Advanced Database Technology

March 20, 2003

QUERY

COMPILATION II

Lecture based on [GUW, 16.4–16.7]

Slides based on
Notes 06–07: Query execution Part I–II
for Stanford CS 245, fall 2002
by Hector Garcia–Molina
This lecture (10–12 AM)

• We assume that we have an algebraic expression (tree), and consider:
  – Simple estimates of the size of relations given by subexpressions.
  – Statistics that improve estimates.
  – Choosing an order for operations, using various optimization techniques.
  – Completing the physical query plan.
Estimating sizes of relations

The **sizes** of intermediate results are important for the choices made when planning query execution.

- Time for operations grow (at least) linearly with size of (largest) argument.
- The total size can even be used as a crude estimate on the running time.
Statistics for computing estimates

The book suggests several statistics on relations that may be used to \textit{(heuristically)} estimate the size of intermediate results.

- \(T(R)\): # tuples in R
- \(S(R)\): # bytes in each R tuple
- \(B(R)\): # blocks to hold all R tuples
- \(V(R, A)\): # distinct values in R for attribute A
Size estimates for $W = R_1 \times R_2$

$T(W) = T(R_1) \times T(R_2)$

$S(W) = S(R_1) + S(R_2)$

**Question:** How good are these estimates?
Size estimate for $W = \sigma_{A=a} (R)$

$S(W) = S(R)$

$T(W) = ?$
Some possible assumptions

• Values in select expression $A = a$ (or at least one of them) are uniformly distributed over the possible $V(R,A)$ values.

• As above, but with uniform distribution over domain with $\text{DOM}(R,A)$ values.

• Zipfian distribution of values.
Selection cardinality

\[ SC(R,A) = \text{expected \# records that satisfy equality condition on } R.A \]

\[ SC(R,A) = \begin{cases} \frac{T(R)}{V(R,A)} & \text{under first assumption} \\ \frac{T(R)}{\text{DOM}(R,A)} & \text{under } 2^{nd} \text{ assumption} \end{cases} \]
Size estimate for $W = \sigma_{A \geq a}(R)$

$T(W) = ?$

- Suggestion # 1: $T(W) = T(R)/2$.
- Suggestion # 2: $T(W) = T(R)/3$.
- Suggestion # 3 (not in book): Be consistent with equality estimate.
Example:
Consistency with 2\textsuperscript{nd} equality estimate.

\[ f = \frac{20-14}{20} \quad \text{(fraction of range)} \]

\[ T(W) = f \times T(R) \]
Problem session

Consider the natural join operation on two relations $R_1$ and $R_2$ with join attribute $A$.

- If values for $A$ are uniformly distributed on $\text{DOM}(R_1,A) = \text{DOM}(R_2,A)$ values, what is the expected size of $R_1 \bowtie R_2$?

- What can you say if values for $A$ are instead uniform on respectively $\text{V}(R_1,A)$ and $\text{V}(R_2,A)$ values?

- What if $A$ is primary key for $R_1$ and/or $R_2$?
Crude estimate

Values uniformly distributed over domain

<table>
<thead>
<tr>
<th>R1</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>A</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

This tuple matches $T(R2)/\text{DOM}(R2, A)$ so

$$T(W) = \frac{T(R2) \cdot T(R1)}{\text{DOM}(R2, A)} = \frac{T(R2) \cdot T(R1)}{\text{DOM}(R1, A)}$$

Assume the same
General crude estimate

Let $W = R_1 \mapsto R_2 \mapsto R_3 \mapsto \ldots \mapsto R_k$

$$T(W) = \frac{T(R_1) T(R_2) \ldots T(R_k)}{\text{DOM}(R_1, A)^{k-1}}$$

Symmetric wrt. the relations. A rare property...
"Better" size estimate for $W = R_1 \implies R_2$

**Assumption:**Containment of value sets

\[
V(R_1,A) \leq V(R_2,A) \implies \text{Every } A \text{ value in } R_1 \text{ is in } R_2 \\
V(R_2,A) \leq V(R_1,A) \implies \text{Every } A \text{ value in } R_2 \text{ is in } R_1
\]
Computing $T(W)$ when $V(R1,A) \leq V(R2,A)$

<table>
<thead>
<tr>
<th>R1</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Take 1 tuple and Match

1 tuple matches with $T(R2)$ tuples...

$\frac{V(R2,A)}{V(R2,A)}$

so $T(W) = \frac{T(R2) \times T(R1)}{V(R2, A)}$
General estimate

Let $W = R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow \ldots \rightarrow R_k$

$T(W) = T(R_1) \ldots T(R_k) \min \{ V(R_1,A), \ldots, V(R_k,A) \}$

$V(R_1,A) \ V(R_2,A) \ V(R_k,A)$

**Underlying assumption:**
Preservation of value sets
Multiple join attributes

The previous estimates are easily extended to several join attributes $A_1,...,A_j$:

• **New assumption**: Values are independent.

• Under assumption 1, the joint values in attributes are uniformly distributed on $V(R,A_1) \ V(R,A_2) \ ... \ V(R,A_j)$ values.

• Under assumption 2, they are uniform on $DOM(R,A_1) \ DOM(R,A_2) \ ... \ DOM(R,A_j)$ values.
Other estimates use similar ideas

$\Pi_{AB}(R) \ldots$ Sec. 16.4.2

$\sigma_{A=a \land B=b}(R) \ldots$ Sec. 16.4.3

Union, intersection, duplicate elimination, difference, \ldots Sec. 16.4.7
Improved estimates through histograms

• **Idea**: Maintain more info than just $V(R,A)$.
• **Histogram**: Number of values in each of a number of intervals.
Problem session

Consider how histograms could be used when estimating the size of:
- A selection.
- A natural join.

You may assume that both relations have histograms using the same intervals.
Maintaining statistics

• There is a cost to maintaining statistics.
• Book recommends recomputing "once in a while" (don’t change rapidly).
• Recomputation may be operator–controlled.

**Question**: How does one compute the statistics (V(R,A), histogram) of a relation?
Choosing a physical plan

Option 1: "Branch and bound".

Query

Generate

Prune

Estimate Cost
(using size est.)

Pick Min

Plans

Estimate Cost
(using size est.)

Pick Min

Generate

Prune

Estimate Cost
(using size est.)

Pick Min

Generate

Prune

Estimate Cost
(using size est.)

Pick Min

Generate

Prune

Estimate Cost
(using size est.)

Pick Min

Generate

Prune

Estimate Cost
(using size est.)

Pick Min

Generate

Prune

Estimate Cost
(using size est.)

Pick Min

Generating Plans

Pruning Plans

Estimated Costs

Pick Min

Choose best plan
Choosing a physical plan

Option 2: "Dynamic programming".

• Find best plans for subexpressions in bottom–up order.
• Might find several best plans:
  – The best plan that produces a sorted relation wrt. a later join or grouping attribute.
  – The best plan in general.
Choosing a physical plan

Options 3,4,...:

Other (heuristic) techniques from optimization.

• Greedy plan selection.
• Hill climbing.
• ...


Order for grouped operations

• Recall that we grouped commutative and associative operators, e.g. 
  $R_1 \circ \rightarrow R_2 \circ \rightarrow R_3 \circ \rightarrow \ldots \circ \rightarrow R_k$.

• For such expressions we must choose an evaluation order (a parenthesized expression), e.g. 
  $(R_1 \circ \rightarrow R_4) \circ \rightarrow (R_3 \circ \rightarrow \ldots) \ldots (\ldots \circ \rightarrow R_k)$. 
Order for grouped operations

• Book recommends considering just left balanced expressions
  \((\ldots((R4 \mapsto R2) \mapsto R7) \mapsto \ldots \ldots) \mapsto Rk\).

• This gives \(k!\) possible expressions.

• Considering all possible expressions gives around \(2^k k!\) possibilities – not so many more.
Choosing final algorithms

• Usually best to use existing indexes.
• Sometimes building indexes or sorting on the fly is advantageous.
• Sorting based algorithms may beat hashing based algorithms if one of the relations is already sorted.
• Just do the calculation and see!
Pipelining and materialization

• Some algorithms (e.g. $\sigma$ implemented as a scan) require little internal memory.
• **Idea:** Don’t write result to disk, but feed it to the next algorithm immediately.
• Such **pipelining** may make many algorithms run "at the same time".
• Sometimes even possible with algorithms using more memory, such as sorting.
Summary

• **Size estimation** (using statistics) is an important part of query optimization.

• Given size estimates and a relational algebra expression, **query optimization** essentially consists of computing the (estimated) cost of all possible query plans.

• Other issues are **pipelining** and memory usage during execution.