CS 457: Database Management Systems

Query Optimization (part 1)
Wrap up of Cost Estimation
Know how to compute the cost of a plan

Next: Find a good plan automatically?

This is the role of the query optimizer
Query Optimization Overview

SQL query

- Parse & Rewrite Query
- Select Logical Plan
- Select Physical Plan
- Query Execution

Query optimization

Disk

Logical plan

Physical plan
What We Already Know…

```
Supplier(sno, sname, scity, sstate)
Part(pno, pname, psize, pcolor)
Supply(sno, pno, price)
```

For each SQL query….

```
SELECT S.sname
FROM Supplier S, Supply U
WHERE S.scity='Seattle' AND S.sstate='WA'
AND S.sno = U.sno
AND U.pno = 2
```

There exist many logical query plans…
Example Query: Logical Plan 1

\[ \pi_{\text{sname}} \]

\[ \sigma_{\text{sscity}='Seattle' \land \text{state}='WA' \land \text{pno}=2} \]

\[ \text{sno = sno} \]

Supplier

Supply
Example Query: Logical Plan 2

\[ \pi_{\text{sname}} \]
\[ \sigma_{\text{sscity}='Seattle' \land \text{sstate}='WA'} \]
\[ \sigma_{\text{pno}=2} \]

\[ \text{Supplier} \]
\[ \text{Supply} \]
What We Also Know

• For each logical plan…

• There exist many physical plans
Example Query: Physical Plan 1

\[(\text{On the fly})\]

\[\pi_{\text{sname}}\]

\[(\text{On the fly})\]

\[\sigma_{\text{scity}=\text{Seattle} \land \text{sstate}=\text{WA} \land \text{pno}=2}\]

\[(\text{Nested loop})\]

\[\text{sno} = \text{sno}\]

\[\text{Supplier (File scan)}\]

\[\text{Supply (File scan)}\]
Example Query: Physical Plan 2

\(\sigma_{\text{scity} = \text{Seattle} \land sstate = \text{WA} \land pno = 2}\) ⋀ \(\pi_{\text{sname}}\)

(On the fly)

(On the fly)

(Index nested loop)

sno = sno

Supplier
(File scan)

Supply
(Index scan)
Query Optimizer Overview

• **Input**: A logical query plan
• **Output**: A good physical query plan
• **Basic query optimization algorithm**
  – Enumerate alternative plans (logical and physical)
  – Compute estimated cost of each plan
    • Compute number of I/Os
    • Optionally take into account other resources
  – Choose plan with lowest cost
  – This is called cost-based optimization
Lessons

• No magic “best” plan: depends on the data

• In order to make the right choice
  – Need to have statistics over the data
  – The B’s, the T’s, the V’s
  – Commonly: histograms over base data
Outline

• Search space

• Algorithm for enumerating query plans
Relational Algebra Equivalences

• Selections
  – Commutative: $\sigma_{c_1}(\sigma_{c_2}(R))$ same as $\sigma_{c_2}(\sigma_{c_1}(R))$
  – Cascading: $\sigma_{c_1} \land c_2(R)$ same as $\sigma_{c_2}(\sigma_{c_1}(R))$

• Projections
  – Cascading

• Joins
  – Commutative: $R \bowtie S$ same as $S \bowtie R$
  – Associative: $R \bowtie (S \bowtie T)$ same as $(R \bowtie S) \bowtie T$
Left-Deep Plans, Bushy Plans, and Linear Plans

Linear plan: One input to each join is a relation from disk
Can be either left or right input
Commutativity, Associativity, Distributivity

\[ R \cup S = S \cup R, \quad R \cup (S \cup T) = (R \cup S) \cup T \]

\[ R \otimes S = S \otimes R, \quad R \otimes (S \otimes T) = (R \otimes S) \otimes T \]

\[ R \otimes (S \cup T) = (R \otimes S) \cup (R \otimes T) \]
Laws Involving Selection

\[ \sigma_{C \text{AND} C'}(R) = \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R) \]
\[ \sigma_{C \text{OR} C'}(R) = \sigma_C(R) \cup \sigma_{C'}(R) \]
\[ \sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S \]

Assuming C on attributes of R

\[ \sigma_C(R - S) = \sigma_C(R) - S \]
\[ \sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S) \]
\[ \sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S \]
Example: Simple Algebraic Laws

- Example: \( R(A, B, C, D), S(E, F, G) \)

\[
\sigma_{F=3} (R \bowtie_{D=E} S) = ? \\
\sigma_{A=5 \text{ AND } G=9} (R \bowtie_{D=E} S) = ?
\]
Example:
Simple Algebraic Laws

• Example: \( R(A, B, C, D), S(E, F, G) \)

\[
\sigma_{F=3} (R \bowtie_{D=E} S) = R \bowtie_{D=E} \sigma_{F=3} (S)
\]

\[
\sigma_{A=5 \text{ AND } G=9} (R \bowtie_{D=E} S) = \sigma_{A=5} (R) \bowtie_{D=E} \sigma_{G=9} (S)
\]
Laws Involving Projections

\[
\Pi_M(R \bowtie S) = \Pi_M(\Pi_P(R) \bowtie \Pi_Q(S))
\]

\[
\Pi_M(\Pi_N(R)) = \Pi_M(R)
/* note that M \subseteq N */
\]

• Example \(R(A,B,C,D), S(E, F, G)\)

\[
\Pi_{A,B,G}(R \bowtie_{D=E} S) = \Pi_? (\Pi_? (R) \bowtie_{D=E} \Pi_? (S))
\]
Laws Involving Projections

\[ \Pi_M(R \bowtie S) = \Pi_M(\Pi_P(R) \bowtie \Pi_Q(S)) \]

/* what are P and Q? see below */

\[ \Pi_M(\Pi_N(R)) = \Pi_M(R) \]

/* note that M \subseteq N */

- Example \( R(A,B,C,D), S(E, F, G) \)
  \[ \Pi_{A,B,G}(R \bowtie_{D=E} S) = \Pi_{A,B,G}(\Pi_{A,B,D}(R) \bowtie_{D=E} \Pi_{E,G}(S)) \]
Laws involving grouping and aggregation

\[ \gamma_{A, \text{agg}(D)} (R(A, B) \bowtie_{B=C} S(C, D)) = \gamma_{A, \text{agg}(D)} (R(A, B) \bowtie_{B=C} (\gamma_{C, \text{agg}(D)} S(C, D)))) \]
Laws Involving Constraints

Product\((pid, pname, price, cid)\)
Company\((cid, cname, city, state)\)

\[ \Pi_{pid, price} (Product \bowtie_{cid=cid} Company) = \Pi_{pid, price} (Product) \]
Search Space Challenges

- Search space is huge!
  - Many possible equivalent trees (we just discussed)
  - Many implementations for each operator (previously)
  - Many access paths for each relation (previously)
    - File scan, index

- Cannot consider ALL plans
  - Heuristics: only partial plans with “low” cost
Outline

• Search space

• Algorithm for enumerating query plans