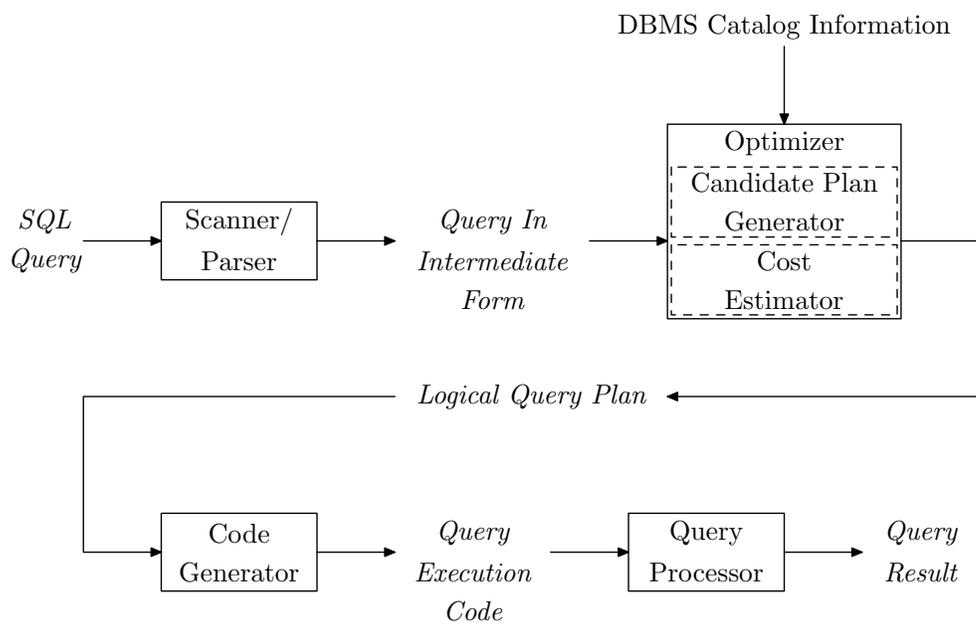


# Topic 15:

## Query Optimization

Query Optimization – CSc 460 v1.1 (McCann) – p. 1/24

## Lifetime of a Query



Query Optimization – CSc 460 v1.1 (McCann) – p. 2/24

## Query Trees (1 / 2)

---

After scanning/parsing, the query is in standard form, in which the operation order is:

Query Optimization – CSc 460 v1.1 (McCann) – p. 3/24

## Query Trees (2 / 2)

---

### Example(s):

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

Query Optimization – CSc 460 v1.1 (McCann) – p. 4/24

# Query Optimization

## Definition: Query Optimization

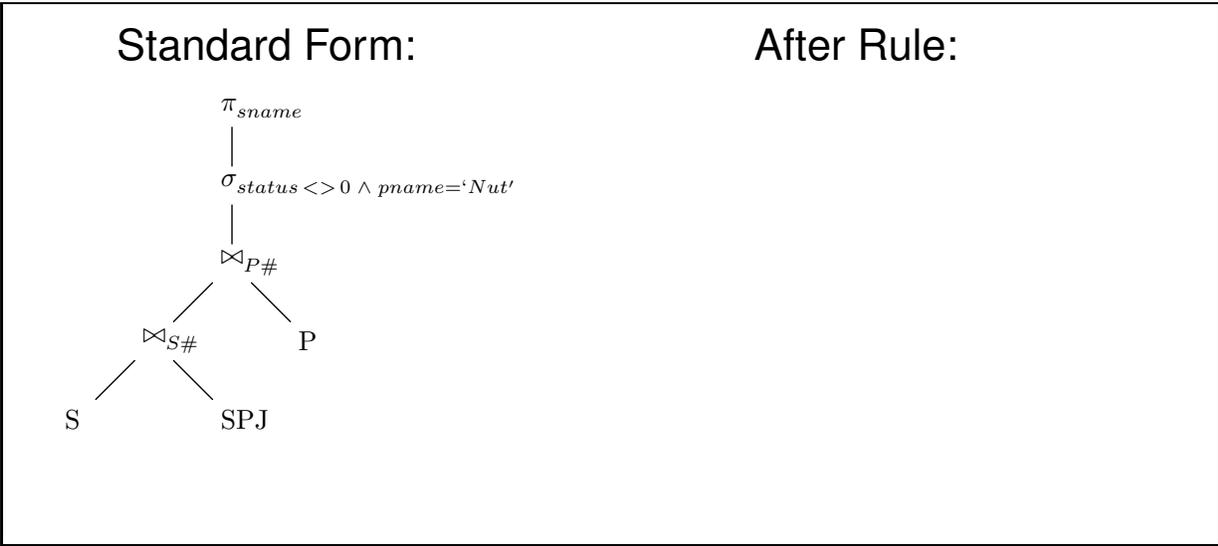
.....

.....

# Rule-based Optimization (RBO)

Idea: Apply “Rules of Thumb” that often improve execution efficiency.

**Example(s):** A classic: Perform selects as soon as possible.



# Cost-based Optimization (CBO)

---

Idea: Estimate the expense of executing each operator/algorithm

Query Optimization – CSc 460 v1.1 (McCann) – p. 7/24

## Algorithms for Project and Select

---

Project ( $\pi$ ): Just one option:

Select ( $\sigma$ ): Three main options:

Query Optimization – CSc 460 v1.1 (McCann) – p. 8/24

# Join Algorithms (1 / 5)

---

## #1: Nested-Loops Join (NLJ)

<i>r</i>	<table border="1"><thead><tr><th>X</th><th>Y</th></tr></thead><tbody><tr><td>D</td><td>3</td></tr><tr><td>C</td><td>4</td></tr><tr><td>B</td><td>1</td></tr><tr><td>D</td><td>2</td></tr></tbody></table>	X	Y	D	3	C	4	B	1	D	2	<i>s</i>	<table border="1"><thead><tr><th>X</th><th>Z</th></tr></thead><tbody><tr><td>D</td><td>8</td></tr><tr><td>B</td><td>7</td></tr><tr><td>D</td><td>6</td></tr><tr><td>A</td><td>1</td></tr><tr><td>B</td><td>1</td></tr></tbody></table>	X	Z	D	8	B	7	D	6	A	1	B	1
X	Y																								
D	3																								
C	4																								
B	1																								
D	2																								
X	Z																								
D	8																								
B	7																								
D	6																								
A	1																								
B	1																								

Query Optimization – CSc 460 v1.1 (McCann) – p. 9/24

# Join Algorithms (2 / 5)

---

## #2: Sort-Merge Join (SMJ)

- Precondition:

### 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

## Join Algorithms (3 / 5)

---

### #2: Sort-Merge Join (cont.)

**Example(s):** Same data as the NLJ example, but sorted on X:

$r$	X	Y	$s$	X	Z
	B	1		A	1
	C	4		B	7
	D	3		B	1
	D	2		D	8
				D	6

Query Optimization – CSc 460 v1.1 (McCann) – p. 11/24

## Join Algorithms (4 / 5)

---

### #3: Hash Join

- Idea:

#### 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

Query Optimization – CSc 460 v1.1 (McCann) – p. 12/24

# Join Algorithms (5 / 5)

---

## #3: Hash Join (cont.)

**Example(s):** NLJ data hashed by even/odd ASCII values:

<i>r</i>	<table border="1"><thead><tr><th>X</th><th>Y</th></tr></thead><tbody><tr><td>D</td><td>3</td></tr><tr><td>C</td><td>4</td></tr><tr><td>B</td><td>1</td></tr><tr><td>D</td><td>2</td></tr></tbody></table>	X	Y	D	3	C	4	B	1	D	2	<i>s</i>	<table border="1"><thead><tr><th>X</th><th>Z</th></tr></thead><tbody><tr><td>D</td><td>8</td></tr><tr><td>B</td><td>7</td></tr><tr><td>D</td><td>6</td></tr><tr><td>A</td><td>1</td></tr><tr><td>B</td><td>1</td></tr></tbody></table>	X	Z	D	8	B	7	D	6	A	1	B	1
X	Y																								
D	3																								
C	4																								
B	1																								
D	2																								
X	Z																								
D	8																								
B	7																								
D	6																								
A	1																								
B	1																								

Query Optimization – CSc 460 v1.1 (McCann) – p. 13/24

# Examining Query Plans (1 / 2)

---

In Oracle 11: 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'));
```

Query Optimization – CSc 460 v1.1 (McCann) – p. 14/24

# Examining Query Plans (2 / 2)

---

The (slightly abbreviated) Oracle 11 output:

Explained.

Plan hash value: 3135777751

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	104	8 (25)	00:00:01
* 1	HASH JOIN		4	104	8 (25)	00:00:01
2	MERGE JOIN		4	56	4 (25)	00:00:01
* 3	TABLE ACCESS BY INDEX ROWID	P	1	8	2 (0)	00:00:01
4	INDEX FULL SCAN	SYS_C0011137	6		1 (0)	00:00:01
* 5	SORT JOIN		24	144	2 (50)	00:00:01
6	INDEX FULL SCAN	SYS_C0011150	24	144	1 (0)	00:00:01
* 7	TABLE ACCESS FULL	S	5	60	3 (0)	00:00:01

Legend:

- 'TABLE ACCESS BY INDEX ROWID' — use index to locate tuple(s)
- 'INDEX FULL SCAN' — get all tuple IDs (in ascending order)
- 'SORT JOIN' — sorts tuples ahead of SMJ
- 'TABLE ACCESS FULL' — read all rows

Query Optimization – CSc 460 v1.1 (McCann) – p. 15/24

---

## “Vacuuming” a Database (1 / 3)

---

The Idea: Tell the DBMS to update what it knows about relations.

---

But first: Here’s how to wipe your DBMS’ mind!

In Oracle 11:

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

In Postgres:

```
vacuum;
```

Query Optimization – CSc 460 v1.1 (McCann) – p. 16/24

## “Vacuuming” a Database (2 / 3)

Without its statistics, Oracle 11 generates a different plan:

Plan hash value: 2693896614

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	220	9 (12)	00:00:01
* 1	HASH JOIN		4	220	9 (12)	00:00:01
2	MERGE JOIN CARTESIAN		5	230	6 (0)	00:00:01
* 3	TABLE ACCESS FULL	P	1	17	3 (0)	00:00:01
4	BUFFER SORT		5	145	3 (0)	00:00:01
* 5	TABLE ACCESS FULL	S	5	145	3 (0)	00:00:01
6	INDEX FAST FULL SCAN	SYS_C0011150	24	216	2 (0)	00:00:01

Legend (cont.):

- ‘MERGE JOIN CARTESIAN’ — performs only a Cartesian Product
- ‘BUFFER SORT’ — just buffering tuples (not always a sort!)
- ‘INDEX FAST FULL SCAN’ — use multiblock reads (no ordering)

Query Optimization – CSC 460 v1.1 (McCann) – p. 17/24

## “Vacuuming” a Database (3 / 3)

We can command Oracle 11 to collect new stats:

```
exec DBMS_STATS.GATHER_SCHEMA_STATS('mccann');  
select count(*) from user_histograms; ==> 180
```

And try again:

Plan hash value: 3135777751

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	104	8 (25)	00:00:01
* 1	HASH JOIN		4	104	8 (25)	00:00:01
2	MERGE JOIN		4	56	4 (25)	00:00:01
* 3	TABLE ACCESS BY INDEX ROWID	P	1	8	2 (0)	00:00:01
4	INDEX FULL SCAN	SYS_C0011137	6		1 (0)	00:00:01
* 5	SORT JOIN		24	144	2 (50)	00:00:01
6	INDEX FULL SCAN	SYS_C0011150	24	144	1 (0)	00:00:01
* 7	TABLE ACCESS FULL	S	5	60	3 (0)	00:00:01

Back to the original plan! (Same plan hash value.)

Query Optimization – CSC 460 v1.1 (McCann) – p. 18/24

# Influencing the Query Optimizer (1 / 6)

---

One way to nudge the DBMS: Rewrite your queries.

Some options that influence some DBMSes:

Query Optimization – CSc 460 v1.1 (McCann) – p. 19/24

# Influencing the Query Optimizer (2 / 6)

---

Oracle has a different scheme: HINTS

Format: `select /*+ <hint> */ ...`

Available hints include:

- RULE → Use RBO instead of CBO
- ALL\_ROWS → Favor turnaround time (final response)
- FIRST\_ROWS (n) → Favor first response time for first n tuples
- ORDERED → Honor query's join order
- USE\_NL (<table(s)>) → Use NLJ w/ <table(s)>
- USE\_MERGE (<table(s)>) → Use SMJ ...
- USE\_HASH (<table(s)>) → Use Hash Join ...

Query Optimization – CSc 460 v1.1 (McCann) – p. 20/24

# 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';
```

Plan hash value: 1885933196

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	NESTED LOOPS	
3	NESTED LOOPS	
4	TABLE ACCESS FULL	SPJ
* 5	TABLE ACCESS BY INDEX ROWID	P
* 6	INDEX UNIQUE SCAN	SYS_C0011137
* 7	INDEX UNIQUE SCAN	SYS_C0011142
* 8	TABLE ACCESS BY INDEX ROWID	S

Note

- rule based optimizer used (consider using cbo)

Query Optimization – CSc 460 v1.1 (McCann) – p. 21/24

# 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';
```

Plan hash value: 3135777751

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	104	8 (25)	00:00:01
* 1	HASH JOIN		4	104	8 (25)	00:00:01
2	MERGE JOIN		4	56	4 (25)	00:00:01
* 3	TABLE ACCESS BY INDEX ROWID	P	1	8	2 (0)	00:00:01
4	INDEX FULL SCAN	SYS_C0011137	6		1 (0)	00:00:01
* 5	SORT JOIN		24	144	2 (50)	00:00:01
6	INDEX FULL SCAN	SYS_C0011150	24	144	1 (0)	00:00:01
* 7	TABLE ACCESS FULL	S	5	60	3 (0)	00:00:01

Query Optimization – CSc 460 v1.1 (McCann) – p. 22/24

# 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';
```

Plan hash value: 2513296039

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	104	8 (0)	00:00:01
1	NESTED LOOPS					
2	NESTED LOOPS		4	104	8 (0)	00:00:01
3	NESTED LOOPS		4	56	4 (0)	00:00:01
* 4	TABLE ACCESS FULL	P	1	8	3 (0)	00:00:01
* 5	INDEX FULL SCAN	SYS_C0011150	4	24	1 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	SYS_C0011142	1		0 (0)	00:00:01
* 7	TABLE ACCESS BY INDEX ROWID	S	1	12	1 (0)	00:00:01

Query Optimization – CSc 460 v1.1 (McCann) – p. 23/24

# 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';
```

Plan hash value: 2745782731

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		4	104	8 (13)	00:00:01
* 1	HASH JOIN		4	104	8 (13)	00:00:01
2	MERGE JOIN CARTESIAN		5	100	6 (0)	00:00:01
* 3	TABLE ACCESS FULL	P	1	8	3 (0)	00:00:01
4	BUFFER SORT		5	60	3 (0)	00:00:01
* 5	TABLE ACCESS FULL	S	5	60	3 (0)	00:00:01
6	INDEX FULL SCAN	SYS_C0011150	24	144	1 (0)	00:00:01

Query Optimization – CSc 460 v1.1 (McCann) – p. 24/24