

Chapter 16 – Lab Solution

Query Optimization 2

Exercise 1.a Answer

- SELECT histogram_bounds FROM pg_stats WHERE tablename='supplier' AND attname='s_suppkey';

```
postgres=# SELECT histogram_bounds FROM pg_stats WHERE tablename='supplier' AND attname='s_suppkey';

 histogram_bounds
-----
{1,100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,2100,2200,2300,2400,2500,2600,2700,2800,2900,3000,3100,3200,3300,3400,3500,3600,3700,3800,3900,4000,4100,4200,4300,4400,4500,4600,4700,4800,4900,5000,5100,5200,5300,5400,5500,5600,5700,5800,5900,6000,6100,6200,6300,6400,6500,6600,6700,6800,6900,7000,7100,7200,7300,7400,7500,7600,7700,7800,7900,8000,8100,8200,8300,8400,8500,8600,8700,8800,8900,9000,9100,9200,9300,9400,9500,9600,9700,9800,9900,10000}
(1개 행)
```

- $$\text{Selectivity} = (3 + (350 - \text{bucket4.min}) / (\text{bucket4.max} - \text{bucket4.min})) / \text{num_bucket}$$
$$= (3 + (350 - 300) / (400 - 300)) / 100$$
$$= 0.035$$
- Estimated result= 10000 x 0.035 = 350
- Actual result:

```
postgres=# SELECT count(*) FROM supplier WHERE s_suppkey<=350;
 count
-----
    350
(1개 행)
```

Exercise 1.b Answer

- SELECT histogram_bounds FROM pg_stats WHERE tablename='supplier' AND attname='s_acctbal';

```
postgres=# SELECT histogram_bounds FROM pg_stats WHERE tablename='supplier' AND attname='s_acctbal';
```

histogram_bounds
{-998.22,-889.12,-789.3,-687.56,-562.38,-459.62,-338.96,-229.07,-112.15,4.83,102.96,214.42,322.65,433.93,540.5,655.4,766.64,882.86,992.69,1102.72,1216.06,1308.49,1417.33,1535.08,1636.13,1765.93,1890.85,2004.44,2110.8,2217.2,2315.87,2435.34,2543.89,2663.51,2777.91,2882.23,2989.28,3094.73,3187.71,3295.4,3423.9,3519.47,3616.49,3730.66,3844.28,3958.63,4097.83,4201.05,4320.26,4403.51,4535.43,4642.64,4754.93,4885.04,4988.55,5081.7,5185.12,5277.41,5389.53,5490.95,5597.26,5706.22,5809.62,5926.41,6024.5,6127.58,6240.69,6372.14,6479.49,6587.12,6716.33,6820.97,6940.25,7047.29,7148.05,7258.51,7364.3,7465.41,7566.99,7675.2,7788.05,7876.55,7992.38,8104.84,8210.13,8335.31,8457.09,8569.52,8677.75,8782.52,8878.97,9001.17,9093.75,9198.31,9312.63,9449.33,9542.91,9643.55,9751.45,9858.45,9999.72}

(1개 행)

- Histogram selectivity= $(12 + (405.68 - \text{bucket13.min}) / (\text{bucket13.max} - \text{bucket13.min})) / \text{num_bucket}$
 $= (12 + (405.68 - 322.65) / (433.93 - 322.65)) / 100$
 ≈ 0.127461359

Exercise 1.b Answer

- SELECT most_common_vals, most_common_freqs FROM pg_stats WHERE tablename='supplier' AND attname='s_acctbal';

[illegible]

- Most common values selectivity= $\text{sum}(\text{most common frequencies})$
 $= 0.0002 + 0.0002 + 0.0002 + 0.0002 + 0.0002 = 0.001$
- Selectivity= most common values selectivity + histogram selectivity x histogram fraction
 $= 0.001 + 0.127461359 \times 0.991 = 0.127314206769$
- Estimated result= $10000 \times 0.127314206769 \approx 1273$
- Actual result:

```
postgres=# SELECT count(*) FROM supplier WHERE s_acctbal<=405.68;
```

```
postgres=# SELECT count(*) FROM supplier WHERE s_acctbal<=405.68;
count
-----
1273
(1개 행)
```

Exercise 2.a Answer

- EXPLAIN ANALYZE SELECT * FROM supplier WHERE s_suppkey<=350;
- Since the selectivity of the condition is sufficiently low to use index, the best query plan is as follows

```
postgres=# EXPLAIN ANALYZE SELECT * FROM supplier WHERE s_suppkey<=350;
               QUERY PLAN
-----
Index Scan using supplier_pkey on supplier  (cost=0.29..26.73 rows=350 width=139) (actual time=0.046..0.133 rows=350 loops=1)
  Index Cond: (s_suppkey <= 350)
Planning Time: 0.180 ms
Execution Time: 0.192 ms
(4개 행)
```

Exercise 2.b Answer

- EXPLAIN ANALYZE SELECT * FROM supplier WHERE s_suppkey>350;
- Since the selectivity of the condition is not sufficiently low to use index, the best query plan is as follows

```
postgres=# EXPLAIN ANALYZE SELECT * FROM supplier WHERE s_suppkey>350;
               QUERY PLAN
-----
Seq Scan on supplier (cost=0.00..342.00 rows=9650 width=139) (actual time=0.030..0.768 rows=9650 loops=1)
  Filter: (s_suppkey > 350)
  Rows Removed by Filter: 350
Planning Time: 0.066 ms
Execution Time: 0.919 ms
(5개 행)
```

Exercise 3 Answer

- Join the tables as small as possible

```
postgres=# EXPLAIN ANALYZE SELECT count(*) FROM t1 NATURAL JOIN t2 NATURAL JOIN t3 NATURAL JOIN t4;
               QUERY PLAN
-----
Aggregate  (cost=1332596.13..1332596.14 rows=1 width=8) (actual time=8089.816..8089.818 rows=1 loops=1)
-> Hash Join (cost=3357.25..1095422.78 rows=94869343 width=0) (actual time=16.182..5510.752 rows=98105001 loops=1)
    Hash Cond: (t1.val = t4.val)
-> Hash Join (cost=273.25..1414.51 rows=95129 width=12) (actual time=1.308..12.744 rows=98561 loops=1)
    Hash Cond: (t1.val = t3.val)
-> Hash Join (cost=3.25..35.32 rows=957 width=8) (actual time=0.036..0.616 rows=975 loops=1)
    Hash Cond: (t2.val = t1.val)
-> Seq Scan on t2 (cost=0.00..15.00 rows=1000 width=4) (actual time=0.013..0.222 rows=1000 loops=1)
-> Hash (cost=2.00..2.00 rows=100 width=4) (actual time=0.018..0.019 rows=100 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 12kB
-> Seq Scan on t1 (cost=0.00..2.00 rows=100 width=4) (actual time=0.005..0.009 rows=100 loops=1)
-> Hash (cost=145.00..145.00 rows=10000 width=4) (actual time=1.228..1.229 rows=10000 loops=1)
    Buckets: 16384 Batches: 1 Memory Usage: 480kB
-> Seq Scan on t3 (cost=0.00..145.00 rows=10000 width=4) (actual time=0.005..0.456 rows=10000 loops=1)
-> Hash (cost=1443.00..1443.00 rows=100000 width=4) (actual time=14.353..14.353 rows=100000 loops=1)
    Buckets: 131072 Batches: 2 Memory Usage: 2676kB
-> Seq Scan on t4 (cost=0.00..1443.00 rows=100000 width=4) (actual time=0.007..4.553 rows=100000 loops=1)
Planning Time: 0.287 ms
Execution Time: 8090.092 ms
(19개 행)
```

t1 -> t2 -> t3 -> t4

```
postgres=# EXPLAIN ANALYZE SELECT count(*) FROM t4 NATURAL JOIN t3 NATURAL JOIN t2 NATURAL JOIN t1;
               QUERY PLAN
-----
Aggregate  (cost=1391623.24..1391623.25 rows=1 width=8) (actual time=8160.073..8160.076 rows=1 loops=1)
-> Hash Join (cost=3131.28..1145067.01 rows=98622491 width=0) (actual time=15.085..5571.102 rows=98105001 loops=1)
    Hash Cond: (t3.val = t4.val)
-> Hash Join (cost=47.28..1427.12 rows=99698 width=12) (actual time=0.359..12.866 rows=98561 loops=1)
    Hash Cond: (t3.val = t2.val)
-> Seq Scan on t3 (cost=0.00..145.00 rows=10000 width=4) (actual time=0.017..0.893 rows=10000 loops=1)
-> Hash (cost=35.32..35.32 rows=957 width=8) (actual time=0.332..0.333 rows=975 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 47kB
-> Hash Join (cost=3.25..35.32 rows=957 width=8) (actual time=0.035..0.234 rows=975 loops=1)
    Hash Cond: (t2.val = t1.val)
-> Seq Scan on t2 (cost=0.00..15.00 rows=1000 width=4) (actual time=0.008..0.062 rows=1000 loops=1)
-> Hash (cost=2.00..2.00 rows=100 width=4) (actual time=0.020..0.021 rows=100 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 12kB
-> Seq Scan on t1 (cost=0.00..2.00 rows=100 width=4) (actual time=0.005..0.009 rows=100 loops=1)
-> Hash (cost=1443.00..1443.00 rows=100000 width=4) (actual time=14.353..14.354 rows=100000 loops=1)
    Buckets: 131072 Batches: 2 Memory Usage: 2748kB
-> Seq Scan on t4 (cost=0.00..1443.00 rows=100000 width=4) (actual time=0.008..4.577 rows=100000 loops=1)
Planning Time: 0.319 ms
Execution Time: 8160.305 ms
(19개 행)
```

t1 -> t2 -> t3 -> t4

Exercise 4 Answer

- The execution time difference a lot, according to the join order.

```
postgres=# SET join_collapse_limit=1;
SET
postgres=# EXPLAIN ANALYZE SELECT count(*) FROM t1 NATURAL JOIN t2 NATURAL JOIN t3 NATURAL JOIN t4;
               QUERY PLAN
-----
Aggregate  (cost=1332596.13..1332596.14 rows=1 width=8) (actual time=8197.409..8197.412 rows=1 loops=1)
  -> Hash Join (cost=3357.25..1095422.78 rows=94869343 width=0) (actual time=16.517..5422.029 rows=98105001 loops=1)
    Hash Cond: (t1.val = t4.val)
    -> Hash Join (cost=273.25..1414.51 rows=95129 width=12) (actual time=1.323..13.182 rows=98561 loops=1)
      Hash Cond: (t1.val = t3.val)
      -> Hash Join (cost=3.25..35.32 rows=957 width=8) (actual time=0.043..0.747 rows=975 loops=1)
        Hash Cond: (t2.val = t1.val)
        -> Seq Scan on t2 (cost=0.00..15.00 rows=1000 width=4) (actual time=0.014..0.273 rows=1000 loops=1)
        -> Hash (cost=2.00..2.00 rows=100 width=4) (actual time=0.024..0.025 rows=100 loops=1)
          Buckets: 1024 Batches: 1 Memory Usage: 12kB
          -> Seq Scan on t1 (cost=0.00..2.00 rows=100 width=4) (actual time=0.005..0.009 rows=100 loops=1)
      -> Hash (cost=145.00..145.00 rows=10000 width=4) (actual time=1.242..1.242 rows=10000 loops=1)
        Buckets: 16384 Batches: 1 Memory Usage: 480kB
        -> Seq Scan on t3 (cost=0.00..145.00 rows=10000 width=4) (actual time=0.005..0.512 rows=10000 loops=1)
    -> Hash (cost=1443.00..1443.00 rows=100000 width=4) (actual time=14.660..14.661 rows=100000 loops=1)
      Buckets: 131072 Batches: 2 Memory Usage: 2676kB
      -> Seq Scan on t4 (cost=0.00..1443.00 rows=100000 width=4) (actual time=0.006..4.756 rows=100000 loops=1)
Planning Time: 0.124 ms
Execution Time: 8197.718 ms
(13개 행)
```

```
postgres=# EXPLAIN ANALYZE SELECT count(*) FROM t4 NATURAL JOIN t3 NATURAL JOIN t2 NATURAL JOIN t1;
               QUERY PLAN
-----
Aggregate  (cost=3367017.48..3367017.49 rows=1 width=8) (actual time=19671.700..19671.703 rows=1 loops=1)
  -> Hash Join (cost=300.75..3120461.25 rows=98622491 width=0) (actual time=2.425..16891.088 rows=98105001 loops=1)
    Hash Cond: (t4.val = t1.val)
    -> Hash Join (cost=297.50..1392539.93 rows=98892421 width=12) (actual time=1.415..7420.991 rows=98860217 loops=1)
      Hash Cond: (t4.val = t2.val)
      -> Hash Join (cost=270.00..117664.67 rows=9945167 width=8) (actual time=1.269..876.216 rows=9940576 loops=1)
        Hash Cond: (t4.val = t3.val)
        -> Seq Scan on t4 (cost=0.00..1443.00 rows=100000 width=4) (actual time=0.006..9.246 rows=100000 loops=1)
        -> Hash (cost=145.00..145.00 rows=10000 width=4) (actual time=1.226..1.228 rows=10000 loops=1)
          Buckets: 16384 Batches: 1 Memory Usage: 480kB
          -> Seq Scan on t3 (cost=0.00..145.00 rows=10000 width=4) (actual time=0.006..0.461 rows=10000 loops=1)
      -> Hash (cost=15.00..15.00 rows=1000 width=4) (actual time=0.141..0.141 rows=1000 loops=1)
        Buckets: 1024 Batches: 1 Memory Usage: 44kB
        -> Seq Scan on t2 (cost=0.00..15.00 rows=1000 width=4) (actual time=0.007..0.061 rows=1000 loops=1)
    -> Hash (cost=2.00..2.00 rows=100 width=4) (actual time=0.026..0.026 rows=100 loops=1)
      Buckets: 1024 Batches: 1 Memory Usage: 12kB
      -> Seq Scan on t1 (cost=0.00..2.00 rows=100 width=4) (actual time=0.011..0.015 rows=100 loops=1)
Planning Time: 0.147 ms
Execution Time: 19671.793 ms
(13개 행)
```