Cost-Based Optimization

*Database Systems: The Complete Book*
Ch 2.3, 6.1-6.4, 15, 16.4-16.5
Optimizing
Optimizing

• Some equivalence rules are *always* good…

• Which?
Optimizing

- Some equivalence rules are always good…
  - Which?

- Some equivalence rules are sometimes good
  - Which?
  - What do we do about it?
Cost Estimation

- Compare many different plans by...
  - ... actually running the query
  - ... estimating the plan’s “cost”
I'm just outside town, so I should be there in fifteen minutes.

Actually, it's looking more like six days.

No, wait, thirty seconds.

The author of the Windows file copy dialog visits some friends.
Costs
Costs

- Memory Cost (Working Set Size)
Costs

- Memory Cost (Working Set Size)
- Compute Cost ("Big-O")
Costs

- Memory Cost (Working Set Size)
- Compute Cost ("Big-O")
- IO Cost (Pages read, Pages written)
The variable in all of these costs is the arity (size) of a relation.
How do you compute Arities?

• Heuristic Assumptions (Pick a “good enough” RF)
• Summary Statistics About The Data…
  • Upper/Lower Bounds or Value Domains
  • Distribution Summaries (Histograms)
• Data Sampling
How do you compute Arities?

There is no perfect solution (yet)!
How do you compute Arities?

There is **no** perfect solution (yet)!

We don’t need a perfect solution…
… we just need one that’s good enough
Summary Statistics

• Per-Attribute Bounds / Domain Statistics
  • Assume a Uniform Distribution.

• Per-Attribute Histograms
  • Use the histogram to model the data distribution

• Data Samples
  • Use the samples to measure the RF
Uniform Distribution

\[ A = 1 \]
Uniform Distribution

\[ A = 1 \]

Chance of Hit = \( \frac{1}{\# \text{ of distinct values of } A} \)
Uniform Distribution

$$A \in (1, 2, \ldots)$$
Uniform Distribution

\[ A \in (1, 2, \ldots) \]

Chance of Hit = \[ \left| \{(1,2,3,\ldots)\} \right| / \text{# of distinct values of } A \]
Uniform Distribution

$A < 3$
Uniform Distribution

\[ A < 3 \]

Chance of Hit = \( \frac{3 - \text{Low}(A)}{\text{High}(A) - \text{Low}(A)} \)
Uniform Distribution

\[ R.A = S.B \]
Uniform Distribution

\( \bowtie R.A = S.B \)

Chance of Hit Per B = \( \frac{1}{\text{# Distinct Values of A}} \)

Chance of Hit Per B = 1 (If B is a FK Reference)

Chance of Hit Per A = 1 (If A is a FK Reference)
Let’s apply it

```sql
SELECT O.Rank, COUNT(*),
FROM Officers O
WHERE O.Rank >= 2
    AND O.Age > 20 AND O.Age < 30
GROUP BY O.Rank
HAVING COUNT(DISTINCT O.Ship) > 2
```

What is the relational algebra plan for this expression?
Stats

O.Rank: 0-5 (Increments of 0.5; 11 total values)

O.Age: 16-100 (Increments of 1; 85 total values)

Officers: 40,000 tuples (over 500 pages)

Tree Indexes available over O.Age, O.Rank

What is the total cost in IOs?
What is the total cost in CPU/Tuples?
Histograms

Uniform Distributions are a strong assumption!
(data is often skewed)
## Histograms

### People

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Alice”</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>“Bob”</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>“Carol”</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>“Dave”</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>“Eve”</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>“Fred”</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>“Gwen”</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>“Harry”</td>
<td>20</td>
<td>3</td>
</tr>
</tbody>
</table>

```sql
SELECT Name
FROM People
WHERE Rank = 3
AND Age = 20

... AND Age = 19
```

\[
\text{RF}_{\text{Age}} = \frac{1}{n\text{keys}} = \frac{1}{4}
\]

\[
\text{RF}_{\text{Rank}} = \frac{1}{n\text{keys}} = \frac{1}{3}
\]

Age is best!
Histograms

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Alice”</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>“Bob”</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>“Carol”</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>“Dave”</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>“Eve”</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>“Fred”</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>“Gwen”</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>“Harry”</td>
<td>20</td>
<td>3</td>
</tr>
</tbody>
</table>

```
SELECT Name
FROM People
WHERE Rank = 3
    AND Age = 20
    VS
...    
    AND Age = 19

RF_{Age-20} = \frac{1}{2}
RF_{Rank} = \frac{1}{3}

Age is worst!
```
Histograms

People

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Carol</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>Eve</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Fred</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>Gwen</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>Harry</td>
<td>20</td>
<td>3</td>
</tr>
</tbody>
</table>

SELECT Name
FROM People
WHERE Rank = 3
AND Age = 20

VS

AND Age = 19

$RF_{Age=19} = \frac{1}{8}$

$RF_{Rank} = \frac{1}{3}$

Age is best!
Histograms
Histograms
Histograms
Histograms

SELECT ... WHERE A = 33
SELECT ... WHERE A > 33
Using Constraints

• A Key attribute has one distinct value per row (equality selects exactly one row)

• Foreign Key joins generate one row for each row in the referencing relation.

• Cascade relationship guarantees EXACTLY one row per reference.
Sampling

• Take a bunch of tuples from each relation.

• Run 2-3 different query plans on these tuples.

• Estimate the sampling factors for each operator in the plan based on how many survive.
Sampling

How big is a “bunch?”
Sampling

• **Problem**: Very Selective Predicates

• **Problem**: Joins and the Birthday Paradox

• **Problem**: Counting Aggregate Groups
Very Selective Predicates
Very Selective Predicates

![Diagram](image-url)
Very Selective Predicates
Join Conditions
Join Conditions

Birthday Paradox
Need $O(\sqrt{|R|+|S|})$ tuples to reliably guess RF for equijoin
Estimating Join Costs

How many query plans are there?

R □ S □ T □ U
Estimating Join Costs

There are \((N-1)!\) (factorial) different ways (plans) to evaluate this join.

Computing costs for all of these plans is expensive!
Left-Deep Plans

RHS Join Input is always a relation

1) Shrinks join search space
2) Allows index scans/lookups

Technique Pioneered by the System R Optimizer
In Practice

Heuristics, Histograms and Sampling are “good enough” to optimize the common cases.
In Practice

Heuristics, Histograms and Sampling are “good enough” to optimize the common cases.

Some relational databases have manual overrides.
Oracle

```sql
SELECT /*+ INDEX (employees emp_department_ix)*/
  employee_id, department_id
FROM employees
WHERE department_id > 50;
```
SELECT attname, inherited, n_distinct,
array_to_string(most_common_vals, E'\n') as most_common_vals
FROM pg_stats
WHERE tablename = 'road';
In Practice

Heuristics, Histograms and Sampling are “good enough” to optimize the common cases.
In Practice

Heuristics, Histograms and Sampling are “good enough” to optimize the common cases.

Some relational databases have manual overrides.
In Practice

Heuristics, Histograms and Sampling are “good enough” to optimize the common cases.

Some relational databases have manual overrides.

All relational databases have an “EXPLAIN” operator
EXPLAIN SELECT sum(i) FROM foo WHERE i < 10;

<table>
<thead>
<tr>
<th>QUERY PLAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate   (cost=23.93..23.93 rows=1 width=4)</td>
</tr>
<tr>
<td>-&gt; Index Scan using fi on foo (cost=0.00..23.92 rows=6 width=4)</td>
</tr>
<tr>
<td>Index Cond: (i &lt; 10)</td>
</tr>
</tbody>
</table>
Backup Slides
# Join Algorithm Comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Can Support Pipelining?</th>
<th>But?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid Hash</td>
<td>Yes</td>
<td>RHS Hash Table needs to fit in memory</td>
</tr>
<tr>
<td>Index Nested Loop</td>
<td>Yes</td>
<td>RHS Table needs an index on the join key</td>
</tr>
<tr>
<td>Sort/Merge Join</td>
<td>Yes</td>
<td>LHS and RHS must both be sorted on the join key</td>
</tr>
<tr>
<td>(Block) Nested Loop</td>
<td>Yes</td>
<td>RHS Table needs to fit in memory</td>
</tr>
<tr>
<td>Hash Join</td>
<td>No</td>
<td>No buts. Hash Join always materializes</td>
</tr>
</tbody>
</table>
Join Algorithm IO Costs

**$R \bowtie S$**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>IO Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid Hash</td>
<td>[#pages of S] (if fits in mem)</td>
</tr>
<tr>
<td>Index Nested Loop</td>
<td>$</td>
</tr>
<tr>
<td>Sort/Merge Join</td>
<td>[#pages of S] (+sorting costs)</td>
</tr>
<tr>
<td>Nested Loop</td>
<td>[#pages of S] (if fits in mem)</td>
</tr>
<tr>
<td>Block Nested Loop</td>
<td>[#pages of R] + [#of block pairs] $\times$</td>
</tr>
<tr>
<td>Hash Join</td>
<td>$2 \times [#pages of R] + [#pages of S]$</td>
</tr>
</tbody>
</table>

42
## Data Access IO Costs

<table>
<thead>
<tr>
<th>Index Type</th>
<th>Full Scan</th>
<th>Range Scan</th>
<th>Lookup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw File</td>
<td>$N$</td>
<td>$N$</td>
<td>$N$</td>
</tr>
<tr>
<td>Sorted File</td>
<td>$N$</td>
<td>$\log_2(N)+</td>
<td>R</td>
</tr>
<tr>
<td>Static Hash Index</td>
<td>$&gt;N$</td>
<td>$&gt;N$</td>
<td>$\approx 1$</td>
</tr>
<tr>
<td>Extendible Hash Index</td>
<td>$&gt;N+</td>
<td>D</td>
<td>$</td>
</tr>
<tr>
<td>Linear Hash Index</td>
<td>$&gt;N$</td>
<td>$&gt;N$</td>
<td>$\approx 1$</td>
</tr>
<tr>
<td>ISAM Tree Index</td>
<td>$\approx N$</td>
<td>$\approx \log_{</td>
<td>T</td>
</tr>
<tr>
<td>B+ Tree Index</td>
<td>$N$</td>
<td>$\log_{</td>
<td>T</td>
</tr>
</tbody>
</table>

Note: $|D|$ represents the number of disk operations, $|T|$ represents the tree size, and $|R|$ represents the number of records.