Lucas Bolanos
3/16/23
Lab9
CS166

Query.sql

```
lbola008@wch133-06 $ psql -h localhost -p $PGPORT "$USER"_DB < query.sql
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=33.72..7964.16 rows=181745 width=492) (actual time=0.668..136.873 rows=175000 loops=1)
   Hash Cond: (sales.itemid = catalog.itemid)
   ->  Hash Join  (cost=16.52..5447.97 rows=181745 width=270) (actual time=0.576..90.440 rows=175000 loops=1)
         Hash Cond: (sales.storeid = stores.storeid)
         ->  Seq Scan on sales  (cost=0.00..2932.45 rows=181745 width=20) (actual time=0.034..29.837 rows=175000 loops=1)
         ->  Hash  (cost=12.90..12.90 rows=290 width=250) (actual time=0.519..0.519 rows=1000 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 58kB
               ->  Seq Scan on stores  (cost=0.00..12.90 rows=290 width=250) (actual time=0.016..0.250 rows=1000 loops=1)
   ->  Hash  (cost=13.20..13.20 rows=320 width=222) (actual time=0.052..0.052 rows=50 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 3kB
         ->  Seq Scan on catalog  (cost=0.00..13.20 rows=320 width=222) (actual time=0.023..0.028 rows=50 loops=1)
 Total runtime: 143.443 ms
(12 rows)

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=44.53..1196.09 rows=627 width=492) (actual time=0.112..7.361 rows=3448 loops=1)
   Hash Cond: (sales.itemid = catalog.itemid)
   ->  Nested Loop  (cost=27.33..1170.27 rows=627 width=270) (actual time=0.095..6.683 rows=3448 loops=1)
         ->  Seq Scan on stores  (cost=0.00..13.62 rows=1 width=250) (actual time=0.017..0.112 rows=20 loops=1)
               Filter: ((state)::text = 'CA'::text)
               Rows Removed by Filter: 980
         ->  Bitmap Heap Scan on sales  (cost=27.33..1147.56 rows=909 width=20) (actual time=0.039..0.310 rows=172 loops=20)
               Recheck Cond: (storeid = stores.storeid)
               ->  Bitmap Index Scan on sales_pkey  (cost=0.00..27.10 rows=909 width=0) (actual time=0.025..0.025 rows=172 loops=20)
                     Index Cond: (storeid = stores.storeid)
   ->  Hash  (cost=13.20..13.20 rows=320 width=222) (actual time=0.012..0.012 rows=50 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 3kB
         ->  Seq Scan on catalog  (cost=0.00..13.20 rows=320 width=222) (actual time=0.003..0.007 rows=50 loops=1)
 Total runtime: 7.483 ms
(14 rows)
```

Index.sql

```
lbola008@wch133-06 $ psql -h localhost -p $PGPORT "$USER"_DB < index.sql
CREATE INDEX
CREATE INDEX
CREATE INDEX
/extra/lbola008/lab9/lab9
```

Query.sql

```
lbola008@wch133-06 $ psql -h localhost -p $PGPORT "$USER"_DB < query.sql
                                                   QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=32.62..7710.12 rows=175000 width=267) (actual time=0.671..141.881 rows=175000 loops=1)
   Hash Cond: (sales.itemid = catalog.itemid)
   ->  Hash Join  (cost=30.50..5301.75 rows=175000 width=45) (actual time=0.586..90.613 rows=175000 loops=1)
         Hash Cond: (sales.storeid = stores.storeid)
         ->  Seq Scan on sales  (cost=0.00..2865.00 rows=175000 width=20) (actual time=0.011..24.164 rows=175000 loops=1)
         ->  Hash  (cost=18.00..18.00 rows=1000 width=25) (actual time=0.550..0.550 rows=1000 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 58kB
               ->  Seq Scan on stores  (cost=0.00..18.00 rows=1000 width=25) (actual time=0.008..0.210 rows=1000 loops=1)
   ->  Hash  (cost=1.50..1.50 rows=50 width=222) (actual time=0.043..0.043 rows=50 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 3kB
         ->  Seq Scan on catalog  (cost=0.00..1.50 rows=50 width=222) (actual time=0.011..0.021 rows=50 loops=1)
 Total runtime: 149.233 ms
(12 rows)

                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=15.03..3619.41 rows=3500 width=267) (actual time=0.156..21.722 rows=3448 loops=1)
   Hash Cond: (sales.itemid = catalog.itemid)
   ->  Hash Join  (cost=12.91..3569.16 rows=3500 width=45) (actual time=0.131..21.097 rows=3448 loops=1)
         Hash Cond: (sales.storeid = stores.storeid)
         ->  Seq Scan on sales  (cost=0.00..2865.00 rows=175000 width=20) (actual time=0.004..9.984 rows=175000 loops=1)
         ->  Hash  (cost=12.66..12.66 rows=20 width=25) (actual time=0.111..0.111 rows=20 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 2kB
               ->  Bitmap Heap Scan on stores  (cost=4.41..12.66 rows=20 width=25) (actual time=0.099..0.107 rows=20 loops=1)
                     Recheck Cond: ((state)::text = 'CA'::text)
                     ->  Bitmap Index Scan on stateindex  (cost=0.00..4.40 rows=20 width=0) (actual time=0.094..0.094 rows=20 loops=1)
                           Index Cond: ((state)::text = 'CA'::text)
   ->  Hash  (cost=1.50..1.50 rows=50 width=222) (actual time=0.015..0.015 rows=50 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 3kB
         ->  Seq Scan on catalog  (cost=0.00..1.50 rows=50 width=222) (actual time=0.003..0.007 rows=50 loops=1)
 Total runtime: 21.857 ms
(15 rows)
```

The EXPLAIN ANALYZE clauses to our queries we get the query plan and time the optimizer choose and calculates for a particular query.

The first query requests all tuples from Stores while the second query requests tuples whose state is in California. This requires the query to filter the data based on state.

Before indexes are introduced:

- Uses a sequential scan on the dataset for the first query. The join is a hash join which is sequentially scanned (record by record/entire table) and hashes the stores records to generate another hash table for the storeID. Then it cross references this to check where stores.storeid is equal to sales.storeid.
- Requires a filter to be applied on the dataset for the states for the second query. No index is available so the optimizer uses a sequential scan.

After indexes are introduced:

- Similar to the previous query a sequential scan is performed since all tuples/records are being requested again.
- The index provides the records of stores with state IDs equal to CA. It does a bitmap index scan instead of a sequential scan, and then a bitmap heap scan in order to only retrieve the pages containing records of stores with their state equal to CA. The child

operation is done first followed by the parent operation. Bitmap index scan helps it find the rows containing the information you need, reading them instead of reading the entire rows. Bitmap heap scan reads the rows found by the index scan, following the index condition.

The joins effect the queries, hash joins/nested joins. Hash join there is two tables being joined based on a condition. The tuples are joined if condition holds. The hash join will use hashing, and take that hash for the inner table (smaller table).Inner table after being hashed and outer table mashed. The inner loop entries are being hashed and put into buckets. Then a sequential scan on sales is followed, then goes to sales entries and does a hash join which references the inner loop hash with the outer loop hash.

lbola008@wch133-06 $ psql -h localhost -p $PGPORT "$USER"_DB < query.sql

QUERY PLAN

Hash Join (cost=47.70..7725.20 rows=175000 width=267) (actual time=0.652..140.273 rows=175000 loops=1)

((Actual join: joins where two hashed tables together))

Hash Cond: (sales.itemid = catalog.itemid)

((Checks where join condition is met for sales and catalog table on itemid))

-> Hash Join (cost=30.50..5301.75 rows=175000 width=45) (actual time=0.559..89.678 rows=175000 loops=1)

((Joins the tables sales and stores together if hash condition holds))

Hash Cond: (sales.storeid = stores.storeid)

((Checks where join condition is met for sales and stores tables))

-> Seq Scan on sales (cost=0.00..2865.00 rows=175000 width=20) (actual time=0.025..23.891 rows=175000 loops=1)

((Scans through entire sales table with sequential scan.))

-> Hash (cost=18.00..18.00 rows=1000 width=25) (actual time=0.501..0.501 rows=1000 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 58kB

((Hashes the store records to create a hash table on storeide))

-> Seq Scan on stores (cost=0.00..18.00 rows=1000 width=25) (actual time=0.009..0.191 rows=1000 loops=1)

((Scans through entire stores table with sequential scan.))

-> Hash (cost=13.20..13.20 rows=320 width=222) (actual time=0.048..0.048 rows=50 loops=1)

Buckets: 1024 Batches: 1 Memory Usage: 3kB
((Hashes the catalog records to create a hash table on storeid))
-> Seq Scan on catalog (cost=0.00..13.20 rows=320 width=222) (actual time=0.015..0.022 rows=50 loops=1)
((Scans through entire catalog table with sequential scan))
Total runtime: 147.630 ms
(12 rows)

QUERY PLAN (where)

Hash Join (cost=37.95..3642.32 rows=3500 width=267) (actual time=0.172..27.155 rows=3448 loops=1)
((Joins tables together if hash condition holds))
Hash Cond: (sales.itemid = catalog.itemid)
((Checks hash condition to see if sales.itemid = catalog.itemid))
-> Hash Join (cost=20.75..3577.00 rows=3500 width=45) (actual time=0.153..26.408 rows=3448 loops=1)
((Joins the two tables if the hash condition holds))
Hash Cond: (sales.storeid = stores.storeid)
((Checks hash condition to see is sales.storeid = stores.storeid))
-> Seq Scan on sales (cost=0.00..2865.00 rows=175000 width=20) (actual time=0.015..13.184 rows=175000 loops=1)
((Sequentially scans entire sales table))
-> Hash (cost=20.50..20.50 rows=20 width=25) (actual time=0.126..0.126 rows=20 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 2kB
((Hashes the filtered stores table on storeid))
-> Seq Scan on stores (cost=0.00..20.50 rows=20 width=25) (actual time=0.017..0.123 rows=20 loops=1)
Filter: ((state)::text = 'CA'::text)
Rows Removed by Filter: 980
((Sequentially scans entire stores table, filtering for wherever state = ‘CA’))
-> Hash (cost=13.20..13.20 rows=320 width=222) (actual time=0.014..0.014 rows=50 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 3kB
((Hashes the catalog table on itemid))
-> Seq Scan on catalog (cost=0.00..13.20 rows=320 width=222) (actual time=0.003..0.006 rows=50 loops=1)

((Sequentially scans entire catalog table))
Total runtime: 27.302 ms
(14 rows)

AFTER INDEXES

```
 QUERY PLAN                (indexed)
------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=32.62..7710.12 rows=175000 width=267) (actual time=0.662..137.973
rows=175000 loops=1)
((If hash condition holds true joins sales and catalog together.))
   Hash Cond: (sales.itemid = catalog.itemid)
((Checks if hash condition sales-itemid = catalog.itemid))
   ->  Hash Join  (cost=30.50..5301.75 rows=175000 width=45) (actual time=0.582..88.381
rows=175000 loops=1)
((Joins sales/stores together if hash condition holds true))
         Hash Cond: (sales.storeid = stores.storeid)
((Checks hash condition to see if sales.storeid = stores.storeid))
         ->  Seq Scan on sales  (cost=0.00..2865.00 rows=175000 width=20) (actual
time=0.060..23.731 rows=175000 loops=1)
((Sequentially scans entire sales table))
         ->  Hash  (cost=18.00..18.00 rows=1000 width=25) (actual time=0.493..0.493 rows=1000
loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 58kB
((Hashes stores table data on storeid))
               ->  Seq Scan on stores  (cost=0.00..18.00 rows=1000 width=25) (actual
time=0.010..0.207 rows=1000 loops=1)
((Sequentially scans entire stores table))
   ->  Hash  (cost=1.50..1.50 rows=50 width=222) (actual time=0.041..0.041 rows=50 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 3kB
((Hashes catalog table data on itemid))
         ->  Seq Scan on catalog  (cost=0.00..1.50 rows=50 width=222) (actual time=0.011..0.017
rows=50 loops=1)
((Sequentially scans entire catalog table))
 Total runtime: 144.954 ms
(12 rows)
```

```
                                                QUERY PLAN   (indexed) (where)
------------------------------------------------------------------------------------------------------------------------
--------
 Hash Join  (cost=15.03..3619.41 rows=3500 width=267) (actual time=0.138..26.532 rows=3448
loops=1)
((Joins the tables sales and catalog if hash condition holds true))
   Hash Cond: (sales.itemid = catalog.itemid)
((Checks if hash condition sales.itemid = catalog.itemid))
```

-> Hash Join (cost=12.91..3569.16 rows=3500 width=45) (actual time=0.114..25.794 rows=3448 loops=1)
((Joins sales and stores tables if hash condition holds true))
Hash Cond: (sales.storeid = stores.storeid)
((Checks if hash condition sales.storeid = stores.storeid))
-> Seq Scan on sales (cost=0.00..2865.00 rows=175000 width=20) (actual time=0.017..13.021 rows=175000 loops=1)
((Sequentially scans through entire sales table))
-> Hash (cost=12.66..12.66 rows=20 width=25) (actual time=0.078..0.078 rows=20 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 2kB
((Previous pages retrieved by index scan are hashed on storeid))
-> Bitmap Heap Scan on stores (cost=4.41..12.66 rows=20 width=25) (actual time=0.066..0.075 rows=20 loops=1)
Recheck Cond: ((state)::text = 'CA'::text)

((Bitmap heap scan is performed which goes into the original stores table and retrieves only pages that contain one or more records with state = ‘CA’))
-> Bitmap Index Scan on stateindex (cost=0.00..4.40 rows=20 width=0) (actual time=0.061..0.061 rows=20 loops=1)
Index Cond: ((state)::text = 'CA'::text)
((Performs bitmap index scan to identify pages needed from index ‘stateindex’. Filters by the index condition of state = ‘CA”))
-> Hash (cost=1.50..1.50 rows=50 width=222) (actual time=0.016..0.016 rows=50 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 3kB
((Hashes catalog table data on itemid))
-> Seq Scan on catalog (cost=0.00..1.50 rows=50 width=222) (actual time=0.003..0.006 rows=50 loops=1)
((Sequentially scans entire catalog table))
Total runtime: 26.653 ms
(15 rows)