Cost Models

• An optimizer estimates costs for plans so that it can choose the least expensive plan from a set of alternatives.
• Inputs to the cost model include:
  • the query
  • database statistics
  • description of computational resources, e.g.
    • CPU speed
    • costs of sequential and random disk operations
    • size of buffer pool, amount of memory available for query operators
  • concurrency environment, e.g., number of concurrent queries
  • system configuration parameters
Query Optimization

What is Cost??

• a cost model assigns a number (the cost) to each query, but what does that number represent?

• some possibilities:
  • query response time
  • total computing resource consumption for query execution
  • dollar cost of executing the query

• a common approach is to use total resource consumption:

\[
\text{cost}(Q) = \text{CPUCost}(Q) + \text{DiskCost}(Q) + \text{NetworkCost}(Q)
\]

where \( \text{CPUCost}(Q) \) is an estimate of the total CPU time required to execute the query, \( \text{DiskCost}(Q) \) is an estimate of the total time required for all disk I/O operations for the query, and so on.
Costing Access Methods

- Consider access to the relation $A$ in the optimization example: $\sigma_{\text{EmStDate like '82%'} \land \text{EmpTime} \geq 0.5}(A)$

- these three access methods were possible:
  
  **Method A1**: table scan, then
  $$\sigma_{\text{EmStDate like '82%'} \land \text{EmpTime} \geq 0.5}(A)$$

  **Method A2**: index scan using unclustered index on $A$.EmStDate, then $\sigma_{\text{EmpTime} \geq 0.5}$(A)

  **Method A3**: index scan using unclustered index on $A$.EmpNo, then
  $$\sigma_{\text{EmStDate like '82%'} \land \text{EmpTime} \geq 0.5}(A)$$

- To estimate the costs of these methods, the optimizer needs to answer some basic questions, e.g.:
  
  - how many tuples in $A$? How many blocks?
  - how large are the indexes? How many leaves? How deep?
  - how many of the tuples will satisfy the conditions?
Database Statistics

- To support costing, the DBMS maintains basic statistics about the database in its catalog. For example:
  - number of rows and tuples in each table
  - number of key values, levels, leaf pages in each index
- In addition, to help answer questions such as “how many tuples have EmpStDate like ‘82%’, the DBMS maintains information about the values in some or all of the columns of the table. For example:
  - number of distinct values
  - minimum and maximum values
  - quantiles, histograms or similar structures describing the distribution of different values for that column

Updates

The DBMS must have some means of maintaining these statistics as the underlying database is updated.
Database Statistics in DB2

```
db2 "select colname,colcard,high2key,low2key
from sysstat.columns where tabname = 'EMPLOYEE'
```

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIRTHDATE</td>
<td>30</td>
<td>'1955-04-12'</td>
<td>'1926-05-17'</td>
</tr>
<tr>
<td>BONUS</td>
<td>8</td>
<td>+0000900.00</td>
<td>+0000400.00</td>
</tr>
<tr>
<td>COMM</td>
<td>32</td>
<td>+0003720.00</td>
<td>+0001272.00</td>
</tr>
<tr>
<td>EDLEVEL</td>
<td>8</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>EMPNO</td>
<td>32</td>
<td>'000330'</td>
<td>'000020'</td>
</tr>
<tr>
<td>FIRSTNME</td>
<td>30</td>
<td>'WILLIAM'</td>
<td>'CHRISTINE'</td>
</tr>
<tr>
<td>HIREDATE</td>
<td>31</td>
<td>'1980-06-19'</td>
<td>'1949-08-17'</td>
</tr>
<tr>
<td>JOB</td>
<td>8</td>
<td>'PRES'</td>
<td>'CLERK'</td>
</tr>
<tr>
<td>LASTNAME</td>
<td>31</td>
<td>'WALKER'</td>
<td>'BROWN'</td>
</tr>
<tr>
<td>MIDINIT</td>
<td>20</td>
<td>'W'</td>
<td>'A'</td>
</tr>
<tr>
<td>PHONENO</td>
<td>32</td>
<td>'9001'</td>
<td>'0942'</td>
</tr>
<tr>
<td>SALARY</td>
<td>32</td>
<td>+0046500.00</td>
<td>+0015900.00</td>
</tr>
<tr>
<td>SEX</td>
<td>2</td>
<td>'M'</td>
<td>'F'</td>
</tr>
<tr>
<td>WORKDEPT</td>
<td>8</td>
<td>'E11'</td>
<td>'B01'</td>
</tr>
</tbody>
</table>
Selectivity Estimation

- an important problem for the optimizer is estimating the selectivity of query predicates
- the filter factor of a predicate $C$ applied to relation $R$ is the fraction of $R$’s tuples that satisfy $C$

$$\frac{|\sigma_C(R)|}{|R|}$$

- filter factors (selectivity) can be estimated
  - using basic statistics about the columns in $C$
  - using histograms on the columns in $C$
  - using sampling
Selectivity Estimation

If no other information is available, selectivity can be estimated using basic column statistics, e.g.:

- \(|\sigma_{R.a=c}(R)| \approx \frac{1}{\text{distinct}(R.a)}\)
- \(|\sigma_{R.a\leq c}(R)| \approx \frac{c - \min(R.a)}{\max(R.a) - \min(R.a)}\)
- \(|\sigma_{C_1 \land C_2}(R)| \approx |\sigma_{C_1}(R)| \cdot |\sigma_{C_2}(R)|\)

These formulas are based on assumptions, e.g.,
- **uniformity** assumptions
- **independence** assumptions

To the extent that these are incorrect, such estimates may be inaccurate.
Two Basic Types of Histograms

**equi-width histogram**

- all buckets ranges are the same width
- store: frequency for each bucket

**equi-depth**

- all buckets have the same freq
- store: bucket boundaries

---

Cheriton School of Computer Science

CS448 Database Systems Implementation (Winter 2011)
Compressed Histograms

- When there is data skew, it is particularly important to have accurate estimates of the number of occurrences of common values.
- In a compressed histogram, some space is devoted to keeping exact counts for the most frequently occurring values. A regular histogram (e.g., equi-depth) is then used to estimate the frequency of other, less frequent values.
**Histograms in DB2**

```sql
db2 "select seqno,colvalue,valcount from sysstat.coldist where tabname = 'EMPLOYEE' and colname = 'SALARY'"
```

<table>
<thead>
<tr>
<th>SEQNO</th>
<th>COLVALUE</th>
<th>VALCOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+0015340.00</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>+0015900.00</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>+0017750.00</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>+0018270.00</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>+0019950.00</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>+0021340.00</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>+0022180.00</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>+0036170.00</td>
<td>26</td>
</tr>
<tr>
<td>17</td>
<td>+0038250.00</td>
<td>27</td>
</tr>
<tr>
<td>18</td>
<td>+0040175.00</td>
<td>29</td>
</tr>
<tr>
<td>19</td>
<td>+0046500.00</td>
<td>31</td>
</tr>
<tr>
<td>20</td>
<td>+0052750.00</td>
<td>32</td>
</tr>
</tbody>
</table>
Complex Predicates

- simple estimation rules or histograms can be used to estimate the selectivity of predicates involving a single attribute
- what about multi-attribute predicates? For example:
  \[ C_1 \land C_2 \land C_3 \]
- some possibilities:
  - combine single attribute estimates, e.g., using independence assumption
  - multi-dimensional histograms
  - tuple sampling
- challenge is to obtain quick and accurate estimates using small synopses (histograms or samples)
Selectivity Estimation via Sampling

- main idea: sample a small set of tuples from a large relation
  - can be done on-demand, when a query is being optimized
  - alternatively, sample can be drawn in advance, stored, and used to estimate selectivity for multiple queries
- to estimate the selectivity of an arbitrary predicate:
  1. measure the number of sample tuples that satisfy the predicate
  2. extrapolate the measurement to the whole relation
Estimating the Cost of An Access Method

Access method $A_2$ for $\sigma_{\text{EmStDate like '82%' \land EmpTime \geq 0.5(\text{A})}}$:

- index scan for tuples with EmStDate like ‘82%’ using unclustered index on EmStDate, then apply predicate $\text{EmpTime} \geq 0.5$

Estimate cost (total CPU + disk I/O time) of $A_2$:

- estimate numbers of tuples scanned and selected (selectivity estimation)
- estimate number of disk blocks read, and whether read sequentially or randomly
- for disk I/O time, charge fixed amount per random disk read, and (smaller) fixed amount per sequential disk read
- for CPU time, charge fixed amount per block read, fixed amount per tuple read from index, and fixed amount per output tuple
- total cost is sum of disk I/O time and CPU time.
Join Size Estimation

- Another important problem is estimating the size of joins.
- We know:
  \[0 \leq |R \bowtie S| \leq |R| \cdot |S|\]
- The ratio
  \[\frac{|R \bowtie S|}{|R| \cdot |S|}\]
  is sometimes called the **join selectivity**
- Several techniques may be used for join size estimation, sometimes in combination:
  - exploit schema information, e.g., for foreign key joins (a common case)
  - exploit histograms if available
  - estimate using simple statistics only
Join Size Estimation (cont’d)

foreign key joins: Consider $P \bowtie_{(\text{RespEmp}=\text{EmpNo})} E$. Since EmpNo is the key of $E$, the join size may be estimated as $|P|$.

using histograms: If histograms are available on the join keys, they can be used to upper-bound the join selectivity. For example, in $R \bowtie_{R.a=S.b} S$, each $t \in R$ can join with tuples from the $S.b$ histogram bucket into which $t.a$ would fall.

using simple statistics: One way to estimate the join selectivity of $R \bowtie_{R.a=S.b} S$ is

$$min\left(\frac{1}{\text{distinct}(R.a)}, \frac{1}{\text{distinct}(S.b)}\right)$$
Query Optimization

Estimating Plan Cost (DB2 Example)

- estimate plan cost by estimating costs of plan operators
- need properties (e.g., size distribution) of intermediate results to estimate costs of non-leaf operators

```
0.639622
  SORT
  66.5245
  2.63962
   | 0.639622 <-- estimated rows
NLJOIN
  66.5166 <-- cumulative cost
  2.63962 <-- cumulative I/Os
/-------------------\
  3.2 0.199882
TBSCAN FETCH
  50.3429 5.11925
   2 0.199882
    | /-------\
```