Query Processing and Optimization Notes

John A. Miller

Department of Computer Science

University of Georgia

July 25, 2020

1 Bank Schema

1. **table** customer (cname, street, ccity) – 1000 3-tuples

$$n_c = 1000 \text{ tuples}, s_c = 400 \text{ bytes}$$

2. **table** deposit (accno, balance, *cname*, *bname*) – 2000 4-tuples

$$n_d = 2000 \text{ tuples}, s_d = 400 \text{ bytes}$$

3. **table** branch (bname, assets, bcity) – 10 3-tuples

$$n_b = 10 \text{ tuples}, s_b = 200 \text{ bytes}$$

4. **table** loan (loanno, amount, *cname*, *bname*) – 3000 4-tuples

$$n_l = 3000 \text{ tuples}, s_l = 400 \text{ bytes}$$

Convention: Primary Key (PK): underline, Foreign Key (FK): italics

2 Steps in Query Processing

1. Translate SQL

```
select c.cname, c.ccity
from customer c, deposit d
where d.bname = 'Alps' and c.cname = d.cname
```

2. to Relational Algebra (RA)

$$\pi_{c.cname,\, c.ccity}(\sigma_{bname=\text{`}Alps' \text{ and } c.cname=d.cname}(customer \times deposit))$$

3. For each of the tables in the query, compute the **blocking factor** bf and **number of blocks** nb.

$$bf_c = \lfloor \frac{s_{block}}{s_c} \rfloor = \lfloor \frac{4096}{400} \rfloor = 10 \text{ tuples per block}$$

$$nb_c = \lceil \frac{n_c}{bf_c} \rceil = \lceil \frac{1000}{10} \rceil = 100 \text{ blocks}$$

$$bf_d = \lfloor \frac{s_{block}}{s_d} \rfloor = \lfloor \frac{4096}{400} \rfloor = 10 \text{ tuples per block}$$

$$nb_d = \lceil \frac{n_d}{bf_d} \rceil = \lceil \frac{2000}{10} \rceil = 200 \text{ blocks}$$

4. Optimize the Query

3 Steps in Query Optimization

1. Direct Transalation. Arrange RA into a Relational Algebra Expression Tree

2. Convert Cartesian Product $\times$ to Join $\bowtie$

$$\pi_{c.cname,\,c.ccity}(\sigma_{bname=`Alps'}(customer \bowtie_{cname} deposit))$$

   This form of equi-join

$$customer \bowtie_{cname} deposit$$

   is equivalent to

$$customer \bowtie_{c.cname=d.cname} deposit$$

3. Move Select Operations σ Down the Tree

$$\pi_{c.cname,\,c.ccity}(customer \bowtie_{cname} (\sigma_{bname=`Alps'}(deposit)))$$

4. Apply Indexes to speed up select and join operations

```
create index d_bname_idx on deposit (bname)
create unique index c_cname_idx on customer (cname)
```

Note: an index on customer (cname) will likely be created by default in most DBMSs.

4 Query Cost Metrics

1. Sum of all output sizes (number of tuples)

   Tuple oriented, rough estimate

2. Sum of all block accesses required to process query

   Block oriented, better estimate for secondary storage uses number of block accesses (nba)

3. Run Time

   Empirical, e.g., Project 3

5 Result Sizes in Tuples (Bottom Up Evaluation)

Ignore the last project operation.

1. **Direct Translation**

$$\pi_{c.cname,\, c.ccity}(\sigma_{bname=\text{`}Alps' \text{ and } c.cname=d.cname}(customer \times deposit))$$

Cartesian Product $\times$ pairs every tuple in c with every tuple in d.

$$\text{Size of Product} = n_c * n_d = 1000 * 2000 = 2,000,000$$

The Select σ has two predicates that may be thought of as filters. The filter strength (or selectivity) for $\sigma_{A=v}(r)$ is the reciprocal of the number of distinct values for attribute A in relation r.

$$\frac{1}{V(A,r)}$$

$$\text{Size of Select} = \frac{n_c * n_d}{V(bname,d) * n_c} = \frac{1000 * 2000}{10 * 1000} = 200$$

total = 2,000,000 + 200 = 2,000,200

2. **Convert Cartesian Product $\times$ to Join $\bowtie$**

$$\pi_{c.cname,\, c.ccity}(\sigma_{bname=`Alps'}(customer \bowtie_{cname} deposit))$$

When a join predicate/condition is found as part of a top-level conjunction, it may be removed from a selection operation to be combined with a Cartesian Product operation to form a join. In this case the join predicate (i.e., predicates involving attributes from two tables) is $c.cname = d.cname$.

The Join $\bowtie$ pairs each tuple in d with its unique match in relation c. The size (in tuples) of the output of an equi-join of the form PK = FK is the size of the FK table.

$$\text{Size of Join} = n_d = 2000$$

Now the select σ has a single predicate and filter.

$$\text{Size of Select} = \frac{n_d}{V(bname, d)} = \frac{2000}{10} = 200$$

total = 2000 + 200 = 2200

3. **Move Select Operations σ Down the Tree**

$$\pi_{c.cname,\, c.ccity}(customer \bowtie_{cname} (\sigma_{bname=\text{`}Alps'}(deposit)))$$

The select σ has a single predicate (filter) and now pulls tuples directly from relation d

$$\text{Size of Select} = n'_d = \frac{n_d}{V(bname, d)} = \frac{2000}{10} = 200$$

The c table and d' table (what remains of d after selection) are now joined. The c relation is still the PK table, so the size of the output is the size relation d'.

$$\text{Size of Join} = n'_d = 200$$

total = 200 + 200 = 400

4. **Apply Indexes**

Improvements not visible with simple cost model.

6 Number of Block Accesses (nba)

Ignore the last project operation.

1. **Direct Translation**

$$\pi_{c.cname,\, c.ccity}(\sigma_{bname=\text{`}Alps' \text{ and } c.cname=d.cname}(customer \times deposit))$$

The **Cartesian Product** $\times$ pairs every tuple in c with every tuple in d.

$$\text{reads} = nb_c + nb_c \cdot nb_d \qquad = 100 + 100 \cdot 200 = 20,100$$

$$\text{writes} = nb_{\bowtie} = \lceil \frac{n_c \cdot n_d}{bf_{\bowtie}} \rceil \qquad = \lceil \frac{1000 \cdot 2000}{5} \rceil = 400,000$$

The **Select** σ has two predicates that may be thought of as filters. The filter strength (or selectivity) for $\sigma_{A=v}(r)$ is the reciprocal of the number of distinct values for attribute A in relation r.

$$\frac{1}{V(A,r)}$$

$$\text{reads} = nb_{\bowtie} \qquad = 400,000$$

$$\text{writes} = \lceil \frac{n_c \cdot n_d}{V(bname,d) \cdot n_c \cdot bf_{\bowtie}} \rceil \qquad = \lceil \frac{1000 \cdot 2000}{10 \cdot 1000 \cdot 5} \rceil = 40$$

total = 20,100 + 400,000 + 400,000 + 40 = 820,140

2. **Convert Cartesian Product $\times$ to Join $\bowtie$**

$$\pi_{c.cname,\, c.ccity}(\sigma_{bname=\text{‘}Alps\text{'}}(customer \bowtie_{cname} deposit))$$

When a join predicate/condition is found as part of a top-level conjunction, it may be removed from a selection operation to be combined with a Cartesian Product operation to form a join. In this case the join predicate (i.e., predicates involving attributes from two tables) is $c.cname = d.cname$.

The **Join** $\bowtie$ pairs each tuple in d with its unique match in relation c. The size (in tuples) of the output of an equi-join of the form PK = FK is the size of the FK table.

$$\text{reads} = nb_c + nb_c \cdot nb_d \qquad = 100 + 100 \cdot 200 = 20,100$$
$$\text{writes} = nb_{\bowtie} = \lceil \frac{n_d}{bf_{\bowtie}} \rceil \qquad = \lceil \frac{2000}{5} \rceil = 400$$

Now the **Select** σ has a single predicate and filter.

$$\text{reads} = nb_{\bowtie} \qquad = 400$$
$$\text{writes} = \lceil \frac{n_d}{V(bname, d) \cdot bf_{\bowtie}} \rceil \qquad = \lceil \frac{2000}{10 \cdot 5} \rceil = 40$$

total = 20,100 + 400 + 400 + 40 = 20,940

3. **Move Select Operations σ Down the Tree**

$$\pi_{c.cname,\, c.ccity}(customer \bowtie_{cname} (\sigma_{bname=`Alps'}(deposit)))$$

The **Select** σ has a single predicate (filter) and now pulls tuples directly from relation d

$$\begin{aligned} \text{reads} &= nb_d & &= 200 \\ \text{writes} &= \lceil \frac{n_d}{V(bname, d) \cdot bf_d} \rceil & &= \lceil \frac{2000}{10 \cdot 10} \rceil = 20 \end{aligned}$$

For the **Join**, the c table and d' table (what remains of d after selection) are now joined. The c relation is still the PK table, so the size of the output is the size relation d'.

$$\begin{aligned} \text{reads} &= nb_{d'} + nb_{d'} \cdot nb_c & &= 20 + 20 \cdot 100 = 2,020 \\ \text{writes} &= nb_{\bowtie} = \lceil \frac{n_{d'}}{bf_{\bowtie}} \rceil & &= \lceil \frac{200}{5} \rceil = 40 \end{aligned}$$

total = 200 + 20 + 2020 + 40 = 2280

4. **Apply Indexes - B+Trees Formulas**

$$h(n) = \lceil \log_p \frac{n}{p-1} \rceil = \lceil \log_{10} \frac{n}{9} \rceil$$

$$\text{select : unique} = h(n) + 2$$
$$\text{select : nonunique} = h(n) + 2 + writes$$
$$\text{join : unique} = nb_1 + n_1(h(n_2) + 2)$$
$$\text{join : nonunique} = \text{complicated}$$

Use a non-unique index for *bname* in the *deposit* relation to reduce reads for the **Select** operation.

$$\text{reads} = h(V(bname, d)) + 2 + writes \qquad 1 + 2 + 20 = 23$$
$$\text{writes} = \lceil \frac{n_d}{V(bname, d) \cdot bf_d} \rceil \qquad = \lceil \frac{2000}{10 \cdot 10} \rceil = 20$$

Use a unique index for *cname* in the *customer* relation to reads for the **Join** operation.

$$\text{reads} = nb_{d'} + n_{d'}(h(n_c) + 2) \qquad = 20 + 200(3 + 2) = 1020$$
$$\text{writes} = nb_{\bowtie} = \lceil \frac{n_{d'}}{bf_{\bowtie}} \rceil \qquad = \lceil \frac{200}{5} \rceil = 40$$

total = 23 + 20 + 1020 + 40 = 1103

7 Summary

Table 1: Results for the Four Optimization Steps

Opt Step	**Description**	**Tuples**	**Blocks**
1	Unoptimized	2,000,200	820,140
2	Products to Joins	2200	20,940
3	Selects Down	400	2280
4	Apply Indexes	NA	1103