Query Processing
19 March 2012
Prof. Chris Clifton

Query Processing

• Q → Query Plan

Focus: Relational System

• Others?
Example

Select B, D
From R, S
Where R.A = “c” \( \land \) S.E = 2 \( \land \) R.C=S.C

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>10</td>
<td></td>
<td>10</td>
<td>x</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>20</td>
<td></td>
<td>20</td>
<td>y</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>10</td>
<td></td>
<td>30</td>
<td>z</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>35</td>
<td></td>
<td>40</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>45</td>
<td></td>
<td>50</td>
<td>y</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Answer  

<table>
<thead>
<tr>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>x</td>
</tr>
</tbody>
</table>
How do we execute query?

One idea

- Do Cartesian product
- Select tuples
- Do projection

<table>
<thead>
<tr>
<th>RXS</th>
<th>R.A</th>
<th>R.B</th>
<th>R.C</th>
<th>S.C</th>
<th>S.D</th>
<th>S.E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td></td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>10</td>
<td>20</td>
<td></td>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>x</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bingo!
Got one...
Relational Algebra - can be used to describe plans...

Ex: Plan I

\[ \Pi_{B,D} \]

\[ \sigma_{R.A=\\text{“c”} \land S.E=2 \land R.C=S.C} \]

\( \times \)

\( R \rightarrow S \)

OR: \[ \Pi_{B,D} \left[ \sigma_{R.A=\\text{“c”} \land S.E=2 \land R.C=S.C} \right] (RXS) \]

Another idea:

Plan II

\[ \Pi_{B,D} \]

\[ \sigma_{R.A=\\text{“c”}} \]

\[ \sigma_{S.E=2} \]

\( R \) \( \bowtie \) \( S \)

natural join

CS54100
Overview of Query Evaluation

- **Plan**: Tree of R.A. ops, with choice of alg for each op.
  - Each operator typically implemented using a `pull' interface: when an operator is `pulled' for the next output tuples, it `pulls' on its inputs and computes them.

- Two main issues in query optimization:
  - For a given query, what plans are considered?
    - Algorithm to search plan space for cheapest (estimated) plan.
    - How is the cost of a plan estimated?
  - Ideally: Want to find best plan. Practically: Avoid worst plans!

- We will study the System R approach.
Plan III

Use R.A and S.C Indexes

1. Use R.A index to select R tuples with R.A = “c”
2. For each R.C value found, use S.C index to find matching tuples
3. Eliminate S tuples S.E ≠ 2
4. Join matching R,S tuples, project B,D attributes and place in result

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>x</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>y</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>z</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>y</td>
<td>3</td>
</tr>
</tbody>
</table>
Some Common Techniques

- Algorithms for evaluating relational operators use some simple ideas extensively:
  - Indexing: Can use WHERE conditions to retrieve small set of tuples (selections, joins)
  - Iteration: Sometimes, faster to scan all tuples even if there is an index. (And sometimes, we can scan the data entries in an index instead of the table itself.)
  - Partitioning: By using sorting or hashing, we can partition the input tuples and replace an expensive operation by similar operations on smaller inputs.

* Watch for these techniques as we discuss query evaluation!

Access Paths

- An access path is a method of retrieving tuples:
  - File scan, or index that matches a selection (in the query)
- A tree index matches (a conjunction of) terms that involve only attributes in a prefix of the search key.
  - E.g., Tree index on \(<a, b, c>\) matches the selection \(a=5\) AND \(b=3\), and \(a=5\) AND \(b>6\), but not \(b=3\).
- A hash index matches (a conjunction of) terms that has a term attribute = value for every attribute in the search key of the index.
  - E.g., Hash index on \(<a, b, c>\) matches \(a=5\) AND \(b=3\) AND \(c=5\); but it does not match \(b=3\), or \(a=5\) AND \(b=3\), or \(a>5\) AND \(b=3\) AND \(c=5\).
**A Note on Complex Selections**

Selection conditions are first converted to **conjunctive normal form (CNF):**

\[(\text{day}<8/9/94 \text{ OR } \text{bid}=5 \text{ OR } \text{sid}=3) \text{ AND } (\text{rname}='Paul' \text{ OR } \text{bid}=5 \text{ OR } \text{sid}=3)\]

We only discuss case with no ORs; see text if you are curious about the general case.

**Using an Index for Selections**

- Cost depends on #qualifying tuples, and clustering.
  - Cost of finding qualifying data entries (typically small) plus cost of retrieving records (could be large w/o clustering).
  - In example, assuming uniform distribution of names, about 10% of tuples qualify (100 pages, 10000 tuples). With a clustered index, cost is little more than 100 I/Os; if unclustered, upto 10000 I/Os!
One Approach to Selections

- Find the most selective access path, retrieve tuples using it, and apply any remaining terms that don’t match the index:
  - Most selective access path: An index or file scan that we estimate will require the fewest page I/Os.
  - Terms that match this index reduce the number of tuples retrieved; other terms are used to discard some retrieved tuples, but do not affect number of tuples/pages fetched.
  - Consider \( \text{day}<8/9/94 \text{ AND bid}=5 \text{ AND sid}=3 \). A B+ tree index on \( \text{day} \) can be used; then, \( \text{bid}=5 \) and \( \text{sid}=3 \) must be checked for each retrieved tuple. Similarly, a hash index on \( <\text{bid},\text{sid}> \) could be used; \( \text{day}<8/9/94 \) must then be checked.

Projection

- The expensive part is removing duplicates.
  - SQL systems don’t remove duplicates unless the keyword DISTINCT is specified in a query.
  - Sorting Approach: Sort on \( <\text{sid},\text{bid}> \) and remove duplicates. (Can optimize this by dropping unwanted information while sorting.)
  - Hashing Approach: Hash on \( <\text{sid},\text{bid}> \) to create partitions. Load partitions into memory one at a time, build in-memory hash structure, and eliminate duplicates.
  - If there is an index with both \( \text{R.sid} \) and \( \text{R.bid} \) in the search key, may be cheaper to sort data entries!
Example: SQL query

```
SELECT title
FROM StarsIn
WHERE starName IN (  
    SELECT name
    FROM MovieStar
    WHERE birthdate LIKE ‘%1960’
);
```

(Find the movies with stars born in 1960)
Example: Parse Tree

```
SELECT <SelList> FROM <FromList> WHERE <Condition>

<Attribute> <RelName> <Tuple> IN <Query>

name MovieStar birthDate '1960'
```

Example: Generating Relational Algebra

```
\Pi_{\text{title}}
\sigma_{\text{StarsIn}} <\text{condition}> IN \Pi_{\text{name}}
\sigma_{\text{birthdate LIKE '1960'}}

\text{starName MovieStar}
```

Fig. 7.15: An expression using a two-argument \( \sigma \), midway between a parse tree and relational algebra.
Example: Logical Query Plan

\[ \Pi_{\text{title}} \]
\[ \sigma_{\text{starName}=\text{name}} \]
\[ \times \]
\[ \text{StarsIn} \]
\[ \Pi_{\text{name}} \]
\[ \sigma_{\text{birthdate LIKE '91960'}} \]
\[ \text{MovieStar} \]

Fig. 7.18: Applying the rule for IN conditions

Example: Improved Logical Query Plan

\[ \Pi_{\text{title}} \]
\[ \triangledown \]
\[ \text{starName}=\text{name} \]
\[ \text{StarsIn} \]
\[ \Pi_{\text{name}} \]
\[ \sigma_{\text{birthdate LIKE '91960'}} \]
\[ \text{MovieStar} \]

Question: Push project to StarsIn?

Fig. 7.20: An improvement on fig. 7.18.
Example: Estimate Result Sizes

\[ \Pi \sigma \]
\[ \text{MovieStar} \]

Need expected size

Example: One Physical Plan

Hash join → Parameters: join order, memory size, project attributes,...

SEQ scan

index scan

\[ \text{StarsIn} \]

MovieStar

→ Parameters: Select Condition,...
Example: Estimate costs

L.Q.P

P1  P2  ....  Pn

|   |   |
C1  C2  ....  Cn

Pick best!

Join: Index Nested Loops

foreach tuple r in R do
  foreach tuple s in S where r_i == s_j do
    add <r, s> to result

- If there is an index on the join column of one relation (say S), can make it the inner and exploit the index.
  - Cost: $M + (M \times p_R) \times \text{cost of finding matching S tuples}$
  - $M=$#pages of R, $p_R=$# R tuples per page
- For each R tuple, cost of probing S index is about 1.2 for hash index, 2-4 for B+ tree. Cost of then finding S tuples (assuming Alt. (2) or (3) for data entries) depends on clustering.
  - Clustered index: 1 I/O (typical), unclustered: upto 1 I/O per matching S tuple.
Examples of Index Nested Loops

- Hash-index (Alt. 2) on \textit{sid} of Sailors (as inner):
  - Scan Reserves: 1000 page I/Os, 100*1000 tuples.
  - For each Reserves tuple: 1.2 I/Os to get data entry in index, plus 1 I/O to get (the exactly one) matching Sailors tuple. Total: 220,000 I/Os.

- Hash-index (Alt. 2) on \textit{sid} of Reserves (as inner):
  - Scan Sailors: 500 page I/Os, 80*500 tuples.
  - For each Sailors tuple: 1.2 I/Os to find index page with data entries, plus cost of retrieving matching Reserves tuples. Assuming uniform distribution, 2.5 reservations per sailor (100,000 / 40,000). Cost of retrieving them is 1 or 2.5 I/Os depending on whether the index is clustered.

Join: Sort-Merge ($R \bowtie_i S$)

- Sort $R$ and $S$ on the join column, then scan them to do a \textasciitilde``merge'' \textasciitilde (on join col.), and output result tuples.
  - Advance scan of $R$ until current $R$-tuple $\geq$ current $S$ tuple, then advance scan of $S$ until current $S$-tuple $\geq$ current $R$ tuple; do this until current $R$ tuple = current $S$ tuple.
  - At this point, all $R$ tuples with same value in $R_i$ (\textit{current $R$ group}) and all $S$ tuples with same value in $S_j$ (\textit{current $S$ group}) match; output $<r, s>$ for all pairs of such tuples.
  - Then resume scanning $R$ and $S$.

- $R$ is scanned once; each $S$ group is scanned once per matching $R$ tuple. (Multiple scans of an $S$ group are likely to find needed pages in buffer.)
Example of Sort-Merge Join

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>28</td>
<td>uppy</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>44</td>
<td>guppy</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
<th>rname</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>103</td>
<td>12/4/96</td>
<td>guppy</td>
</tr>
<tr>
<td>28</td>
<td>103</td>
<td>11/3/96</td>
<td>yuppy</td>
</tr>
<tr>
<td>31</td>
<td>101</td>
<td>10/10/96</td>
<td>dustin</td>
</tr>
<tr>
<td>31</td>
<td>102</td>
<td>10/12/96</td>
<td>lubber</td>
</tr>
<tr>
<td>31</td>
<td>101</td>
<td>10/11/96</td>
<td>lubber</td>
</tr>
<tr>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
<td>dustin</td>
</tr>
</tbody>
</table>

- **Cost:** $M \log M + N \log N + (M+N)$
  - The cost of scanning, $M+N$, could be $M*N$ (very unlikely!)
- With 35, 100 or 300 buffer pages, both Reserves and Sailors can be sorted in 2 passes; total join cost: 7500.

Query Optimization - In class order

- Relational algebra level
- Detailed query plan level
  - Estimate Costs
    - without indexes
    - with indexes
  - Generate and compare plans
Relational algebra optimization

• Transformation rules
  – (preserve equivalence)
• What are good transformations?

Rules: Natural joins & cross products & union

\[ R \bowtie S = S \bowtie R \]
\[ (R \bowtie S) \bowtie T = R \bowtie (S \bowtie T) \]
Note:

- Carry attribute names in results, so order is not important
- Can also write as trees, e.g.:

```
R          S
\   \      \   \       \   \       \   \\
T        R
\       /       \       \\
S       T
```

Rules: Natural joins & cross products & union

- $R \Join S = S \Join R$
- $(R \Join S) \Join T = R \Join (S \Join T)$
- $R \times S = S \times R$
- $(R \times S) \times T = R \times (S \times T)$
- $R \cup S = S \cup R$
- $R \cup (S \cup T) = (R \cup S) \cup T$
**Rules: Selects**

\[ \sigma_{p_1 \land p_2}(R) = \sigma_{p_1} [ \sigma_{p_2}(R)] \]

\[ \sigma_{p_1 \lor p_2}(R) = [ \sigma_{p_1}(R)] \cup [ \sigma_{p_2}(R)] \]

**Bags vs. Sets**

R = \{a,a,b,b,b,c\}
S = \{b,b,c,c,d\}
RUS = ?

- **Option 1** SUM
  RUS = \{a,a,b,b,b,b,b,c,c,c,d\}

- **Option 2** MAX
  RUS = \{a,a,b,b,b,c,c,d\}
Option 2 (MAX) makes this rule work:

\[ \sigma_{p1 \land p2}(R) = \sigma_{p1}(R) \cup \sigma_{p2}(R) \]

**Example:** \( R = \{a,a,b,b,b,c\} \)

- \( P1 \) satisfied by \( a,b \); \( P2 \) satisfied by \( b,c \)

\[ \sigma_{p1 \land p2}(R) = \{a,a,b,b,b,c\} \]

\[ \sigma_{p1}(R) = \{a,a,b,b,b\} \]

\[ \sigma_{p2}(R) = \{b,b,b,c\} \]

\[ \sigma_{p1}(R) \cup \sigma_{p2}(R) = \{a,a,b,b,b,c\} \]

---

“Sum” option makes more sense:

<table>
<thead>
<tr>
<th>Senators (…….)</th>
<th>Rep (…….)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_{yr, state} ) Senators; ( T1 = \pi_{yr, state} ) Senators; ( T1 = \pi_{yr, state} ) Senators; ( T2 = \pi_{yr, state} ) Reps</td>
<td></td>
</tr>
</tbody>
</table>

\begin{array}{|c|c|} \hline Yr & State \ \hline \hline 97 & CA \hline 99 & CA \hline 98 & AZ \hline \end{array} \quad \begin{array}{|c|c|} \hline Yr & State \ \hline \hline 99 & CA \hline 99 & CA \hline 98 & CA \hline \end{array}
Executive Decision

- Use “SUM” option for bag unions
- Some rules cannot be used for bags
### Rules: Project

Let: $X =$ set of attributes  
    $Y =$ set of attributes  
    $XY =$ $X \cup Y$

$$\pi_{xy}(R) = \pi_x[\pi_y(R)]$$

### Rules: $\sigma + \bowtie$ combined

Let $p =$ predicate with only $R$ attrs  
$q =$ predicate with only $S$ attrs  
$m =$ predicate with only $R,S$ attrs

$$\sigma_p (R \bowtie S) = [\sigma_p (R)] \bowtie S$$

$$\sigma_q (R \bowtie S) = R \bowtie [\sigma_q (S)]$$
Some Rules can be Derived:

\[ \sigma_{p \land q} (R \bowtie S) = \]

\[ \sigma_{p \land q \land m} (R \bowtie S) = \]

\[ \sigma_{p v q} (R \bowtie S) = \]

--- Derivation for first one:

\[ \sigma_{p \land q} (R \bowtie S) = \]

\[ \sigma_p [ \sigma_q (R \bowtie S) ] = \]

\[ \sigma_p [ R \bowtie \sigma_q (S) ] = \]

\[ [\sigma_p (R)] \bowtie [\sigma_q (S)] \]
**Rules: \( \pi, \sigma \) combined**

Let \( x \) = subset of \( R \) attributes  
\( z \) = attributes in predicate \( P \)  
(subset of \( R \) attributes)

\[
\pi_x[\sigma_p(R)] = \pi_x\{\sigma_p(\pi_x(R))\}
\]

**Rules: \( \pi, \bowtie \) combined**

Let \( x \) = subset of \( R \) attributes  
\( y \) = subset of \( S \) attributes  
\( z \) = intersection of \( R,S \) attributes

\[
\pi_{xy}(R \bowtie S) = \pi_{xy}\{[\pi_{xz}(R) \bowtie \pi_{yz}(S)]\}
\]

CS54100
\[ \pi_{xy} \{ \sigma_p (R \bowtie S) \} = \]
\[ \pi_{xy} \{ \sigma_p [\pi_{xz'} (R) \bowtie \pi_{yz'} (S)] \} \]
\[ z' = z \cup \{ \text{attributes used in } P \} \]

Rules for \( \sigma, \pi \) combined with \( X \)

similar...

e.g., \( \sigma_p (R \times S) = ? \)
Rules $\sigma, U$ combined:

\[
\sigma_p(R \cup S) = \sigma_p(R) \cup \sigma_p(S)
\]
\[
\sigma_p(R - S) = \sigma_p(R) - S = \sigma_p(R) - \sigma_p(S)
\]

Which are “good” transformations?

\[
\sigma_{p1 \land p2} (R) \rightarrow \sigma_{p1} [\sigma_{p2} (R)]
\]
\[
\sigma_p (R \bowtie S) \rightarrow [\sigma_p (R)] \bowtie S
\]
\[
R \bowtie S \rightarrow S \bowtie R
\]
\[
\pi_x [\sigma_p (R)] \rightarrow \pi_x \{\sigma_p [\pi_{xz} (R)]\}
\]
Conventional wisdom: do projects early

Example: \( R(A,B,C,D,E) \) \( x=\{E\} \)
\( P: (A=3) \land (B=\text{“cat”}) \)

\( \pi_x \{ \sigma_P (R) \} \) vs. \( \pi_E \{ \sigma_P \{ \pi_{ABE}(R) \} \} \)

What if we have \( A, B \) indexes?

B = “cat” \( \rightarrow \) \( \rightarrow \) A = 3

Intersect pointers to get pointers to matching tuples
Bottom line:

- No transformation is always good
- Usually good: early selections

More transformations

- Eliminate common sub-expressions
- Other operations: duplicate elimination
Outline - Query Processing

- Relational algebra level
  - transformations
  - good transformations
- Detailed query plan level
  - estimate costs
  - generate and compare plans

Estimating cost of query plan

1. Estimating size of results
2. Estimating # of IOs
Estimating result size

- 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

Example

<table>
<thead>
<tr>
<th></th>
<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>a</td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

A: 20 byte string  
B: 4 byte integer  
C: 8 byte date  
D: 5 byte string

\( T(R) = 5 \)  \( S(R) = 37 \)

\( V(R,A) = 3 \)  \( V(R,C) = 5 \)

\( V(R,B) = 1 \)  \( V(R,D) = 4 \)
Size estimates for $W = R_1 \times R_2$

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

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

Size estimate for $W = s$

$A = a (R)$

- $S(W) = S(R)$

- $T(W) = ?$
Example

\[
\begin{array}{cccc}
\text{A} & \text{B} & \text{C} & \text{D} \\
\text{cat} & 1 & 10 & \text{a} \\
\text{cat} & 1 & 20 & \text{b} \\
\text{dog} & 1 & 30 & \text{a} \\
\text{dog} & 1 & 40 & \text{c} \\
\text{bat} & 1 & 50 & \text{d} \\
\end{array}
\]

\[
\begin{align*}
V(R,A) &= 3 \\
V(R,B) &= 1 \\
V(R,C) &= 5 \\
V(R,D) &= 4
\end{align*}
\]

\[
W = \sigma_{z=\text{val}}(R) \quad T(W) = \frac{T(R)}{V(R,Z)}
\]
Assumption:

Values in select expression $Z = \text{val}$ are \textit{uniformly distributed} over possible $V(R,Z)$ values.

Alternate Assumption:

Values in select expression $Z = \text{val}$ are \textit{uniformly distributed} over domain with $\text{DOM}(R,Z)$ values.
**Example**

<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>a</td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>a</td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>c</td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>d</td>
</tr>
</tbody>
</table>

Alternate assumption

\[ V(R,A)=3 \quad \text{DOM}(R,A)=10 \]

\[ V(R,B)=1 \quad \text{DOM}(R,B)=10 \]

\[ V(R,C)=5 \quad \text{DOM}(R,C)=10 \]

\[ V(R,D)=4 \quad \text{DOM}(R,D)=10 \]

\[ W = \sigma_{z=\text{val}(R)} \quad T(W) = ? \]

\[ C=\text{val} \Rightarrow T(W) = (1/10)1 + (1/10)1 + \ldots \]
\[ = (5/10) = 0.5 \]

\[ B=\text{val} \Rightarrow T(W) = (1/10)5 + 0 + 0 = 0.5 \]

\[ A=\text{val} \Rightarrow T(W) = (1/10)2 + (1/10)2 + (1/10)1 \]
\[ = 0.5 \]
### Example

<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>a</td>
</tr>
<tr>
<td>cat</td>
<td>1</td>
<td>20</td>
<td>b</td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>30</td>
<td>a</td>
</tr>
<tr>
<td>dog</td>
<td>1</td>
<td>40</td>
<td>c</td>
</tr>
<tr>
<td>bat</td>
<td>1</td>
<td>50</td>
<td>d</td>
</tr>
</tbody>
</table>

Alternate assumption

\[ V(R,A)=3 \quad \text{DOM}(R,A)=10 \]
\[ V(R,B)=1 \quad \text{DOM}(R,B)=10 \]
\[ V(R,C)=5 \quad \text{DOM}(R,C)=10 \]
\[ V(R,D)=4 \quad \text{DOM}(R,D)=10 \]

\[
W = \sigma_{z=\text{val}(R)}(R) \\
T(W) = \frac{T(R)}{\text{DOM}(R,Z)}
\]

### Selection cardinality

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

\[ SC(R,A) = \begin{cases} \\
\frac{T(R)}{V(R,A)} \\
\frac{T(R)}{\text{DOM}(R,A)}
\end{cases} \]
What about $W = \sigma_{z \geq \text{val}} (R)$? $T(W) = ?$

- Solution #1:
  \[ T(W) = \frac{T(R)}{2} \]

- Solution #2:
  \[ T(W) = \frac{T(R)}{3} \]

- Solution #3: Estimate values in range

Example $R$

\[
\begin{array}{c|c}
Z & V(R,Z) = 10 \\
\hline
\text{Min}=1 & W = \sigma_{z \geq 15} (R) \\
\text{Max}=20 & \\
\end{array}
\]

\[ f = \frac{20-15+1}{20-1+1} = \frac{6}{20} \] (fraction of range)

\[ T(W) = f \times T(R) \]
Equivalently:
\[ f \times V(R, Z) = \text{fraction of distinct values} \]
\[ T(W) = [f \times V(Z, R)] \times T(R) = f \times \frac{T(R)}{V(Z, R)} \]

Size estimate for \( W = R_1 \bowtie R_2 \):

- Let \( x = \) attributes of \( R_1 \)
- \( y = \) attributes of \( R_2 \)

**Case 1**
\[ X \cap Y = \emptyset \]

Same as \( R_1 \times R_2 \)
Case 2

\[ W = R_1 \bowtie R_2 \quad X \cap Y = A \]

**Assumption:**

\[ V(R_1,A) \leq V(R_2,A) \Rightarrow \text{Every A value in } R_1 \text{ is in } R_2 \]
\[ V(R_2,A) \leq V(R_1,A) \Rightarrow \text{Every A value in } R_2 \text{ is in } R_1 \]

“containment of value sets”  Sec. 7.4.4

**Computing** \( T(W) \) **when** \( V(R_1,A) \leq V(R_2,A) \)

Take 1 tuple

Match

1 tuple matches with \( \frac{T(R_2)}{V(R_2,A)} \) tuples...

so \[ T(W) = \frac{T(R_2) \times T(R_1)}{V(R_2, A)} \]
V(R1,A) \leq V(R2,A) \quad T(W) = \frac{T(R2) \cdot T(R1)}{V(R2,A)}

V(R2,A) \leq V(R1,A) \quad T(W) = \frac{T(R2) \cdot T(R1)}{V(R1,A)}

[A is common attribute]

In general \quad W = R1 \bowtie R2

T(W) = \frac{T(R2) \cdot T(R1)}{\max\{ V(R1,A), V(R2,A) \}}
Case 2 with alternate assumption

Values uniformly distributed over domain

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

In all cases:

\[
S(W) = S(R1) + S(R2) - S(A)
\]

size of attribute A
Using similar ideas, we can estimate sizes of:

\[ \Pi_{AB}(R) \] ..... Sec. 16.4.2

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

\( R \bowtie S \) with common attributes A,B,C  
Sec. 16.4.5

Union, intersection, diff, .... Sec. 16.4.7

Note: for complex expressions, need intermediate T,S,V results.

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

Treat as relation U

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

Also need V(U, *) !!
To estimate $V_s$

E.g., $U = \sigma_{A=a} (R_1)$

Say $R_1$ has attrs $A, B, C, D$

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

Example

$R_1$

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

$V(R_1, A) = 3$
$V(R_1, B) = 1$
$V(R_1, C) = 5$
$V(R_1, D) = 3$

$U = \sigma_{A=a} (R_1)$

$V(U, A) = 1$  $V(U, B) = 1$  $V(U, C) = T(R_1)$  $V(R_1, A)$

$V(D, U) \ldots$ somewhere in between
Possible Guess  \[ U = \sigma_{A=a}(R) \]

\[ V(U,A) = 1 \]
\[ V(U,B) = V(R,B) \]

For Joins  \[ U = R1(A,B) \bowtie R2(A,C) \]

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

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

- **R1**
  - \( T(R_1) = 1000 \)
  - \( V(R_1,A) = 50 \)
  - \( V(R_1,B) = 100 \)

- **R2**
  - \( T(R_2) = 2000 \)
  - \( V(R_2,B) = 200 \)
  - \( V(R_2,C) = 300 \)

- **R3**
  - \( T(R_3) = 3000 \)
  - \( V(R_3,C) = 90 \)
  - \( V(R_3,D) = 500 \)

Partial Result: \( U = R \bowtie S \)

- \( T(U) = \frac{1000 \times 2000}{200} \)
- \( V(U,A) = 50 \)
- \( V(U,B) = 100 \)
- \( V(U,C) = 300 \)
\[ Z = U \bowtie R3 \]

\[
T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300} \\
V(Z,A) = 50 \\
V(Z,B) = 100 \\
V(Z,C) = 90 \\
V(Z,D) = 500
\]

**Summary**

- Estimating size of results is still a mix of science and art
  - *Research opportunity?*
- But it does work reasonably well
- Don’t forget:
  Statistics must be kept up to date…
  (cost?)
Outline

• Estimating cost of query plan
  – Estimating size of results done!
  – Estimating # of IOs next…

• Generate and compare plans