Lifetime of a Query

SQL Query \( \rightarrow \) Scanner/Parser \( \rightarrow \) Query In Intermediate Form \( \rightarrow \) Optimizer \( \rightarrow \) Logical Query Plan

\( \rightarrow \) DBMS Catalog Information

\( \rightarrow \) Optimizer Candidate Plan Generator Cost Estimator


\( \rightarrow \) Code Generator

\( \rightarrow \) Query Execution Code

\( \rightarrow \) Query Processor

\( \rightarrow \) Query Result
Example(s):

```sql
select sname
from s, p, spj
where s.sno = spj.sno
  and spj.pno = p.pno
  and status <> 0
  and pname = 'Nut';
```
Rule-based Optimization (RBO)

Example(s):
Cost-based Optimization (CBO)

Algorithms for Project and Select
Join Algorithms (1 / 5)

#1: Nested-Loops Join (NLJ)

Join Algorithms (2 / 5)

#2: Sort-Merge Join (SMJ)

The Algorithm:

Make the first tuple of relation r the current tuple
Loop until one of the relations is exhausted:
  Make a set of all tuples from relation r that have the same join values as the current tuple
  For each tuple from relation s that has a join match:
    Output marriages of it to all set members
  The next tuple from relation r that is not a member of the set becomes the new current tuple
#2: Sort-Merge Join (cont.)

Example(s):

#3: Hash Join

The Algorithm:

Hash each relation using the same hash function on the join attributes of the tuples.
For each of the $M$ corresponding pairs of buckets:
  Build an in-memory hash index on the contents of the first bucket in the pair
  For each tuple in the second bucket:
    Probe the index with its join attribute
    For each matching tuple found:
      Output the marriage
Examining Query Plans (1 / 2)

In Oracle 10: Prefix query with `explain plan for`.

```
explain plan for
select sname
  from s, spj, p
  where s.sno = spj.sno
    and spj.pno = p.pno
    and status <> 0
    and pname = ’Nut’;
```

And then issue the following to see the generated plan:

```
set linesize 100
set pagesize 0
select plan_table_output
  from table(dbms_xplan.display(’plan_table’,null,’serial’));
```
The (slightly abbreviated) output:

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>4</td>
<td>104</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td></td>
<td>4</td>
<td>104</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td>4</td>
<td>56</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS FULL</td>
<td>P</td>
<td>1</td>
<td>8</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>INDEX FULL SCAN</td>
<td>SYS_</td>
<td>4</td>
<td>24</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>S</td>
<td>5</td>
<td>60</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

“Vacuuming” a Database (1 / 3)

How to wipe your DBMS' mind:

In Oracle 10:

```sql
exec DBMS_STATS.DELETE_SCHEMA_STATS('mccann');
select count(*) from user_histograms; ==> 0
```

In Postgres:

```sql
vacuum;
```
Without its statistics, Oracle generates a different plan:

```
<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost(%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>4</td>
<td>220</td>
<td>7 (15)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*1</td>
<td>HASH JOIN</td>
<td></td>
<td>4</td>
<td>220</td>
<td>7 (15)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>MERGE JOIN CARTESIAN</td>
<td></td>
<td>5</td>
<td>230</td>
<td>4 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*3</td>
<td>TABLE ACCESS FULL P</td>
<td></td>
<td>1</td>
<td>17</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>BUFFER SORT</td>
<td></td>
<td>5</td>
<td>145</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*5</td>
<td>TABLE ACCESS FULL S</td>
<td></td>
<td>5</td>
<td>145</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>6</td>
<td>TABLE ACCESS FULL SPJ</td>
<td></td>
<td>24</td>
<td>216</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>
```

We can command Oracle to collect new stats:
```
exec DBMS_STATS.GATHER_SCHEMA_STATS('mccann');
```
```
select count(*) from user_histograms;  ==> 180
```
And try again:
```
<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost(%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>4</td>
<td>104</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*1</td>
<td>HASH JOIN</td>
<td></td>
<td>4</td>
<td>104</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td>4</td>
<td>56</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*3</td>
<td>TABLE ACCESS FULL P</td>
<td></td>
<td>1</td>
<td>8</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*4</td>
<td>INDEX FULL SCAN SYS_</td>
<td></td>
<td>4</td>
<td>24</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*5</td>
<td>TABLE ACCESS FULL S</td>
<td></td>
<td>5</td>
<td>60</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>
```
Back to the original plan!
Influencing the Query Optimizer (1 / 6)

One way to nudge the DBMS: Rewrite your queries.

Influencing the Query Optimizer (2 / 6)

Oracle has a different scheme: HINTS
Influencing the Query Optimizer (3 / 6)

Example: **RULE**

```
explain plan for select /*+ RULE */ sname from s, p, spj
  where s.sno = spj.sno and spj.pno = p.pno
  and status <> 0 and pname = 'Nut';
```

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS FULL</td>
<td>SPJ</td>
</tr>
</tbody>
</table>
* 4 | TABLE ACCESS BY INDEX ROWID | P           |
* 5 | INDEX UNIQUE SCAN    | SYS_C00665352  |
* 6 | TABLE ACCESS BY INDEX ROWID | S           |
* 7 | INDEX UNIQUE SCAN    | SYS_C00665351  |

Note: - rule based optimizer used (consider using cbo)

Influencing the Query Optimizer (4 / 6)

Example: **ALL_ROWS**

```
explain plan for select /*+ ALL_ROWS */ sname from s, p, spj
  where s.sno = spj.sno and spj.pno = p.pno
  and status <> 0 and pname = 'Nut';
```

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost(%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>4</td>
<td>104</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>
* 1 | HASH JOIN             |      | 4    | 104   | 6 (17)     | 00:00:01      |
| 2  | NESTED LOOPS         | P    | 4    | 56    | 3 (0)      | 00:00:01      |
* 3 | TABLE ACCESS FULL    | P    | 1    | 8     | 2 (0)      | 00:00:01      |
* 4 | INDEX FULL SCAN      | SYS_ | 4    | 24    | 1 (0)      | 00:00:01      |
* 5 | TABLE ACCESS FULL    | S    | 5    | 60    | 2 (0)      | 00:00:01      |
Influencing the Query Optimizer (5 / 6)

Example: **FIRST_ROWS**

```
explain plan for select /*+ FIRST_ROWS */ sname
  from s, p, spj
where s.sno = spj.sno and spj.pno = p.pno
  and status <> 0 and pname = 'Nut';
```

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost(%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>4</td>
<td>104</td>
<td>7 (0)</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>NESTED LOOPS</td>
<td></td>
<td>4</td>
<td>104</td>
<td>7 (0)</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td>4</td>
<td>56</td>
<td>3 (0)</td>
<td>...</td>
</tr>
<tr>
<td>*3</td>
<td>TABLE ACCESS FULL</td>
<td>P</td>
<td>1</td>
<td>8</td>
<td>2 (0)</td>
<td>...</td>
</tr>
<tr>
<td>*4</td>
<td>INDEX FULL SCAN</td>
<td>SYS_</td>
<td>4</td>
<td>24</td>
<td>1 (0)</td>
<td>...</td>
</tr>
<tr>
<td>*5</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>S</td>
<td>1</td>
<td>12</td>
<td>1 (0)</td>
<td>...</td>
</tr>
<tr>
<td>*6</td>
<td>INDEX UNIQUE SCAN</td>
<td>SYS_</td>
<td>1</td>
<td></td>
<td>0 (0)</td>
<td>...</td>
</tr>
</tbody>
</table>

Query Optimization – CSc 460 v1.0 (McCann) – p. 23

Influencing the Query Optimizer (6 / 6)

Example: **USE_MERGE()**

```
explain plan for select /*+ USE_MERGE(s,p) */ sname
  from p, spj, s
where s.sno = spj.sno and spj.pno = p.pno
  and status <> 0 and pname = 'Nut';
```

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost(%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>4</td>
<td>104</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*1</td>
<td>HASH JOIN</td>
<td></td>
<td>4</td>
<td>104</td>
<td>6 (17)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>MERGE JOIN CARTESIAN</td>
<td></td>
<td>5</td>
<td>100</td>
<td>4 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*3</td>
<td>TABLE ACCESS FULL</td>
<td>P</td>
<td>1</td>
<td>8</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>BUFFER SORT</td>
<td></td>
<td>5</td>
<td>60</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>*5</td>
<td>TABLE ACCESS FULL</td>
<td>S</td>
<td>5</td>
<td>60</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>6</td>
<td>INDEX FULL SCAN</td>
<td>SYS_</td>
<td>24</td>
<td>144</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Query Optimization – CSc 460 v1.0 (McCann) – p. 24