2019320097_조이강

Exercise 1

- 각 nationkey로 Equi-Join할 경우, a는 다른 조건이 없으므로 Hash join을 사용할 것 같고, b는 order by가 있으므로 Merge join을 사용할 것 같습니다.

```
d2019320097=# Explain analyze select * from nation a join supplier s on a.n_nationKey = s.s_nationKey;
                                                      QUERY PLAN
                                                      
------------------------------------------------------------------------------------------------------------------------
----
 Hash Join  (cost=442.00..1024.40 rows=27000 width=261) (actual time=4.541..6.725 rows=9580 loops=1)
   Hash Cond: (a.n_nationkey = s.s_nationkey)
   ->  Seq Scan on nation a  (cost=0.00..15.40 rows=540 width=122) (actual time=0.028..0.032 rows=26 loops=1)
   ->  Hash  (cost=317.00..317.00 rows=10000 width=139) (actual time=4.419..4.420 rows=10000 loops=1)
         Buckets: 16384  Batches: 1  Memory Usage: 1848kB
         ->  Seq Scan on supplier s  (cost=0.00..317.00 rows=10000 width=139) (actual time=0.014..1.501 rows=10000 loops
=1)
 Planning Time: 2.755 ms
 Execution Time: 7.479 ms
(8개 행)
```

- a의 경우, Hash join을 사용하게 됩니다.

```
d2019320097=# Explain analyze select * from nation a join supplier s on a.n_nationKey = s.s_nationKey order by s_nationK
ey;
                                                      QUERY PLAN
                                                      
------------------------------------------------------------------------------------------------------------------------
----
 Merge Join  (cost=1021.29..1428.99 rows=27000 width=261) (actual time=5.174..9.117 rows=9580 loops=1)
   Merge Cond: (a.n_nationkey = s.s_nationkey)
   ->  Sort  (cost=39.91..41.26 rows=540 width=122) (actual time=0.043..0.047 rows=25 loops=1)
         Sort Key: a.n_nationkey
         Sort Method: quicksort  Memory: 26kB
         ->  Seq Scan on nation a  (cost=0.00..15.40 rows=540 width=122) (actual time=0.028..0.032 rows=26 loops=1)
   ->  Sort  (cost=981.39..1006.39 rows=10000 width=139) (actual time=4.983..6.342 rows=10000 loops=1)
         Sort Key: s.s_nationkey
         Sort Method: quicksort  Memory: 2061kB
         ->  Seq Scan on supplier s  (cost=0.00..317.00 rows=10000 width=139) (actual time=0.010..1.210 rows=10000 loops
=1)
 Planning Time: 0.181 ms
 Execution Time: 10.031 ms
(12개 행)
```

- b의 경우, Merge join이 사용되었습니다.

Exercise 2

- a의 경우, 동일한 테이블에서 nationkey를 가지고 Join하므로, Hash Join을 사용할 것 같습니다.
- b의 경우, Non equi-Join이므로 Nested loop join을 사용할 것 같습니다.

```
d2019320097=# Explain analyze select * from nation a join nation b on a.n_nationkey = b.n_nationkey;
                                                    QUERY PLAN
                                                    
------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=22.15..89.93 rows=1458 width=244) (actual time=0.075..0.087 rows=26 loops=1)
   Hash Cond: (a.n_nationkey = b.n_nationkey)
   ->  Seq Scan on nation a  (cost=0.00..15.40 rows=540 width=122) (actual time=0.029..0.031 rows=26 loops=1)
   ->  Hash  (cost=15.40..15.40 rows=540 width=122) (actual time=0.024..0.024 rows=26 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 10kB
         ->  Seq Scan on nation b  (cost=0.00..15.40 rows=540 width=122) (actual time=0.005..0.007 rows=26 loops=1)
 Planning Time: 0.181 ms
 Execution Time: 0.116 ms
(8개 행)
```

- a의 경우, Hash join을 사용합니다.

```
d2019320097=# Explain analyze select * from nation a join nation b on not a.n_nationkey = b.n_nationkey;
                                                          QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.00..4406.15 rows=290142 width=244) (actual time=0.019..0.080 rows=650 loops=1)
   Join Filter: (a.n_nationkey <> b.n_nationkey)
   Rows Removed by Join Filter: 26
   ->  Seq Scan on nation a  (cost=0.00..15.40 rows=540 width=122) (actual time=0.012..0.012 rows=26 loops=1)
   ->  Materialize  (cost=0.00..18.10 rows=540 width=122) (actual time=0.000..0.001 rows=26 loops=26)
         ->  Seq Scan on nation b  (cost=0.00..15.40 rows=540 width=122) (actual time=0.002..0.003 rows=26 loops=1)
 Planning Time: 1.030 ms
 Execution Time: 0.110 ms
(8개 행)
```

- b의 경우, Nested loop join을 사용합니다.

Exercise 3

- a의 경우, sorted가 정렬된 데이터이므로, Merge Join을 사용할 것 같습니다. 실행시간은 b에 비해 짧을 것이라고 예상합니다.
- b의 경우, unsorted가 정렬되지 않은 데이터이므로, Hash join을 사용할 것 같습니다. 또한 a보다 긴 실행시간을 가질 것 같습니다.

```
d2019320097=# Explain analyze select * from supplier s join table1 t on s.s_suppkey = t.sorted;
                                                          QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------
------
 Hash Join  (cost=442.00..229816.97 rows=143891 width=192) (actual time=3.713..7195.077 rows=50000 loops=1)
   Hash Cond: (t.sorted = s.s_suppkey)
   ->  Seq Scan on table1 t  (cost=0.00..203125.48 rows=9999748 width=53) (actual time=0.068..4135.603 rows=10000000 loo
ps=1)
   ->  Hash  (cost=317.00..317.00 rows=10000 width=139) (actual time=3.546..3.548 rows=10000 loops=1)
         Buckets: 16384  Batches: 1  Memory Usage: 1848kB
         ->  Seq Scan on supplier s  (cost=0.00..317.00 rows=10000 width=139) (actual time=0.009..1.215 rows=10000 loops
=1)
 Planning Time: 1.928 ms
 Execution Time: 7198.323 ms
(8개 행)
```

- a의 경우 실제로는 Hash join이 사용되었으며, 실행시간은 약 7198ms입니다.

```
d2019320097=# Explain analyze select * from supplier s join table1 t on s.s_suppkey = t.unsorted;
                                                          QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------
------
 Hash Join  (cost=442.00..229816.89 rows=55124 width=192) (actual time=3.942..7687.348 rows=50164 loops=1)
   Hash Cond: (t.unsorted = s.s_suppkey)
   ->  Seq Scan on table1 t  (cost=0.00..203125.48 rows=9999748 width=53) (actual time=0.099..4017.604 rows=10000000 loo
ps=1)
   ->  Hash  (cost=317.00..317.00 rows=10000 width=139) (actual time=3.572..3.573 rows=10000 loops=1)
         Buckets: 16384  Batches: 1  Memory Usage: 1848kB
         ->  Seq Scan on supplier s  (cost=0.00..317.00 rows=10000 width=139) (actual time=0.015..0.929 rows=10000 loops
=1)
 Planning Time: 0.331 ms
 Execution Time: 7691.675 ms
(8개 행)
```

- b의 경우에도 마찬가지로 Hash join이 사용되었으며, 실행시간은 약 7692ms입니다.
- a가 약간 더 빠르게 실행되지만, 둘 사이에 큰 시간 간격은 없습니다.

Exercise 4

- a는 정렬된 데이터를 가지고 있으므로, Merge Join을 사용할 것 같습니다.
  b는 정렬되지 않았으나, index를 가지고 있으므로 Nested Loop join을 사용할 것 같습니다.
  a는 정렬된 데이터와 인덱스 모두를 가지고 있으므로 b에 비해 매우 빠른 실행 시간을 가질 것입니다.
  b는 인덱스는 있으나, Nested Loop join을 수행하면서 실행시간이 길어질 것 같습니다.

```
d2019320097=# Explain analyze select * from supplier s join table1 t on s.s_suppkey = t.sorted;
                                                                  QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------
--------------------
 Merge Join  (cost=6.24..3640.76 rows=150017 width=192) (actual time=0.063..13.061 rows=50000 loops=1)
   Merge Cond: (s.s_suppkey = t.sorted)
   ->  Index Scan using suppkey_idx on supplier s  (cost=0.29..491.40 rows=10000 width=139) (actual time=0.025..1.059 ro
ws=10000 loops=1)
   ->  Index Scan using sorted_idx on table1 t  (cost=0.43..310074.58 rows=9999676 width=53) (actual time=0.032..6.747 r
ows=50006 loops=1)
 Planning Time: 2.645 ms
 Execution Time: 13.991 ms
(6개 행)
```

- a는 Merge join을 사용하며, Index scan을 통해서 약 14ms의 시간으로 빠르게 join이 일어남을 알 수 있습니다.

```
d2019320097=# Explain analyze select * from supplier s join table1 t on s.s_suppkey = t.unsorted;
                                                                    QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------
 Merge Join  (cost=0.82..4327.19 rows=58838 width=192) (actual time=0.132..2558.816 rows=50164 loops=1)
   Merge Cond: (s.s_suppkey = t.unsorted)
   ->  Index Scan using suppkey_idx on supplier s  (cost=0.29..491.40 rows=10000 width=139) (actual time=0.012..9.631 ro
ws=10000 loops=1)
   ->  Index Scan using unsorted_idx on table1 t  (cost=0.43..620097.52 rows=9999676 width=53) (actual time=0.042..2521.
267 rows=50168 loops=1)
 Planning Time: 0.542 ms
 Execution Time: 2563.702 ms
(6개 행)
```

- 마찬가지로 b도 Merge join을 사용합니다. 하지만 정렬되지 않은 데이터 특성 상 a보다 긴 실행 시간을 가지게 됩니다.

Exercise 5-1

- 앞서 4번 문제에서는 a와 b 모두 Merge join을 사용하였고, index scan을 통해서 각 테이블에 접근했었습니다.
  하지만 nationkey의 경우에는 인덱스를 가지고 있지 않아, seq scan을 통해서 접근해야 합니다. 이 경우 정렬된 sorted로 join하는 a는 Merge join을 사용할 것이며, b는 Hash join을 사용할 것 같습니다.
  a는 sorted가 정렬되어 있어 빠른 실행 시간을 가질 것이고, b는 unsorted가 정렬되어있지 않아 느린 실행 시간을 가질 것입니다.

```
d2019320097=# Explain analyze select * from supplier s join table1 t on s.s_nationkey = t.sorted;
                                                                  QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------
------------------
 Merge Join  (cost=981.82..3200.73 rows=147612 width=192) (actual time=3.757..26.271 rows=50000 loops=1)
   Merge Cond: (t.sorted = s.s_nationkey)
   ->  Index Scan using sorted_idx on table1 t  (cost=0.43..310085.66 rows=10000415 width=53) (actual time=0.065..0.233
rows=126 loops=1)
   ->  Sort  (cost=981.39..1006.39 rows=10000 width=139) (actual time=3.683..8.776 rows=49996 loops=1)
         Sort Key: s.s_nationkey
         Sort Method: quicksort  Memory: 2061kB
         ->  Seq Scan on supplier s  (cost=0.00..317.00 rows=10000 width=139) (actual time=0.019..0.900 rows=10000 loops
=1)
 Planning Time: 3.699 ms
 Execution Time: 30.042 ms
(9개 행)
```

- a는 Merge Join을 사용했으며, 약 30ms의 짧은 실행 시간을 가집니다.

```
d2019320097=# Explain analyze select * from supplier s join table1 t on s.s_nationkey = t.unsorted;
                                                                  QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------
--------------------
 Merge Join  (cost=981.82..1952.25 rows=64136 width=192) (actual time=3.473..11.779 rows=47954 loops=1)
   Merge Cond: (t.unsorted = s.s_nationkey)
   ->  Index Scan using unsorted_idx on table1 t  (cost=0.43..620105.69 rows=10000415 width=53) (actual time=0.053..1.38
1 rows=120 loops=1)
   ->  Sort  (cost=981.39..1006.39 rows=10000 width=139) (actual time=3.412..5.099 rows=47952 loops=1)
         Sort Key: s.s_nationkey
         Sort Method: quicksort  Memory: 2061kB
         ->  Seq Scan on supplier s  (cost=0.00..317.00 rows=10000 width=139) (actual time=0.017..0.914 rows=10000 loops
=1)
 Planning Time: 1.804 ms
 Execution Time: 12.932 ms
(9개 행)
```

- b 또한 Merge Join을 사용했으며, a 보다도 짧은 약 13ms의 실행 시간을 가집니다.

Exercise 5-2

- supplier에 엄청나게 많은 양의 데이터가 추가되어, 인덱스를 하나하나 검색하는 것 보다 seq scan으로 supplier 전체를 훑게 될 것 같습니다.
- 따라서 a, b 모두 Merge join보다는 Hash join을 사용하여 많은 양의 데이터를 처리하려고 할 것 같습니다.
- 하지만 처리해야 하는 데이터의 크기가 매우 커, a와 b a가 b에 비해 짧지만, 둘 모두 긴 실행 시간을 가지게 될 것입니다.

```
d2019320097=# Explain analyze select * from supplier s join table1 t on s.s_nationkey = t.sorted;
                                                                  QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------
------------
 Hash Join  (cost=425733.32..1395364.33 rows=12995684 width=190) (actual time=5857.360..18009.897 rows=9999995 loops=1)
   Hash Cond: (s.s_nationkey = t.sorted)
   ->  Seq Scan on supplier s  (cost=0.00..144412.80 rows=9999380 width=137) (actual time=0.035..1319.795 rows=10000000
loops=1)
   ->  Hash  (cost=203112.14..203112.14 rows=9998414 width=53) (actual time=5839.250..5839.252 rows=10000000 loops=1)
         Buckets: 131072  Batches: 128  Memory Usage: 7550kB
         ->  Seq Scan on table1 t  (cost=0.00..203112.14 rows=9998414 width=53) (actual time=0.072..2058.817 rows=100000
00 loops=1)
 Planning Time: 1.827 ms
 Execution Time: 18485.559 ms
(8개 행)
```

- a의 경우 Hash join을 사용했으며, 약 18486ms의 아주 긴 실행 시간을 가지게 되었습니다.

```
d2019320097=# Explain analyze select * from supplier s join table1 t on s.s_nationkey = t.unsorted;
                                                              QUERY PLAN
______________________________________________________________________________________________________________________________________
____________
 Hash Join  (cost=425733.32..1347930.54 rows=11996928 width=192) (actual time=6569.725..21491.305 rows=9997787 loops=1)
   Hash Cond: (s.s_nationkey = t.unsorted)
   ->  Seq Scan on supplier s  (cost=0.00..144420.58 rows=10000158 width=139) (actual time=0.030..1234.360 rows=10000000
 loops=1)
   ->  Hash  (cost=203112.14..203112.14 rows=9998414 width=53) (actual time=6550.201..6550.204 rows=10000000 loops=1)
         Buckets: 131072  Batches: 128  Memory Usage: 7607kB
         ->  Seq Scan on table1 t  (cost=0.00..203112.14 rows=9998414 width=53) (actual time=0.084..1962.084 rows=100000
00 loops=1)
 Planning Time: 0.352 ms
 Execution Time: 22011.836 ms
(8개 행)
```

- b의 경우에도 Hash join이 사용되었으며, 약 22012ms의 아주 긴 실행 시간을 가지게 되었습니다.