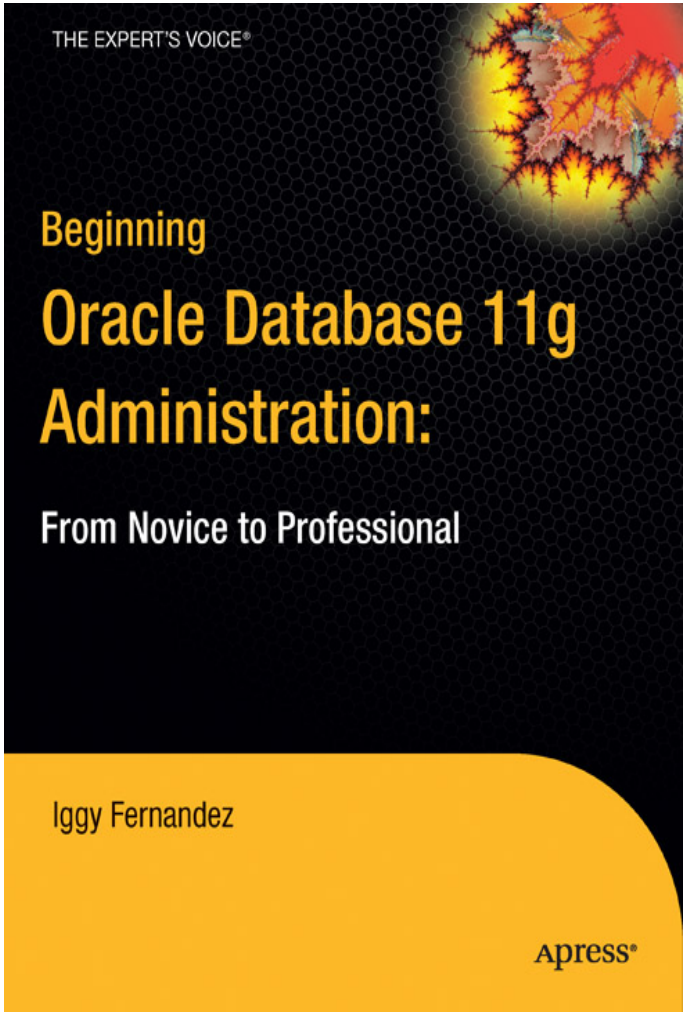
COLLABORATE 10
Technology and Applications Forum for the Oracle Community

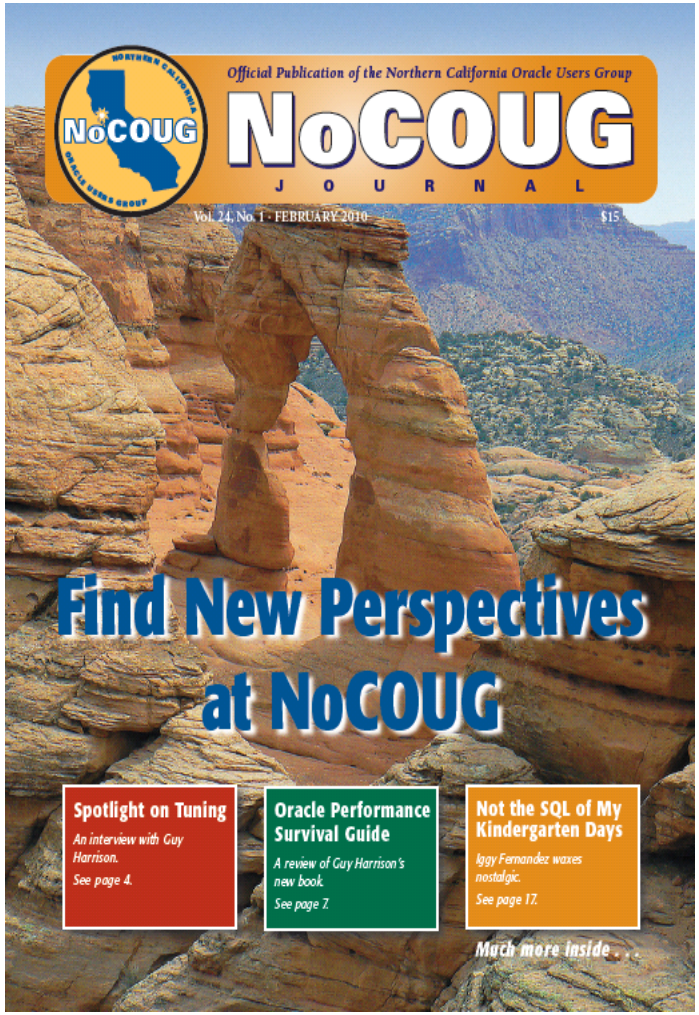
IOUG
independent oracle users group

INTERPRETING QUERY EXECUTION PLANS

Iggy Fernandez
Database Specialists
Session #430

DatabaseSpecialists


THE EXPERT'S VOICE®
Beginning
Oracle Database 11g
Administration:
From Novice to Professional
Iggy Fernandez
Apress®


Official Publication of the Northern California Oracle Users Group
NoCOUG
JOURNAL
Vol. 24, No. 1 · FEBRUARY 2010
$15
Find New Perspectives at NoCOUG
Spotlight on Tuning
An interview with Guy Harrison.
See page 4.
Oracle Performance Survival Guide
A review of Guy Harrison's new book.
See page 7.
Not the SQL of My Kindergarten Days
Iggy Fernandez waxes nostalgic.
See page 17.
Much more inside . . .

SQL QUIZ #1

The execution plans in the test and production databases are identical.

True or False?

SQL QUIZ #2

Explain Plan displays the execution plan used to execute the query.

True or False?

SQL QUIZ #3

Autotrace displays the execution plan used to execute the query.

True or False?

SQL QUIZ #4

The execution plan does not change.

True or False?

SQL QUIZ #5

The cost-based optimizer selects the optimal execution plan—the one with the lowest cost.

True or False?

SQL QUIZ #6

There is an optimal execution plan—one with the lowest cost.

True or False?

Three Tips For Improving Readability

1. Format queries using Toad or SQL Developer
2. Qualify column names with the table names
3. Use Common Table Expressions
4. Use ANSI join syntax

Inline Views

```
SELECT *
FROM    (SELECT *
         FROM    Suppliers
         MINUS
         SELECT *
         FROM    (SELECT SupplierName
                  FROM    (SELECT *
                           FROM    (SELECT *
                                    FROM    Suppliers
                                    Parts)
                           MINUS
                           SELECT *
                           FROM    (SELECT SupplierName,
                                    PartName
                                    FROM    Quotes))));
```


DatabaseSpecialists

Common Table Expressions
All Supplier Part Pairs

```
WITH

AllSupplierPartPairs AS
(
  SELECT *
  FROM Suppliers CROSS JOIN Parts
),
```

Common Table Expressions
Valid Supplier Part Pairs

```
ValidSupplierPartPairs AS
(
  SELECT SupplierName, PartName
  FROM Quotes
),
```

Common Table Expressions
Invalid Supplier Part Pairs

```
InvalidSupplierPartPairs AS
(
  SELECT *
  FROM AllSupplierPartPairs
  MINUS
  SELECT *
  FROM ValidSupplierPartPairs
),
```

Common Table Expressions
Suppliers Who Don't Supply All Parts

```
SuppliersWhoDontSupplyAllParts AS
(
  SELECT SupplierName
  FROM InvalidSupplierPartPairs
),
```

Common Table Expressions Suppliers Who Supply All Parts

```
SuppliersWhoSupplyAllParts AS
(
  SELECT *
  FROM Suppliers
  MINUS
  SELECT *
  FROM SuppliersWhoDontSupplyAllParts
)
```

Common Table Expressions
Suppliers Who Supply All Parts

```
SELECT *
FROM SuppliersWhoSupplyAllParts;
```

EXPLAIN PLAN lies!

```
SELECT
  cardinality,
  cost,
  lpad(' ',level-1)||operation||' '|| options||' '||object_name AS operation
FROM plan_table
CONNECT BY prior id = parent_id AND prior statement_id = statement_id
START WITH id = 0 AND statement_id = 'TEST'
ORDER BY id;

CARDINALITY       COST OPERATION
----------- ---------- ----------------------------------------
          1          6 SELECT STATEMENT
          1          6  HASH JOIN
         10          2   TABLE ACCESS BY INDEX ROWID EMPLOYEES
         10          1    INDEX RANGE SCAN EMP_DEPARTMENT_IX
        107          3   TABLE ACCESS FULL EMPLOYEES
```

AUTOTRACE lies!

```
Execution Plan
----------------------------------------------------------
Plan hash value: 2254211361

-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                    |     1 |    23 |     6  (17)| 00:00:01 |
|*  1 |  HASH JOIN                    |                    |     1 |    23 |     6  (17)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMPLOYEES          |    10 |   150 |     2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | EMP_DEPARTMENT_IX |    10 |       |     1   (0)| 00:00:01 |
|   4 |   TABLE ACCESS FULL           | EMPLOYEES          |   107 |   856 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

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

   1 - access("E1"."MANAGER_ID"="E2"."EMPLOYEE_ID")
       filter("E1"."SALARY">"E2"."SALARY")
   3 - access("E1"."DEPARTMENT_ID"=TO_NUMBER(:DEPARTMENT_ID))
```

TKPROF Tells Half The Story

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      0.00       0.00          0         14          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.00       0.00          0         14          0           0

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 180

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  HASH JOIN  (cr=14 pr=0 pw=0 time=1681 us)
     45   TABLE ACCESS FULL EMPLOYEES (cr=7 pr=0 pw=0 time=446 us)
    107   TABLE ACCESS FULL EMPLOYEES (cr=7 pr=0 pw=0 time=25 us)
```

DBMS_XPLAN Tells It All

```
-------------------------------------------------------------------------------------------
| Id  | Operation           | Name      | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   |
-------------------------------------------------------------------------------------------
|*  1 |  HASH JOIN          |           |      1 |      2 |    46 |     7  (15)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| EMPLOYEES |      1 |     45 |   675 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| EMPLOYEES |      1 |    107 |   856 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

---------------------------------------------------------------------
| Id  | Operation           | Name      | A-Rows |   A-Time    | Buffers |
---------------------------------------------------------------------
|*  1 |  HASH JOIN          |           |      0 |00:00:00.01 |      14 |
|*  2 |   TABLE ACCESS FULL| EMPLOYEES |     45 |00:00:00.01 |       7 |
|   3 |   TABLE ACCESS FULL| EMPLOYEES |    107 |00:00:00.01 |       7 |
---------------------------------------------------------------------
Peeked Binds (identified by position):
1 - (NUMBER): 50
Predicate Information (identified by operation id):
1 - access("E1"."MANAGER_ID"="E2"."EMPLOYEE_ID")
       filter("E1"."SALARY">"E2"."SALARY")
   2 - filter("E1"."DEPARTMENT_ID"=:DEPARTMENT_ID)
```

How To Read Query Plans (From The Documentation)

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.

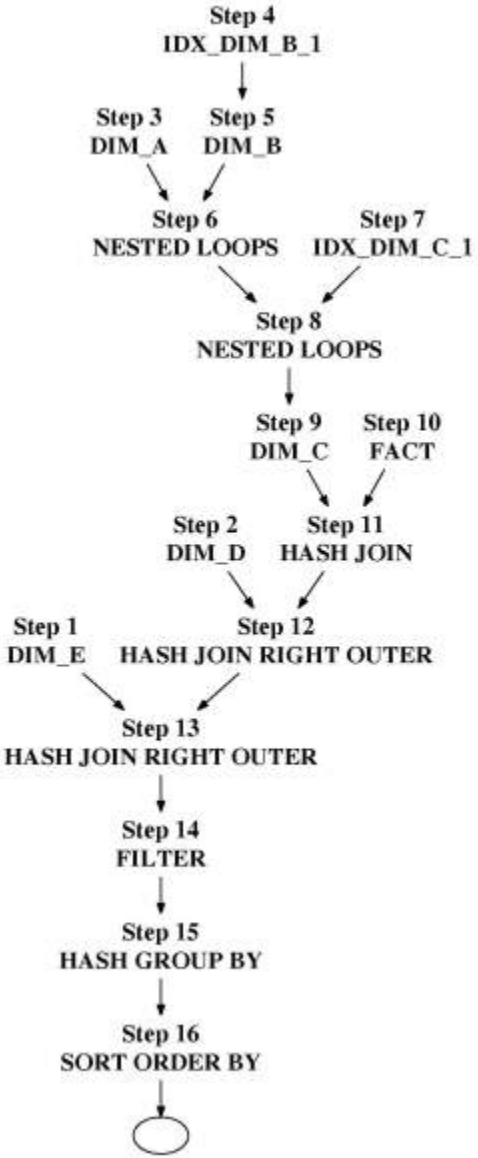
How To Read Query Plans (More Precisely)

Perform operations in the order in which they are listed except that if the operations listed after a certain operation are more deeply indented in the listing, then perform those operations first (in the order in which those operations are listed.)

Navigating The Maze

Id	Operation	Name	A-Time
1	SORT ORDER BY		00:00:30.33
2	HASH GROUP BY		00:00:30.30
* 3	FILTER		00:00:28.28
* 4	HASH JOIN RIGHT OUTER		00:00:27.12
5	TABLE ACCESS FULL	DIM_E	00:00:00.01
* 6	HASH JOIN RIGHT OUTER		00:00:23.63
7	TABLE ACCESS FULL	DIM_D	00:00:00.01
* 8	HASH JOIN		00:00:20.72
9	TABLE ACCESS BY INDEX ROWID	DIM_C	00:00:00.87
10	NESTED LOOPS		00:00:00.04
11	NESTED LOOPS		00:00:00.01
* 12	TABLE ACCESS FULL	DIM_A	00:00:00.01
* 13	TABLE ACCESS BY INDEX ROWID	DIM_B	00:00:00.01
* 14	INDEX RANGE SCAN	IDX_DIM_B_1	00:00:00.01
* 15	INDEX RANGE SCAN	IDX_DIM_C_1	00:00:00.02
* 16	TABLE ACCESS FULL	FACT	00:00:13.41

A Better Way


Step 4
IDX_DIM_B_1
Step 3
DIM_A
Step 5
DIM_B
Step 6
NESTED LOOPS
Step 7
IDX_DIM_C_1
Step 8
NESTED LOOPS
Step 9
DIM_C
Step 10
FACT
Step 2
DIM_D
Step 11
HASH JOIN
Step 1
DIM_E
Step 12
HASH JOIN RIGHT OUTER
Step 13
HASH JOIN RIGHT OUTER
Step 14
FILTER
Step 15
HASH GROUP BY
Step 16
SORT ORDER BY

Thanks For Listening

iggy_fernandez@hotmail.com
http://iggyfernandez.wordpress.com

Please submit evaluation forms

Session #430