

Executing Explain Plans
and
Explaining Execution Plans

Craig Martin

- Applications DBA for Nationwide in Columbus, OH
- Worked with SQL and PL/SQL on Oracle since 2002
- Just like all of you!

Nationwide Disclaimer

- The opinions I share are mine, and do not necessarily reflect those of Nationwide
- It is up to each individual to evaluate the content in this presentation and do their own testing
- Use this information at your own risk

Why?



Gene Kranz



John Wooden

Why?



“Work the problem people. Don’t make things worse by guessing” - Gene Kranz

“If you don't have time to do it right, when will you have time to do it over?” - John Wooden

Explain Plan - Example

```
1  create table t1
2  as
3  select
4      rownum rn,
5      dbms_random.string('l',10) rand_string,
6      mod(rownum, 50) + 1 t2_id,
7      ceil(rownum/1000) t3_id
8  from dual
9  connect by level <= 100000;
10
11 create table t2
12 as
13 select
14     rownum t2_id,
15     dbms_random.string('l',10) rand_string,
16     round(dbms_random.value(1, 75)) t4_id
17 from dual
18 connect by level <= 50;
19
20 create table t3
21 as
22 select
23     rownum t3_id,
24     dbms_random.string('l',10) rand_string,
25     round(dbms_random.value(1, 75)) t4_id
26 from dual
27 connect by level <= 100;
28
29 create table t4
30 as
31 select
32     rownum t4_id,
33     dbms_random.string('l',10) rand_string
34 from dual
35 connect by level <= 75;
36
```

```
38 ► explain plan for
39     select *
40     from t1
41         inner join t2
42             on t2.t2_id = t1.t2_id
43         inner join t3
44             on t3.t3_id = t1.t3_id
45         inner join t4
46             on t4.t4_id = t2.t4_id
47     where
48         t4.rand_string like 'f%';
49
```

Displaying Results

- Query PLAN_TABLE Directly
- Explain Plan feature in SQL Developer 
- DBMS_XPLAN package

DBMS_XPLAN.DISPLAY

```
select *  
from table(dbms_xplan.display());
```

- Pros
 - Actual SQL doesn't have to be executed just explained
- Cons
 - The plan produced may not be the actual plan used when executing the SQL

DBMS_XPLAN.DISPLAY_CURSOR

```
select *  
from table(dbms_xplan.display_cursor());
```

- Pros
 - Can display plan information for any SQL in cursor cache
 - Allows displaying of additional statistics (I/O, memory, timing)
- Cons
 - SQL must have already been executed

DBMS_XPLAN.DISPLAY_CURSOR

- Parameters
 - SQL_ID
 - Child Number
 - Format
- Gathering Additional Details
 - Memory Management Statistics
 - Set parameter *pga_aggregate_target* to non-zero
 - I/O Statistics
 - Set parameter *statistics_level* to “ALL”
 - Use *gather_plan_statistics* hint during execution

DISPLAY_CURSOR: Output

- Up to 7 sections possible
 - Information about SQL Statement
 - Execution Plan
 - Query Blocks
 - Outline
 - Predicates
 - Column Projection
 - Notes

DISPLAY_CURSOR: Output

Information about SQL

```
SQL_ID 4p77kv72anazz, child number 0
```

```
-----  
select /*+ gather_plan_statistics */ * from t1 inner join t2 on t2.t2_id = t1.t2_id inner join t3  
on t3.t3_id = t1.t3_id inner join t4 on t4.t4_id = t2.t4_id where t4.rand_string like 'f%'
```

DISPLAY_CURSOR: Output

Execution Plan

Plan hash value: 1012227572

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	A-Rows	A-Time	Buffers	OMem	lMem	O/1/M
* 1	HASH JOIN		1	5407	41M	249 (43)	00:00:01	500	00:00:00.02	105	1023K	1023K	1/0/0
2	TABLE ACCESS FULL	T3	1	100	198K	3 (0)	00:00:01	100	00:00:00.01	3			
* 3	HASH JOIN		1	5407	31M	244 (44)	00:00:01	500	00:00:00.01	102	899K	899K	1/0/0
* 4	HASH JOIN		1	3	12129	7 (15)	00:00:01	1	00:00:00.01	6	1078K	1078K	1/0/0
* 5	TABLE ACCESS FULL	T4	1	2	4030	3 (0)	00:00:01	2	00:00:00.01	3			
6	TABLE ACCESS FULL	T2	1	50	99K	3 (0)	00:00:01	50	00:00:00.01	3			
7	TABLE ACCESS FULL	T1	1	97326	189M	206 (36)	00:00:01	24988	00:00:00.01	96			

DISPLAY_CURSOR: Output

Query Blocks

```
Query Block Name / Object Alias (identified by operation id):
```

```
-----  
1 - SEL$EE94F965  
2 - SEL$EE94F965 / T3@SEL$2  
5 - SEL$EE94F965 / T4@SEL$3  
6 - SEL$EE94F965 / T2@SEL$1  
7 - SEL$EE94F965 / T1@SEL$1
```

DISPLAY_CURSOR: Output

Outline

```
Outline Data
-----

/*+
  BEGIN_OUTLINE_DATA
  IGNORE_OPTIM_EMBEDDED_HINTS
  OPTIMIZER_FEATURES_ENABLE('10.2.0.4')
  OPT_PARAM('_optimizer_max_permutations' 100)
  OPT_PARAM('_b_tree_bitmap_plans' 'false')
  OPT_PARAM('optimizer_index_cost_adj' 10)
  OPT_PARAM('optimizer_index_caching' 10)
  ALL_ROWS
  OUTLINE_LEAF(@"SEL$EE94F965")
  MERGE(@"SEL$9E43CB6E")
  OUTLINE(@"SEL$4")
  OUTLINE(@"SEL$9E43CB6E")
  MERGE(@"SEL$58A6D7F6")
  OUTLINE(@"SEL$3")
  OUTLINE(@"SEL$58A6D7F6")
  MERGE(@"SEL$1")
  OUTLINE(@"SEL$2")
  OUTLINE(@"SEL$1")
  FULL(@"SEL$EE94F965" "T4"@"SEL$3")
  FULL(@"SEL$EE94F965" "T2"@"SEL$1")
  FULL(@"SEL$EE94F965" "T1"@"SEL$1")
  FULL(@"SEL$EE94F965" "T3"@"SEL$2")
  LEADING(@"SEL$EE94F965" "T4"@"SEL$3" "T2"@"SEL$1" "T1"@"SEL$1"
    "T3"@"SEL$2")
  USE_HASH(@"SEL$EE94F965" "T2"@"SEL$1")
  USE_HASH(@"SEL$EE94F965" "T1"@"SEL$1")
  USE_HASH(@"SEL$EE94F965" "T3"@"SEL$2")
  SWAP_JOIN_INPUTS(@"SEL$EE94F965" "T3"@"SEL$2")
  END_OUTLINE_DATA
*/
```

DISPLAY_CURSOR: Output

Predicates

```
Predicate Information (identified by operation id):
```

```
-----  
1 - access("T3"."T3_ID"="T1"."T3_ID")  
3 - access("T2"."T2_ID"="T1"."T2_ID")  
4 - access("T4"."T4_ID"="T2"."T4_ID")  
5 - filter("T4"."RAND_STRING" LIKE 'f%')
```

- Access = Only matching rows are retrieved
- Filter = All rows are retrieved, matching rows kept

DISPLAY_CURSOR: Output

Column Projection

Column Projection Information (identified by operation id):

```
-----  
1 - (#keys=1) "T3"."T3_ID"[NUMBER,22], "T1"."T3_ID"[NUMBER,22], "T3"."T4_ID"[NUMBER,22], "T3"."RAND_STRING"[VARCHAR2,4000],  
   "T2"."T2_ID"[NUMBER,22], "T1"."T2_ID"[NUMBER,22], "T4"."T4_ID"[NUMBER,22], "T2"."T4_ID"[NUMBER,22], "T4"."RAND_STRING"[VARCHAR2,4000],  
   "T2"."RAND_STRING"[VARCHAR2,4000], "T1"."T1_ID"[NUMBER,22], "T1"."RAND_STRING"[VARCHAR2,4000]  
2 - "T3"."T3_ID"[NUMBER,22], "T3"."RAND_STRING"[VARCHAR2,4000], "T3"."T4_ID"[NUMBER,22]  
3 - (#keys=1) "T2"."T2_ID"[NUMBER,22], "T1"."T2_ID"[NUMBER,22], "T4"."T4_ID"[NUMBER,22], "T2"."T4_ID"[NUMBER,22],  
   "T4"."RAND_STRING"[VARCHAR2,4000], "T2"."RAND_STRING"[VARCHAR2,4000], "T1"."T1_ID"[NUMBER,22], "T1"."RAND_STRING"[VARCHAR2,4000],  
   "T1"."T3_ID"[NUMBER,22]  
4 - (#keys=1) "T4"."T4_ID"[NUMBER,22], "T2"."T4_ID"[NUMBER,22], "T4"."RAND_STRING"[VARCHAR2,4000], "T2"."T2_ID"[NUMBER,22],  
   "T2"."RAND_STRING"[VARCHAR2,4000]  
5 - "T4"."T4_ID"[NUMBER,22], "T4"."RAND_STRING"[VARCHAR2,4000]  
6 - "T2"."T2_ID"[NUMBER,22], "T2"."RAND_STRING"[VARCHAR2,4000], "T2"."T4_ID"[NUMBER,22]  
7 - "T1"."T1_ID"[NUMBER,22], "T1"."RAND_STRING"[VARCHAR2,4000], "T1"."T2_ID"[NUMBER,22], "T1"."T3_ID"[NUMBER,22]
```

DISPLAY_CURSOR: Output

Notes

Note

- dynamic sampling used for this statement

DISPLAY_CURSOR: Output

Plan hash value: 1012227572

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		5407	41M	249 (43)	00:00:01
* 1	HASH JOIN		5407	41M	249 (43)	00:00:01
2	TABLE ACCESS FULL	T3	100	198K	3 (0)	00:00:01
* 3	HASH JOIN		5407	31M	244 (44)	00:00:01
* 4	HASH JOIN		3	12129	7 (15)	00:00:01
* 5	TABLE ACCESS FULL	T4	2	4030	3 (0)	00:00:01
6	TABLE ACCESS FULL	T2	50	99K	3 (0)	00:00:01
7	TABLE ACCESS FULL	T1	97326	189M	206 (36)	00:00:01

Plan hash value: 1012227572

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	A-Rows	A-Time	Buffers	OMem	lMem	O/1/M
* 1	HASH JOIN		1	5407	41M	249 (43)	00:00:01	500	00:00:00.02	105	1023K	1023K	1/0/0
2	TABLE ACCESS FULL	T3	1	100	198K	3 (0)	00:00:01	100	00:00:00.01	3			
* 3	HASH JOIN		1	5407	31M	244 (44)	00:00:01	500	00:00:00.01	102	899K	899K	1/0/0
* 4	HASH JOIN		1	3	12129	7 (15)	00:00:01	1	00:00:00.01	6	1078K	1078K	1/0/0
* 5	TABLE ACCESS FULL	T4	1	2	4030	3 (0)	00:00:01	2	00:00:00.01	3			
6	TABLE ACCESS FULL	T2	1	50	99K	3 (0)	00:00:01	50	00:00:00.01	3			
7	TABLE ACCESS FULL	T1	1	97326	189M	206 (36)	00:00:01	24988	00:00:00.01	96			

DISPLAY_CURSOR: Output

56
57
58
59

```
select *  
from table(dbms_xplan.display_cursor('4p77kv72anazz',0,'IOSTATS'));
```

SQL_ID 4p77kv72anazz, child number 0

```
-----  
select /*+ gather_plan_statistics */ * from t1 inner join t2 on t2.t2_id =  
t1.t2_id inner join t3 on t3.t3_id = t1.t3_id inner join t4 on t4.t4_id  
= t2.t4_id where t4.rand_string like 'f%'
```

Plan hash value: 1012227572

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
* 1	HASH JOIN		1	5407	500	00:00:00.02	105
2	TABLE ACCESS FULL	T3	1	100	100	00:00:00.01	3
* 3	HASH JOIN		1	5407	500	00:00:00.01	102
* 4	HASH JOIN		1	3	1	00:00:00.01	6
* 5	TABLE ACCESS FULL	T4	1	2	2	00:00:00.01	3
6	TABLE ACCESS FULL	T2	1	50	50	00:00:00.01	3
7	TABLE ACCESS FULL	T1	1	97326	24988	00:00:00.01	96

Predicate Information (identified by operation id):

```
-----  
1 - access("T3"."T3_ID"="T1"."T3_ID")  
3 - access("T2"."T2_ID"="T1"."T2_ID")  
4 - access("T4"."T4_ID"="T2"."T4_ID")  
5 - filter("T4"."RAND_STRING" LIKE 'f%')
```

Note

```
-----  
- dynamic sampling used for this statement
```

Reading Execution Plans

- The only operation without a parent is the root node
- A parent can have one or many children
- A child can only have one parent
- Children are displayed indented to the right of their parent
- All children of a single parent have the same indentation
- A parent is displayed before its children
- The ID of a parent is less than the IDs of its children
- If there are several nodes with the same indentation as the parent, the node closest to the child is the parent

Reading Execution Plans

• From Oracle Documentation (11.2)

- http://download.oracle.com/docs/cd/E11882_01/server.112/e16638/optimops.htm#i82029

11.4.1 Overview of EXPLAIN PLAN

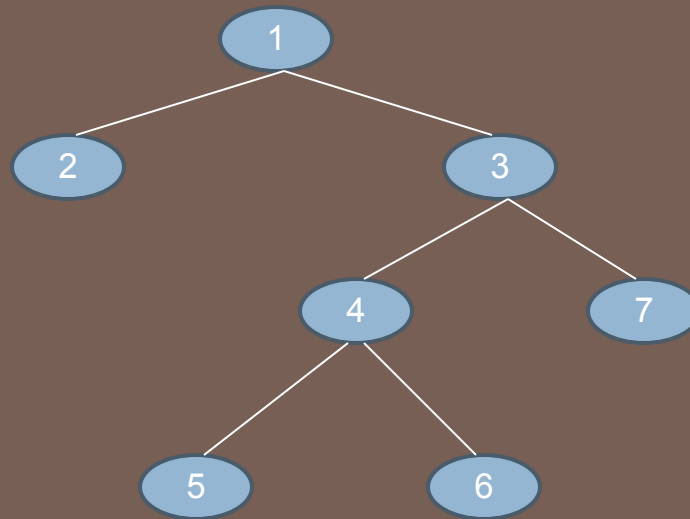
You can examine the execution plan chosen by the optimizer for a SQL statement by using the `EXPLAIN PLAN` statement. When the statement is issued, the optimizer chooses an execution plan and then inserts data describing the plan into a database table. Simply issue the `EXPLAIN PLAN` statement and then query the output table.

These are the basics of using the `EXPLAIN PLAN` statement:

- Use the SQL script `CATPLAN.SQL` to create a sample output table called `PLAN_TABLE` in your schema. See ["The PLAN TABLE Output Table"](#).
- Include the `EXPLAIN PLAN FOR` clause before the SQL statement. See ["Running EXPLAIN PLAN"](#).
- After issuing the `EXPLAIN PLAN` statement, use one of the scripts or package provided by Oracle Database to display the most recent plan table output. See ["Displaying PLAN TABLE Output"](#).
- The execution order in `EXPLAIN PLAN` output begins with the line that is the furthest indented to the right. The next step is the parent of that line. If two lines are indented equally, then the top line is normally executed first.



Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
* 1	HASH JOIN		1	5407	500	00:00:00.02	105
2	TABLE ACCESS FULL	T3	1	100	100	00:00:00.01	3
* 3	HASH JOIN		1	5407	500	00:00:00.01	102
* 4	HASH JOIN		1	3	1	00:00:00.01	6
* 5	TABLE ACCESS FULL	T4	1	2	2	00:00:00.01	3
6	TABLE ACCESS FULL	T2	1	50	50	00:00:00.01	3
7	TABLE ACCESS FULL	T1	1	97326	24988	00:00:00.01	96



Common Operations

- Stand-Alone Operations
 - Most operations of this type
 - Parents have at most one child
 - Child executed at most once
 - Child feeds its parent
 - Examples:
 - TABLE ACCESS
 - HASH GROUP BY
 - COUNT

Common Operations

- Unrelated-Combine Operations
 - Parents have multiple children that are executed independently
 - Children are executed sequentially
 - Starts with child with smallest ID
 - Every child executed at most once
 - Every child feeds its parent
 - Examples:
 - HASH JOIN
 - MERGE JOIN
 - UNION-ALL

Common Operations

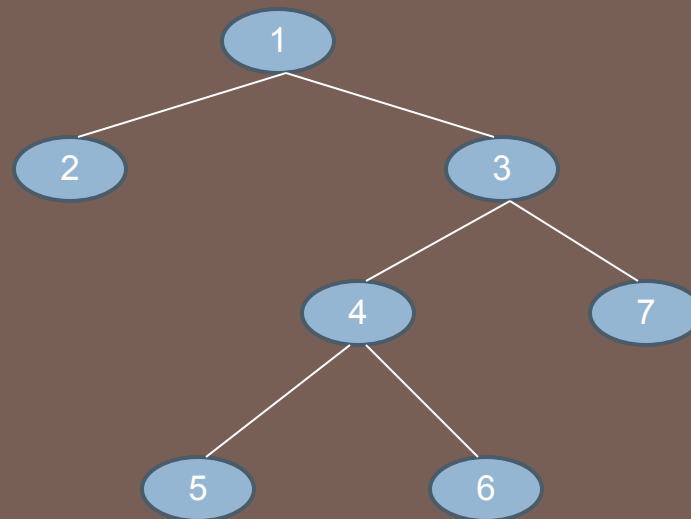
- Related-Combine Operations
 - Parents have multiple children where one child controls execution of the other children
 - Children are not executed sequentially
 - Only first child executed at most once
 - All others may be executed many times or not at all
 - Not every child feeds its parent
 - Some are used only as restrictions
 - Examples:
 - NESTED LOOPS
 - UPDATE
 - FILTER

Plan hash value: 1997971088

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
* 1	HASH JOIN		1	5407	500	00:00:01.41	150K
* 2	TABLE ACCESS FULL	T4	1	2	2	00:00:00.01	3
3	NESTED LOOPS		1	97326	24988	00:00:01.50	150K
4	NESTED LOOPS		1	97326	24988	00:00:00.67	75060
5	TABLE ACCESS FULL	T1	1	97326	24988	00:00:00.03	96
* 6	TABLE ACCESS FULL	T2	24988	1	24988	00:00:00.63	74964
* 7	TABLE ACCESS FULL	T3	24988	1	24988	00:00:00.80	74964

Predicate Information (identified by operation id):

- 1 - access("T4"."T4_ID"="T2"."T4_ID")
- 2 - filter("T4"."RAND_STRING" LIKE 'f%')
- 6 - filter("T2"."T2_ID"="T1"."T2_ID")
- 7 - filter("T3"."T3_ID"="T1"."T3_ID")



Plan hash value: 2553113010

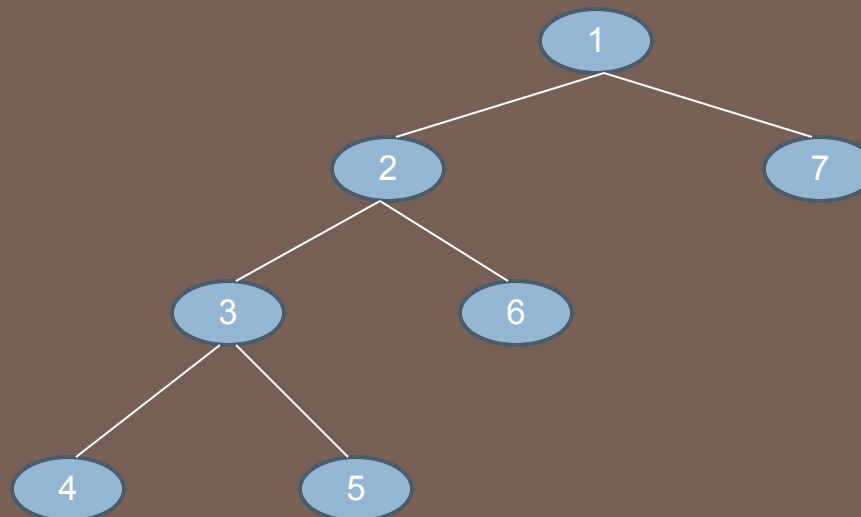
Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
1	NESTED LOOPS		1	5407	500	00:00:02.32	224K
2	NESTED LOOPS		1	97326	24988	00:00:01.47	150K
3	NESTED LOOPS		1	97326	24988	00:00:00.65	75060
4	TABLE ACCESS FULL	T1	1	97326	24988	00:00:00.03	96
* 5	TABLE ACCESS FULL	T2	24988	1	24988	00:00:00.61	74964
* 6	TABLE ACCESS FULL	T3	24988	1	24988	00:00:00.80	74964
* 7	TABLE ACCESS FULL	T4	24988	1	500	00:00:00.88	74964

Predicate Information (identified by operation id):

5 - filter("T2"."T2_ID"="T1"."T2_ID")

6 - filter("T3"."T3_ID"="T1"."T3_ID")

7 - filter(("T4"."RAND_STRING" LIKE 'f%' AND "T4"."T4_ID"="T2"."T4_ID"))



Plan hash value: 1189412883

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers
* 1	HASH JOIN		1	2595	500	00:00:00.88	14393
* 2	TABLE ACCESS FULL	T4	1	2	2	00:00:00.01	3
3	VIEW		1	97327	76500	00:00:00.92	14390
4	UNION-ALL		1		76500	00:00:00.92	14390
* 5	HASH JOIN RIGHT OUTER		1	97326	76500	00:00:00.84	14390
6	TABLE ACCESS FULL	T3	1	100	100	00:00:00.01	3
7	NESTED LOOPS		1	97326	76500	00:00:00.77	14387
8	TABLE ACCESS FULL	T2	1	50	39	00:00:00.01	3
* 9	TABLE ACCESS FULL	T1	39	1947	76500	00:00:00.69	14384
* 10	HASH JOIN ANTI		0	1	0	00:00:00.01	0
11	TABLE ACCESS FULL	T3	0	100	0	00:00:00.01	0
12	VIEW	VW_SQ_1	0	97326	0	00:00:00.01	0
* 13	HASH JOIN		0	97326	0	00:00:00.01	0
14	TABLE ACCESS FULL	T2	0	50	0	00:00:00.01	0
15	TABLE ACCESS FULL	T1	0	97326	0	00:00:00.01	0

Predicate Information (identified by operation id):

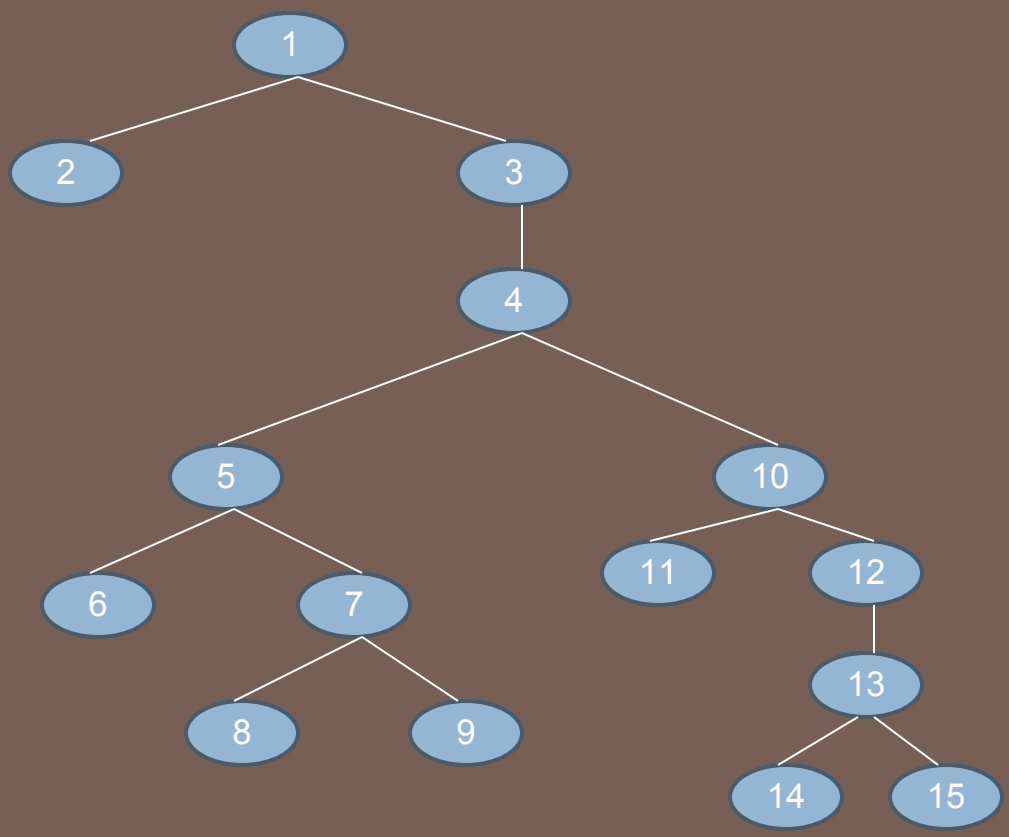
- 1 - access("T4"."T4_ID"="T2"."T4_ID")
- 2 - filter("T4"."RAND_STRING" LIKE 'f%')
- 5 - access("T3"."T3_ID"="T1"."T3_ID")
- 9 - filter("T2"."T2_ID"="T1"."T2_ID")
- 10 - access("T3"."T3_ID"="T3_ID")
- 13 - access("T2"."T2_ID"="T1"."T2_ID")

Plan hash value: 1189412883

Id	Operation	Name	Starts
* 1	HASH JOIN		1
* 2	TABLE ACCESS FULL	T4	1
3	VIEW		1
4	UNION-ALL		1
* 5	HASH JOIN RIGHT OUTER		1
6	TABLE ACCESS FULL	T3	1
7	NESTED LOOPS		1
8	TABLE ACCESS FULL	T2	1
* 9	TABLE ACCESS FULL	T1	39
* 10	HASH JOIN ANTI		0
11	TABLE ACCESS FULL	T3	0
12	VIEW	VW_SQ_1	0
* 13	HASH JOIN		0
14	TABLE ACCESS FULL	T2	0
15	TABLE ACCESS FULL	T1	0

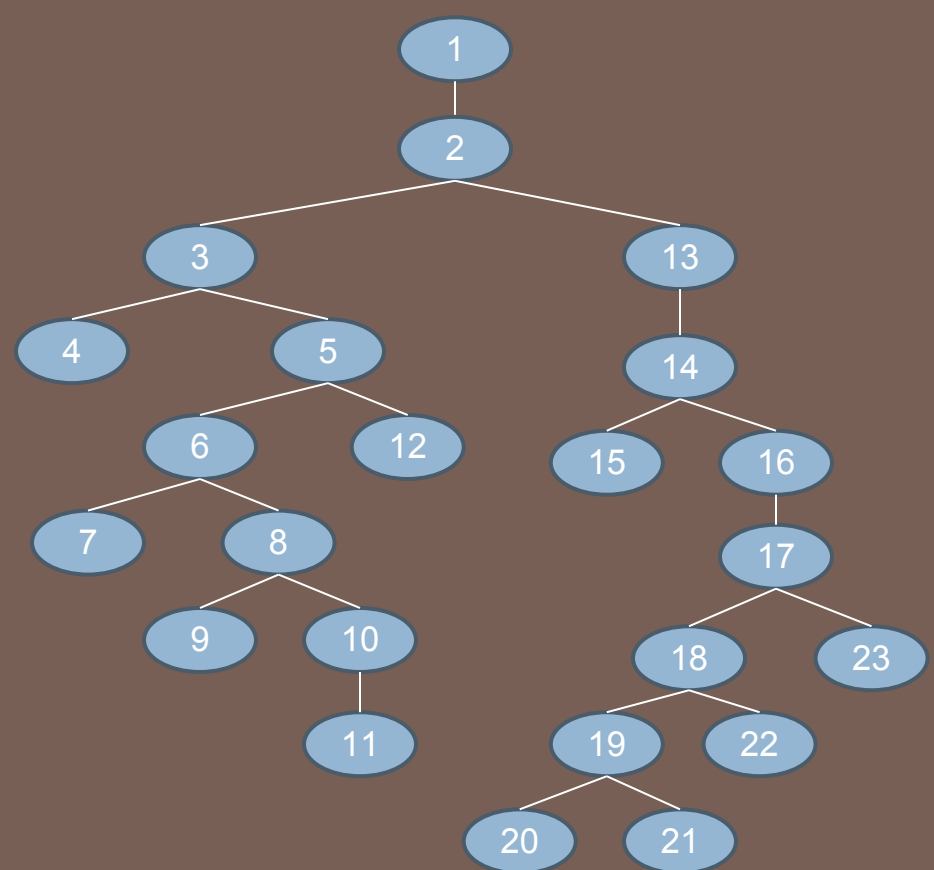
Predicate Information (identified by operation id):

- 1 - access("T4"."T4_ID"="T2"."T4_ID")
- 2 - filter("T4"."RAND_STRING" LIKE 'f%')
- 5 - access("T3"."T3_ID"="T1"."T3_ID")
- 9 - filter("T2"."T2_ID"="T1"."T2_ID")
- 10 - access("T3"."T3_ID"="T3_ID")
- 13 - access("T2"."T2_ID"="T1"."T2_ID")



Plan hash value: 3474588145

Id	Operation	Name	Starts
1	VIEW		1
2	UNION-ALL		1
* 3	HASH JOIN RIGHT OUTER		1
4	TABLE ACCESS FULL	T3	1
* 5	HASH JOIN		1
* 6	HASH JOIN		1
7	TABLE ACCESS FULL	T2	1
8	MERGE JOIN CARTESIAN		1
* 9	TABLE ACCESS FULL	T4	1
10	BUFFER SORT		1
11	TABLE ACCESS FULL	T3	1
12	TABLE ACCESS FULL	T1	0
* 13	FILTER		1
* 14	HASH JOIN ANTI		0
15	TABLE ACCESS FULL	T3	0
16	VIEW	VW_SQ_1	0
* 17	HASH JOIN		0
* 18	HASH JOIN		0
* 19	HASH JOIN OUTER		0
20	TABLE ACCESS FULL	T2	0
21	TABLE ACCESS FULL	T4	0
22	TABLE ACCESS FULL	T3	0
23	TABLE ACCESS FULL	T1	0



What to Ignore



- Cost!
 - Can't be directly controlled
 - May not be an accurate indication of performance
- Rules of Always
 - Full table scans are always bad
 - Indexes are always good

What to Look For

- Estimates that don't match the actual data
 - Inaccurate statistics may exist
- Wasted Operations / Rows
 - Many more rows are read for an operation than are used
 - Full scans
 - Unselective range scans
 - Wrong join order
 - Late filter operations
- High Execution Time / Buffer Count

References

- Oracle Documentation
 - Oracle Database Performance Tuning Guide
 - Oracle Database PL/SQL Packages and Types Reference
- Troubleshooting Oracle Performance by Christian Antognini

Questions?



<http://www.cmartin2.com>
cmartin2@cmartin2.com