CS 347: Distributed Databases and Transaction Processing  
**Notes04: Query Optimization**  
Hector Garcia-Molina

---

**Query optimization**

- Cost estimation
- Strategies for exploring plans

---

**Cost estimation**

- As in centralized system: estimate result sizes

---

But: # IOs may not be best metric

- Transmission time may dominate
- e.g., Transmission time may dominate
- e.g., Transmission time may dominate

---

Another reason why plain IOs not enough: Parallelism

**Plan A**

- site 1 50 IOs
- site 2 70 IOs
- site 3 50 IOs

**Plan B**

- site 1 50 IOs
- site 2 70 IOs
- site 3 50 IOs

---

- Cost metrics
  - IOs, Bytes transmitted, $, ...
  - Can add together
- Response time metric
  - Cannot add
  - Need scheduling and dependency info
  - Skew important
Take into account:
(in parallel/distributed system)
• Start up costs (for parallel operation)
• Data distribution costs/time
• Contention
  – memory, disk, network,…
• Assembling result

Example: Response time

<table>
<thead>
<tr>
<th>Site</th>
<th>Startup</th>
<th>Distribution</th>
<th>Searching</th>
<th>Final proc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Site 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Site 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Site 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Searching strategies

1. Exhaustive (with pruning)
2. Hill climbing (greedy)
3. Query separation

(1) Exhaustive
- consider “all” query plans with a set of techniques
- prune some plans - heuristics

Example: join

\[
\begin{align*}
R \times A \rightarrow S \rightarrow T \quad |R|>|S|>|T|
\end{align*}
\]

\[
\begin{align*}
R \bowtie S & \rightarrow R \times T \\
S & \rightarrow T \rightarrow R
\end{align*}
\]

1. Prune because cross-product not necessary
2. Prune because larger relation first

In generating plans, keep goal in mind:

- e.g.: Goal is parallelism in system with fast net, consider partitioning relation(s) first
- e.g.: Goal is reduction of net traffic, consider semi-joins
(2) Hill climbing

Better plans

Worse plans

Initial plan

Example

<table>
<thead>
<tr>
<th>Rel</th>
<th>Site</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>S</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>V</td>
<td>4</td>
<td>40</td>
</tr>
</tbody>
</table>

Goal: minimize data transmission

Initial plan: send relations to one site

What site do we send all relations to?
To site 1: cost = 20 + 30 + 40 = 90
To site 2: cost = 10 + 30 + 40 = 80
To site 3: cost = 10 + 20 + 40 = 70
To site 4: cost = 10 + 20 + 30 = 60

Local search

- Consider sending each relation to neighbor:
  e.g.:
Assume: Size $R \bowtie S = 20$
$S \bowtie T = 5$
$T \bowtie V = 1$

Option (a)

Option (b)

Option (c)

Option (d)

P1: P1a: S (2 $\rightarrow$ 3)
    $\alpha = S \bowtie T$
P1b: R (1 $\rightarrow$ 4)
    $\alpha (3 \rightarrow 4)$
    compute answer at site 4

Repeat local search
- Treat $\alpha = S \bowtie T$ as relation
Hill climbing may miss best plan!

Example: best plan could be:

PB: T (3 → 4)

β = T ≻ V

β' = β ≻ S

β'' = β' ≻ R

[optional] β'' (1 → 4)

Compute answer

Costs could be low because β is very selective

(3) Query separation

- separate query into 2 or more steps
- optimize each step independently

Example: simple queries

e.g.:

\[ \sigma_{c1} \]

\[ \sigma_{c2} \]

\[ \sigma_{c3} \]

1. Compute \( R' = \pi_{A}[\sigma_{c2} R] \)

2. Compute \( S' = \pi_{A}[\sigma_{c3} S] \)

3. Compute \( \text{Ans} = \sigma_{c1}\{[J \bowtie \sigma_{c2} R] \bowtie [J \bowtie \sigma_{c3} S]\} \)

In other words:

(a) Compute \( A \) values in answer

(b) Get tuples from sites with matching \( A \) values and compute answer

(steps 1,2)

(step 3)
Simple query
- Relations have a single attribute
- Output has a single attribute
  e.g., \( J \leftarrow R' \bowtie S' \)

Idea
- Decompose query into
  - Local processing
  - Simple query (or queries)
  - Final processing
- Optimize simple query

Philosophy
- Hard part is distributed join
- Do this part with only keys;
  get rest of data later
- Simpler to optimize simple queries

Summary: Query Optimization
- Cost estimation
- Strategies
  - Exhaustive
  - Hill climbing
  - Separation

Words of wisdom
“Optimization is like chess playing”
i.e., May have to make sacrifices
  (move data, partition relations, build indexes)
  for later gains!