rows=6 width=0) (actual time=0.004..0.004 rows=12 loops=1)
                           Index Cond: (unsorted = 969)
 Planning Time: 0.128 ms
 Execution Time: 0.190 ms
(16 rows)
```

d. union

```
hw6=# explain analyzeselect unsorted from table1 where unsorted=967
hw6-# union select unsorted from table1 where unsorted=968
hw6-# union select unsorted from table1 where unsorted=969;
ERROR:  syntax error at or near "analyzeselect"
LINE 1: explain analyzeselect unsorted from table1 where unsorted=96...
                ^
hw6=# explain analyze select unsorted from table1 where unsorted=967
hw6-# union select unsorted from table1 where unsorted=968
hw6-# union select unsorted from table1 where unsorted=969;
                                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=84.63..84.81 rows=18 width=4) (actual time=0.073..0.074 rows=3 loops=1)
   Group Key: table1.unsorted
   ->  Append  (cost=0.00..84.59 rows=18 width=4) (actual time=0.018..0.058 rows=18 loops=1)
         ->  Index Scan using table1_unsorted_idx on table1  (cost=0.00..28.11 rows=6 width=4) (actual time=0.017..0.018 rows=1 loops=1)
               Index Cond: (unsorted = 967)
         ->  Index Scan using table1_unsorted_idx on table1 table1_1  (cost=0.00..28.11 rows=6 width=4) (actual time=0.006..0.015 rows=5 loops=1)
               Index Cond: (unsorted = 968)
         ->  Index Scan using table1_unsorted_idx on table1 table1_2  (cost=0.00..28.11 rows=6 width=4) (actual time=0.004..0.021 rows=12 loops=1)
               Index Cond: (unsorted = 969)
 Planning Time: 0.187 ms
 Execution Time: 0.138 ms
(11 rows)
```

(b)(c)에선 우선 quick sort을 실행하고 seq scan을 실행합니다.

union 인(d)의 경우 index scan으로 빠른 곳도를 보여줍니다.

(a)between연산에서는 quick sort후 parallel하게 seq scan을 사용하고, 속도가 다른 것에 비해 많이 느린 것을 확인할 수 있습니다.

Q4 . Compare each SQL statements' performances on three cases(no index, Btree index, hash index)

no index: 전반적으로 느립니다. 특히 union 사용시 더 느린 속도를 보여줍니다.

b-tree: or연산에 적합하지 않습니다.

hash: between과 같은 범위 계산에 적합하지 않는 것 같습니다.

Query Plan

Q6. Write the queries and use EXPLAIN ANALYZE to see how the query execution is actually planned

a. Union tables(poo11, pool2), and then perform aggregation with COUNT function

```
select count(val) from (select * from pool1 union all select * from pool2)as t;
```

```
hw6=# explain analyze select count(val) from (select * from pool1 union all select * from pool2)as t;
                                                                    QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=118164.88..118164.89 rows=1 width=8) (actual time=1014.009..1014.010 rows=1 loops=1)
   ->  Gather  (cost=118164.66..118164.87 rows=2 width=8) (actual time=1013.867..1020.803 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=117164.66..117164.67 rows=1 width=8) (actual time=992.645..992.645 rows=1 loops=3)
               ->  Parallel Append  (cost=0.00..106748.00 rows=4166666 width=4) (actual time=0.075..671.326 rows=3333333 loops=3)
                     ->  Parallel Seq Scan on pool1  (cost=0.00..42957.33 rows=2083333 width=4) (actual time=0.213..201.684 rows=1666667 loops=3)
                     ->  Parallel Seq Scan on pool2  (cost=0.00..42957.33 rows=2083333 width=4) (actual time=0.076..287.273 rows=2500000 loops=2)
 Planning Time: 0.118 ms
 JIT:
   Functions: 8
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 2.304 ms, Inlining 0.000 ms, Optimization 9.582 ms, Emission 12.098 ms, Total 23.984 ms
 Execution Time: 1022.430 ms
(14 rows)
```

b.

Perform aggregation with COUNT function on each table, and then aggregate them again with SUM function on the union
of the aggregated results

```
select sum(c) from (select count(val) as c from pool1 union all select count(val) as c from pool2 )as t;
```

```
hw6=# explain analyze select sum(c) from (select count(val) as c from pool1 union all select count(val) as c from pool2 )as t;
                                                              QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=85624.23..85624.24 rows=1 width=32) (actual time=792.509..792.509 rows=1 loops=1)
   ->  Gather  (cost=85624.00..85624.22 rows=2 width=8) (actual time=725.686..794.433 rows=2 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Parallel Append  (cost=84624.00..84624.02 rows=1 width=8) (actual time=488.496..488.496 rows=1 loops=3)
               ->  Aggregate  (cost=84624.00..84624.01 rows=1 width=8) (actual time=740.058..740.059 rows=1 loops=1)
                     ->  Seq Scan on pool1  (cost=0.00..72124.00 rows=5000000 width=4) (actual time=0.089..407.425 rows=5000000 loops=1)
               ->  Aggregate  (cost=84624.00..84624.01 rows=1 width=8) (actual time=725.425..725.425 rows=1 loops=1)
                     ->  Seq Scan on pool2  (cost=0.00..72124.00 rows=5000000 width=4) (actual time=1.107..388.119 rows=5000000 loops=1)
 Planning Time: 0.137 ms
 Execution Time: 794.492 ms
(11 rows)
```

(a) POOL1와 POOL2를 union한 후, 카운트를 다시 합니다.

(b) POOL1와 POOL2를 각각 카운트 한 값을 union하고, 이를 더합니다.

루프를 더 많이 도는 (a)가 시간이 더 많이 걸리는 것을 확인할 수 있습니다.

Q7.Write the queries and use EXPLAIN ANALYZE to see how query execution is actually planned

a. SELECT tuple WHERE value is above 250 on each table and then union them

```
select *
from (select * from pool1 where val >250
      union all
      select * from pool2 where val>250
)as t;
```

```
hw6=# explain analyze select * from (select val from pool1 where val >250 union all select val from pool2 where val>250)as t;

                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Append  (cost=0.00..244101.95 rows=4990263 width=4) (actual time=5.980..1182.503 rows=4986584 loops=1)
   ->  Seq Scan on pool1  (cost=0.00..84624.00 rows=2507333 width=4) (actual time=5.978..446.648 rows=2494590 loops=1)
         Filter: (val > 250)
         Rows Removed by Filter: 2505410
   ->  Seq Scan on pool2  (cost=0.00..84624.00 rows=2482930 width=4) (actual time=0.038..405.884 rows=2491994 loops=1)
         Filter: (val > 250)
         Rows Removed by Filter: 2508006
 Planning Time: 0.135 ms
 JIT:
   Functions: 4
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 1.422 ms, Inlining 0.000 ms, Optimization 0.416 ms, Emission 4.580 ms, Total 6.418 ms
 Execution Time: 1362.484 ms
(13 rows)
```

b. Union two tables and SELECT tuples WHERE value is above 250

```
select *
from (
  select * from pool1
  union all
```

```
  select * from pool2
 ) as t
 where val>250;
```

```
hw6=# explain analyze select * from (select * from pool1 union all select * from pool2) as t where val>250;
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Append  (cost=0.00..194199.32 rows=4990263 width=4) (actual time=8.054..1187.359 rows=4986584 loops=1)
   ->  Seq Scan on pool1  (cost=0.00..84624.00 rows=2507333 width=4) (actual time=8.053..450.011 rows=2494590 loops=1)
         Filter: (val > 250)
         Rows Removed by Filter: 2505410
   ->  Seq Scan on pool2  (cost=0.00..84624.00 rows=2482930 width=4) (actual time=0.033..405.915 rows=2491994 loops=1)
         Filter: (val > 250)
         Rows Removed by Filter: 2508006
 Planning Time: 15.169 ms
 JIT:
   Functions: 4
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 1.271 ms, Inlining 0.000 ms, Optimization 0.407 ms, Emission 6.700 ms, Total 8.378 ms
 Execution Time: 1367.173 ms
(13 rows)
```

(a)pool1에서 250보다 큰 값, pool2에서 250보다 큰 값을 각각 찾아서 합칩니다.

(b)pool1와 pool2를 합친 후, 그 union에서 250보다 큰 값을 찾습니다.

쿼리플랜이 동일하고, 수행시간또한 유의미한 차이가 보이지 않습니다.

다만 (b)에 planning time이 (a)보다 많이 걸리는 것을 볼 수 있습니다.

Q8.Why does the user-level optimization important?

동일한 작업을 수행하더라도, 쿼리에 따라 다른 성능의 차이가 크게 나타날 수 있기 때문입니다.