CS4432: Database Systems II

Lecture #15
Query Processing

Professor Elke A. Rundensteiner
• Estimating cost of query plan

(1) Estimating size of results
(2) Estimating # of IOs
Estimating result statistics

- Keep statistics for relation R
  - $T(R)$: # tuples in R
  - $S(R)$: # of bytes in each R tuple
  - $B(R)$: # of blocks to hold all R tuples
  - $V(R, A)$: # distinct values in R for attribute A
Note: for complex expressions, need intermediate T,S,V results.

E.g. \( W = [\sigma_{A=a} (R1)] \bowtie R2 \)

Treat as relation U

\( T(U) = T(R1)/V(R1,A) \quad S(U) = S(R1) \)

Also need V (U, *) !!
To estimate Vs

E.g., $U = \sigma_{A=a}(R1)$

Say R1 has attribs A,B,C,D

$V(U, A) =$
$V(U, B) =$
$V(U, C) =$
$V(U, D) =$
Example

R1

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>10</td>
</tr>
</tbody>
</table>

V(R1,A) = 3  
V(R1,B) = 1  
V(R1,C) = 5  
V(R1,D) = 3

U = \sigma_{A=a} (R1)

V(U,A) = 1  
V(U,B) = 1  
V(U,C) = \frac{T(R1)}{V(R1,A)}

V(D,U) ... somewhere in between
Possible Guess  \( U = \sigma_{A=a}(R) \)

\[
\begin{align*}
V(U, A) &= 1 \\
V(U, B) &= V(R, B)
\end{align*}
\]
For Joins: \[ U = R1(A,B) \bowtie R2(A,C) \]

\[
V(U,A) = \min \{ V(R1, A), V(R2, A) \} \\
V(U,B) = V(R1, B) \\
V(U,C) = V(R2, C)
\]

[called “preservation of value sets” in section 7.4.4]
Example:

\[ Z = R_1(A,B) \Join R_2(B,C) \bowtie R_3(C,D) \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>T(R1) = 1000</td>
<td>V(R1,A)=50</td>
</tr>
<tr>
<td>R2</td>
<td>T(R2) = 2000</td>
<td>V(R2,B)=200</td>
</tr>
<tr>
<td>R3</td>
<td>T(R3) = 3000</td>
<td>V(R3,C)=90</td>
</tr>
</tbody>
</table>
Partial Result: \( U = R \bowtie S \)

\[
T(U) = \frac{1000 \times 2000}{200} \quad V(U,A) = 50
\]

\[
V(U,B) = 100 \quad V(U,C) = 300
\]
\[ Z = U \Join R3 \]

\[
T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300} \quad V(Z,A) = 50 \\
V(Z,B) = 100 \\
V(Z,C) = 90 \\
V(Z,D) = 500
\]
Summary on Estimation of “Sizes”

- Estimating size of results is an “art”

- Don’t forget: Statistics must be kept up to date... (cost?)
Query Optimization

--> Generating and comparing plans

Generate
Pruning
Estimate Cost
Select

Pick Min

Plans
Cost
To generate plans consider:

- Transforming relational algebra expression (e.g. order of joins)
- Use of existing indexes
- Building indexes or sorting on the fly
• Implementation details:
  e.g. - Join algorithm
  - Memory management
Estimating IOs:

• Count # of disk blocks that must be read (or written) to execute query plan
To estimate costs, we may have additional parameters:

\[ B(R) = \# \text{ of blocks containing } R \text{ tuples} \]

\[ f(R) = \max \# \text{ of tuples of } R \text{ per block} \]

\[ M = \# \text{ memory blocks available} \]

\[ HT(i) = \# \text{ levels in index } i \]

\[ LB(i) = \# \text{ of leaf blocks in index } i \]
Clustering index

Index that allows tuples to be read in an order that corresponds to physical order

A

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
</tr>
</tbody>
</table>
Notions of clustering

• Clustered file organization
  
  | R1 | R2 | S1 | S2 | R3 | R4 | S3 | S4 | ..... |

• Clustered relation
  
  | R1 | R2 | R3 | R4 | R5 | R5 | R7 | R8 | ..... |

• Clustering index
Example \( R_1 \bowtie R_2 \) over common attribute \( C \)

\[
\begin{align*}
T(R_1) & = 10,000 \\
T(R_2) & = 5,000 \\
S(R_1) = S(R_2) & = 1/10 \text{ block} \\
\text{Memory available} & = 101 \text{ blocks}
\end{align*}
\]

\( \rightarrow \) Metric: \( \# \) of IOs

\( \text{(ignoring writing of result)} \)
Caution!

This may not be the best way to compare:
- ignoring CPU costs
- ignoring timing
- ignoring double buffering requirements
Options

- Transformations: R1 $\bowtie$ R2, R2 $\bowtie$ R1
- Join algorithms:
  - Iteration (nested loops)
  - Merge join
  - Join with index
  - Hash join
• **Iteration join** (conceptually)
  
  for each $r \in R_1$ do
  
  for each $s \in R_2$ do
  
  if $r.C = s.C$ then output $r,s$ pair
Example 1(a) Iteration Join $R_1 \Join R_2$

- Relations not contiguous
- Recall
  \[
  \begin{align*}
  T(R_1) &= 10,000 \\
  T(R_2) &= 5,000 \\
  S(R_1) &= S(R_2) = \frac{1}{10} \text{ block} \\
  \text{MEM} &= 101 \text{ blocks}
  \end{align*}
  \]

Cost: for each $R_1$ tuple:

\[
[\text{Read tuple} + \text{Read } R_2]
\]

Total = 10,000 \[1+5000\] = 50,010,000 IOs
• Can we do better?

Use our memory
(1) Read 100 blocks of R1
(2) Read all of R2 (using 1 block) + join
(3) Repeat until done
**Cost:** for each R1 chunk:

Read chunk: 1000 IOs

Read R2: \[
\begin{array}{c}
5000 \\
\hline
6000
\end{array}
\]

Total = \[
\begin{array}{c}
10,000 \\
\hline
1,000
\end{array}
\] \times 6000 = 60,000 IOs
• Can we do better?

Reverse join order:  R2 ▷◁ R1

Total = \(\frac{5000}{1000} \times (1000 + 10,000) = 5 \times 11,000 = 55,000\) IOs
Example 1(b) Iteration Join $R2 \bowtie R1$

- Relations contiguous

Cost
For each $R2$ chunk:
- Read chunk: 100 IOs
- Read $R1$: 1000 IOs

Total = 5 chunks x 1,100 = 5,500 IOs
Some Factors that affect performance

(1) Tuples of relation stored physically together?

(2) Buffer-aware data movement

(3) Relations sorted by join attribute?

(3) Indexes exist?