CS166 Section 22 Alejandro Sherman

Lab 9 Report Results:

First I logged into my designated lab machine - wch 133-38.
Next I copied over the new and needed files from the lab 9 zip folder to the /tmp folder.
Then I ran source ./startPostgreSQL.sh inside the /tmp folder
Next I ran source ./createPostgreDB.sh

Next I ran the given commands:
cp *.csv /tmp/$USER/myDB/data/
export USERNAME_DB=$USER"_DB"
psql -h localhost -p $PGPORT $USERNAME_DB < schema.sql
psql -h localhost -p $PGPORT $USERNAME_DB < insert.sql

Next I ran
psql -h localhost -p $PGPORT $USERNAME_DB < query.sql
Result:

```
/tmp
asher011@wch133-38 $ psql -h localhost -p $PGPORT $USERNAME_DB < query.sql
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=53.52..7983.96 rows=181745 width=180) (actual time=0.755..149.595 rows=175000 loops=1)
   Hash Cond: (sales.itemid = catalog.itemid)
   ->  Hash Join  (cost=25.52..5456.97 rows=181745 width=108) (actual time=0.642..99.459 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=0.041..33.298 rows=175000
loops=1)
         ->  Hash  (cost=16.90..16.90 rows=690 width=88) (actual time=0.577..0.577 rows=1000 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 58kB
               ->  Seq Scan on stores  (cost=0.00..16.90 rows=690 width=88) (actual time=0.019..0.262 rows=1000
loops=1)
   ->  Hash  (cost=18.00..18.00 rows=800 width=72) (actual time=0.057..0.057 rows=50 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 3kB
         ->  Seq Scan on catalog  (cost=0.00..18.00 rows=800 width=72) (actual time=0.026..0.035 rows=50 loops=1)
 Total runtime: 156.619 ms
(12 rows)

                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=55.33..2731.00 rows=790 width=180) (actual time=0.126..8.753 rows=3448 loops=1)
   Hash Cond: (sales.itemid = catalog.itemid)
   ->  Nested Loop  (cost=27.33..2692.14 rows=790 width=108) (actual time=0.106..7.963 rows=3448 loops=1)
         ->  Seq Scan on stores  (cost=0.00..18.62 rows=3 width=88) (actual time=0.018..0.129 rows=20 loops=1)
               Filter: ((state)::text = 'CA'::text)
               Rows Removed by Filter: 980
         ->  Bitmap Heap Scan on sales  (cost=27.33..882.08 rows=909 width=20) (actual time=0.048..0.366 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=0.031..0.031
rows=172 loops=20)
```

```
                  Index Cond: (storeid = stores.storeid)
  ->  Hash  (cost=18.00..18.00 rows=800 width=72) (actual time=0.015..0.015 rows=50 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 3kB
        ->  Seq Scan on catalog  (cost=0.00..18.00 rows=800 width=72) (actual time=0.004..0.008 rows=50 loops=1)
 Total runtime: 8.893 ms
(14 rows)
```

Now I added indexes:
psql -h localhost -p $PGPORT $USERNAME_DB < index.sql
Result:

```
/tmp
asher011@wch133-38 $ psql -h localhost -p $PGPORT $USERNAME_DB < index.sql
CREATE INDEX
CREATE INDEX
CREATE INDEX
/tmp
asher011@wch133-38 $
```

Next I ran the query again, this time with indexes active
psql -h localhost -p $PGPORT $USERNAME_DB < query.sql
Result:

```
/tmp
asher011@wch133-38 $ psql -h localhost -p $PGPORT $USERNAME_DB < query.sql
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=32.62..7710.12 rows=175000 width=117) (actual time=0.646..141.392 rows=175000 loops=1)
   Hash Cond: (sales.itemid = catalog.itemid)
   ->  Hash Join  (cost=30.50..5301.75 rows=175000 width=45) (actual time=0.556..90.286 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..23.850 rows=175000
loops=1)
         ->  Hash  (cost=18.00..18.00 rows=1000 width=25) (actual time=0.525..0.525 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..0.208 rows=1000
loops=1)
   ->  Hash  (cost=1.50..1.50 rows=50 width=72) (actual time=0.040..0.040 rows=50 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 3kB
         ->  Seq Scan on catalog  (cost=0.00..1.50 rows=50 width=72) (actual time=0.010..0.017 rows=50 loops=1)
 Total runtime: 148.681 ms
(12 rows)

                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=15.03..3619.41 rows=3500 width=117) (actual time=0.140..26.167 rows=3448 loops=1)
   Hash Cond: (sales.itemid = catalog.itemid)
   ->  Hash Join  (cost=12.91..3569.16 rows=3500 width=45) (actual time=0.116..25.409 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..11.834 rows=175000
loops=1)
         ->  Hash  (cost=12.66..12.66 rows=20 width=25) (actual time=0.095..0.095 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=0.081..0.089 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=0.076..0.076
rows=20 loops=1)
                        Index Cond: ((state)::text = 'CA'::text)
  ->  Hash  (cost=1.50..1.50 rows=50 width=72) (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=72) (actual time=0.003..0.007 rows=50 loops=1)
 Total runtime: 26.311 ms
(15 rows)
```

Through examination of the difference of each Query Plan from before and after the index creation, I have come to a few realizations about the difference between the two.

Firstly, for the first Query Plan, when indexes were introduced, the query plan overall reduced the cost of hash joins, and overall used less rows and had smaller width, resulting in less execution time. Additionally, for the first query, adding indexes didn't change much in terms of the execution of the Plan as the same sequence of instructions were used, just with faster finding of the information with index use.

As for the second Query Plan, when indexes were introduced, the overall runtime actually increased as the Plan performed extra steps that the original query plan without indexes didn't. This new query plan has an extra Hash Cond and Hash Join as opposed to the original plan's Nested Loop. Additionally, the original Query Plan without index use had the filter use here:
Filter: ((state)::text = 'CA'::text)
Rows Removed by Filter: 980
Allowing data to be thrown out, which was not present in the updated plan. As such, for the second Query, when indexes were introduced, more time was spent on the Plan as the indexes caused the plan to waste more time.

I then ran source ./stopPostgreDB.sh to shut down the database.

In elearn I will submit this report.