데이터베이스 #assignment5

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

Index Creation and Scan

Q1.create index

```
create index index on table1 (sorted); //clustered index
create index index on table1 (unsorted);//non-clustered index
```

Q2.Make queries eachuses 'seq scan', 'index scan', and 'index only scan'

seq scan

```
explain analyze select * from table1;
```

```
hw5=# explain analyze select * from table1;
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..203093.00 rows=10000000 width=53) (actual time=1.829..1406.920 rows=10000000 loops=1)
 Planning Time: 0.418 ms
 Execution Time: 1776.791 ms
(3 rows)
```

index scan

```
explain analyze select rndm, sorted from table1 where sorted=320211;
explain analyze select unrndm, sorted from table1 where unsorted=320211;
```

```
hw5=# explain analyze select rndm, sorted from table1 where sorted=320211;
                                                      QUERY PLAN

------------------------------------------------------------------------------------------------------------------------
---
 Index Scan using table1_sorted_idx on table1  (cost=0.43..8.70 rows=15 width=8) (actual time=0.023..0.027 rows=5 loops=
1)
   Index Cond: (sorted = 320211)
 Planning Time: 0.114 ms
 Execution Time: 0.049 ms
(4 rows)

hw5=# explain analyze select rndm, unsorted from table1 where unsorted=320211;
                                                       QUERY PLAN

------------------------------------------------------------------------------------------------------------------------
-----
 Index Scan using table1_unsorted_idx on table1  (cost=0.43..28.54 rows=6 width=8) (actual time=0.051..0.055 rows=4 loop
s=1)
   Index Cond: (unsorted = 320211)
 Planning Time: 0.065 ms
 Execution Time: 0.068 ms
(4 rows)
```

index only

```
explain analyze select sorted from table1 where sorted=320211;
explain analyze select unsorted from table1 where unsorted=320211;
```

```
hw5=# explain analyze select sorted from table1 where sorted=320211;
                                                       QUERY PLAN

------------------------------------------------------------------------------------------------------------------------
--------
 Index Only Scan using table1_sorted_idx on table1  (cost=0.43..8.70 rows=15 width=4) (actual time=1.307..1.312 rows=5 l
oops=1)
   Index Cond: (sorted = 320211)
   Heap Fetches: 5
 Planning Time: 2.624 ms
 Execution Time: 1.343 ms
(5 rows)

hw5=# explain analyze select unsorted from table1 where unsorted=320211;
                                                        QUERY PLAN

------------------------------------------------------------------------------------------------------------------------
----------
 Index Only Scan using table1_unsorted_idx on table1  (cost=0.43..28.54 rows=6 width=4) (actual time=1.778..1.786 rows=4
 loops=1)
   Index Cond: (unsorted = 320211)
   Heap Fetches: 4
 Planning Time: 0.113 ms
 Execution Time: 1.811 ms
(5 rows)
```

Q3.

```
explain analyze select sorted, rndm from table where sorted > 100 and sorted < 500 and rndm = 55;
explain analyze select unsorted, rndm from table where unsorted > 100 and unsorted < 500 and rndm = 55
```

```
hw5=# explain analyze SELECT sorted, rndm FROM table1 WHERE sorted > 100 and sorted < 500 AND rndm = 55;
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Index Scan using index1 on table1  (cost=0.43..92.94 rows=1 width=8) (actual time=4.266..4.266 rows=0 loops=1)
   Index Cond: ((sorted > 100) AND (sorted < 500))
   Filter: (rndm = 55)
   Rows Removed by Filter: 1995
 Planning Time: 0.779 ms
 Execution Time: 4.288 ms
(6 rows)

hw5=# explain analyze SELECT unsorted, rndm FROM table1 WHERE unsorted > 100 and unsorted < 500 AND rndm = 55;
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on table1  (cost=44.37..7155.78 rows=1 width=8) (actual time=44.142..44.142 rows=0 loops=1)
   Recheck Cond: ((unsorted > 100) AND (unsorted < 500))
   Filter: (rndm = 55)
   Rows Removed by Filter: 2021
   Heap Blocks: exact=2005
   ->  Bitmap Index Scan on index2  (cost=0.00..44.37 rows=1993 width=0) (actual time=3.830..3.830 rows=2021 loops=1)
         Index Cond: ((unsorted > 100) AND (unsorted < 500))
 Planning Time: 0.300 ms
 Execution Time: 44.224 ms
(9 rows)
```

clustered index가 4.288ms, non-clustered index가 44.244ms이다.

clustered index에서는 데이터가 sorted되어 있기 때문에 bitmap heap scan과 같은 disk access에서 시간을 생략하거나 단축할 수 있어서 더 효율적이다.

Q4.

```
explain analyze select sorted, rndm from table1 where sorted >1999231 and rndm=1005;
explain analyze SELECT sorted , rndm FROM table1 WHERE sorted < 1999231 AND rndm = 1005;
```

```
hw5=# explain analyze select sorted, rndm from table1 where sorted >1999231 and rndm=1005;
                                                      QUERY PLAN

-----------------------------------------------------------------------------------------------------------------------
----
 Index Scan using table1_sorted_idx on table1  (cost=0.43..153.84 rows=1 width=8) (actual time=0.653..0.653 rows=0 loops
=1)
   Index Cond: (sorted > 1999231)
   Filter: (rndm = 1005)
   Rows Removed by Filter: 3840
 Planning Time: 0.139 ms
 Execution Time: 0.665 ms
(6 rows)
```

```
hw5=# explain analyze SELECT sorted , rndm FROM table1 WHERE sorted < 1999231 AND rndm = 1005;
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..253093.00 rows=100 width=8) (actual time=228.958..17110.581 rows=114 loops=1)
   Filter: ((sorted < 1999231) AND (rndm = 1005))
   Rows Removed by Filter: 9999886
 Planning Time: 0.149 ms
 JIT:
   Functions: 4
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 3.073 ms, Inlining 0.000 ms, Optimization 0.827 ms, Emission 5.387 ms, Total 9.287 ms
 Execution Time: 17113.892 ms
(9 rows)
```

(1)는 0.665ms, (2)는 17113.892이다

우선 총 범위가 20000000인데, 1999231보다 큰 값을 조사하는 (1)이 1999231보다 작은 값을 조사하는 (2)보다 검사해야 할 row가 적기 때문에, (1)이 더 빠른 것이다.

B-tree and Hash

create two indexs

```
create index on table_btree using btree (recordid);
create index on table_hash using hash (recordid);
```

```
hw5=# create index on table_btree using btree (recordid);
CREATE INDEX
hw5=# create index on table_hash using hash (recordid);
CREATE INDEX
```

Q5.

```
explain analyze select * from tagble_btreee where recordid=10001;
explain analyze select * from tagble_hash where recordid=10001;
```

```
hw5=# explain analyze Select * from table_btree where recordid = 10001;
                                                              QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------------
---------------
 Index Scan using table_btree_recordid_idx on table_btree  (cost=0.43..8.45 rows=1 width=49) (actual time=1.231..1.233 r
ows=1 loops=1)
   Index Cond: (recordid = 10001)
 Planning Time: 4.287 ms
 Execution Time: 1.261 ms
(4 rows)

hw5=# explain analyze Select * from table_hash where recordid = 10001;
                                                              QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------------
-------------
 Index Scan using table_hash_recordid_idx on table_hash  (cost=0.00..8.02 rows=1 width=49) (actual time=0.049..0.051 row
s=1 loops=1)
   Index Cond: (recordid = 10001)
 Planning Time: 0.295 ms
 Execution Time: 0.070 ms
(4 rows)
```

모두 index scan을 사용하고, hash의 스캔이 더 빠른 것을 볼 수 있다.

Q6.

```
explain analyze select * from table_btree where recordid >250 and recordid <500;
explain analyze select * from table_noindex where recordid >250 and recordid <500;
```

```
[hw5=# explain analyze Select * from table_btree where recordid > 250 and recordid < 550;
                                                                 QUERY PLAN

-------------------------------------------------------------------------------------------------------------------------------------------
--------------------
 Index Scan using table_btree_recordid_idx on table_btree  (cost=0.43..15.78 rows=267 width=49) (actual time=0.452..1.58
4 rows=299 loops=1)
   Index Cond: ((recordid > 250) AND (recordid < 550))
 Planning Time: 3.819 ms
 Execution Time: 1.650 ms
(4 rows)

[hw5=# explain analyze Select * from table_hash where recordid > 250 and recordid < 550;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on table_hash  (cost=0.00..253093.00 rows=1 width=49) (actual time=6.799..1497.806 rows=299 loops=1)
   Filter: ((recordid > 250) AND (recordid < 550))
   Rows Removed by Filter: 9999701
 Planning Time: 3.023 ms
 JIT:
   Functions: 2
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 2.025 ms, Inlining 0.000 ms, Optimization 0.578 ms, Emission 3.374 ms, Total 5.977 ms
 Execution Time: 1500.018 ms
(9 rows)
```

hash에서 index scan이 아닌 seq scan을 사용하고, 더 많은 시간이 소비되는 것을 볼 수 있다.

Q7.

```
explain analyze update table_btree
set recordid=3
where recordid=2;
```

```
[hw5-# where recordid=2;
                                                              QUERY PLAN

-------------------------------------------------------------------------------------------------------------------------------------------
--------------------
 Update on table_btree  (cost=0.43..8.45 rows=1 width=55) (actual time=1.363..1.363 rows=0 loops=1)
   ->  Index Scan using table_btree_recordid_idx on table_btree  (cost=0.43..8.45 rows=1 width=55) (actual time=0.016..0
.017 rows=1 loops=1)
         Index Cond: (recordid = 2)
 Planning Time: 4.977 ms
 Execution Time: 2.874 ms
(5 rows)

...skipping...
                                                              QUERY PLAN
```

```
explain analyze update table_noindex
set recordid=3
where recordid=2;
```

```
[hw5=# explain analyze update table_noindex
[hw5-# set recordid=3
[hw5-# where recordid=2;
                                                       QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Update on table_noindex  (cost=0.00..228093.26 rows=1 width=55) (actual time=2172.172..2172.172 rows=0 loops=1)
   ->  Seq Scan on table_noindex  (cost=0.00..228093.26 rows=1 width=55) (actual time=6.407..2171.457 rows=1 loops=1)
         Filter: (recordid = 2)
         Rows Removed by Filter: 9999999
 Planning Time: 1.119 ms
 JIT:
   Functions: 4
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 2.020 ms, Inlining 0.000 ms, Optimization 0.348 ms, Emission 4.193 ms, Total 6.561 ms
 Execution Time: 2174.360 ms
(10 rows)
```

btree는 index scan로 1.119ms, noindex는 seq scan으로 2174.360ms 이다.

한 개의 key를 찾고 업데이트를 하는 것이니, btree가 더 효과적인 것을 확인할 수 있다.

Q8

```
explain analyze update table_btree
set recordidd =recordid *2
where recordid > 8000000;

explain analyze update table_noindex
set recordidd =recordid *2
where recordid > 8000000;
```

```
hw5=# explain analyze update table_btree
set recordid = recordid*2
where recordid>8000000;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Update on table_btree  (cost=0.43..81708.38 rows=1979347 width=55) (actual time=23300.290..23300.290 rows=0 loops=1)
   ->  Index Scan using table_btree_recordid_idx on table_btree  (cost=0.43..81708.38 rows=1979347 width=55) (actual tim
e=1.215..2018.561 rows=1999999 loops=1)
         Index Cond: (recordid > 8000000)
 Planning Time: 0.094 ms
 Execution Time: 23300.320 ms
(5 rows)

hw5=# explain analyze update table_noindex
hw5-# set recordid=recordid*2
hw5-# where recordid>8000000;
                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Update on table_noindex  (cost=0.00..233067.23 rows=1989587 width=55) (actual time=12134.881..12134.882 rows=0 loops=1)
   ->  Seq Scan on table_noindex  (cost=0.00..233067.23 rows=1989587 width=55) (actual time=1108.439..1898.394 rows=1999
999 loops=1)
         Filter: (recordid > 8000000)
         Rows Removed by Filter: 8000001
 Planning Time: 0.072 ms
 JIT:
   Functions: 4
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 2.059 ms, Inlining 0.000 ms, Optimization 0.509 ms, Emission 4.388 ms, Total 6.956 ms
 Execution Time: 12137.024 ms
(10 rows)
```

btree 는 index scan으로 23300.320ms, noindex는 seq scan으로 12137.023ms이다

부분(한 개가 아닌 다수)을 업데이트를 하는 것은 noindex가 더 효율적인 것을 볼 수 있다.

Q9.

```
explain analyze update table_btree
set recordid = recordid*5;

explain analyze update table_noindex
set recordid = recordid*5;
```

```
hw5=# explain analyze update table_btree
hw5-# set recordid=recordid*5;
                                                                  QUERY PLAN

-----------------------------------------------------------------------------------------------------------------------------------------------
----------
 Update on table_btree  (cost=0.00..248712.00 rows=10000000 width=55) (actual time=83121.647..83121.647 rows=0 loops=1)
   ->  Seq Scan on table_btree  (cost=0.00..248712.00 rows=10000000 width=55) (actual time=4.871..9542.640 rows=10000000
 loops=1)
 Planning Time: 3.420 ms
 JIT:
   Functions: 2
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 1.199 ms, Inlining 0.000 ms, Optimization 0.569 ms, Emission 3.281 ms, Total 5.049 ms
 Execution Time: 83123.115 ms
(8 rows)

hw5=# explain analyze update table_noindex
hw5-# set recordid=recordid*5;
                                                                  QUERY PLAN

-----------------------------------------------------------------------------------------------------------------------------------------------
-------------
 Update on table_noindex  (cost=0.00..248712.00 rows=10000000 width=55) (actual time=34539.422..34539.422 rows=0 loops=1
)
   ->  Seq Scan on table_noindex  (cost=0.00..248712.00 rows=10000000 width=55) (actual time=5.701..12588.149 rows=10000
000 loops=1)
 Planning Time: 2.786 ms
 JIT:
   Functions: 2
   Options: Inlining false, Optimization false, Expressions true, Deforming true
   Timing: Generation 0.925 ms, Inlining 0.000 ms, Optimization 0.276 ms, Emission 3.254 ms, Total 4.455 ms
 Execution Time: 34540.471 ms
(8 rows)
```

btree는 seq scan으로 83123.155ms, noindex는 seq scan으로 34540.471ms이다.

noindex의 효율이 더 많이 좋은 것을 확인할 수 있다.