Explain Plan : Execution Plan

Hemant K Chitale

SG RACSIG 12-Oct-11

Introduction

- The command “EXPLAIN PLAN” is an instruction to *present* the Execution Plan.
- The Execution Plan is normally determined at RunTime based on :
  - Presence of the same SQL, with Execution Plan, in the Shared Pool
  - Bind Peeking
  - Hard Parse (note : tkprof and autotrace EXPLAIN do Hard Parses)

Methods

- Simplest command :
  - EXPLAIN PLAN FOR select …. ;
  - SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY));
- Plan for existing SQL already parsed and executed
  - SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR('&sql_id','&child_number'));
- Plan from AWR
  - SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_AWR('&sql_id'));

Example -- 1

- EXPLAIN PLAN shows
  - Join Method
  - Expected Cardinality
  - Indexes
  - Access and Filter Predicates
  - Whether Dynamic Sampling is used

```
SQL> l
  1  explain plan for select c.cust_id, c.cust_name, f.prod_id, f.time_id, f.quantity_sold
  2  from sales_fact f, customer_dim c
  3  where f.cust_id = c.cust_id
  4* order by c.cust_id
SQL> /
Explained.
SQL> select * from table(dbms_xplan.display);
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------
Plan hash value: 3912614538
--------------------------------------------------------------------------------------------
| Id  | Operation           | Name         | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |              |   966K|    56M|       |  6858   (2)| 00:01:23 |
|   1 |  MERGE JOIN         |              |   966K|    56M|       |  6858   (2)| 00:01:23 |
|   2 |   SORT JOIN         |              |  7426 |   304K|   776K|   102   (1)| 00:00:02 |
|   3 |    TABLE ACCESS FULL| CUSTOMER_DIM |  7426 |   304K|       |    19   (0)| 00:00:01 |
|*  4 |   SORT JOIN         |              |   966K|    17M|    59M|  6756   (2)| 00:01:22 |
|   5 |    TABLE ACCESS FULL| SALES_FACT   |   966K|    17M|       |  1038   (2)| 00:00:13 |
--------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   4 - access("F"."CUST_ID"="C"."CUST_ID")
       filter("F"."CUST_ID"="C"."CUST_ID")
Note
-----
   - dynamic sampling used for this statement
22 rows selected.
SQL>
SQL> select count(*) from sales_fact;
  COUNT(*)
----------
    959040
SQL> select count(*) from customer_dim;
  COUNT(*)
----------
      7005
SQL>
```

Observations -- 1

- Can you identify which table / index it used Dynamic Sampling on ?
- Why are there two “SORT JOIN” and one “MERGE JOIN” operations ?

More Info:

From an Event 10053 trace

```
***************************************
BASE STATISTICAL INFORMATION
***********************
Table Stats::
  Table: CUSTOMER_DIM  Alias:  C  (NOT ANALYZED)
    #Rows: 6535  #Blks:  80  AvgRowLen:  100.00
  Column (#1): CUST_ID(NUMBER)  NO STATISTICS (using defaults)
    AvgLen: 22.00 NDV: 204 Nulls: 0 Density: 0.0048967
Index Stats::
  Index: CUST_ID_PK  Col#: 1
    LVLS: 1  #LB: 14  #DK: 7005  LB/K: 1.00  DB/K: 1.00  CLUF: 6332.00
***********************
Table Stats::
  Table: SALES_FACT  Alias:  F
    #Rows: 966158  #Blks:  4655  AvgRowLen:  29.00
  Column (#2): CUST_ID(NUMBER)
    AvgLen: 5.00 NDV: 4893 Nulls: 0 Density: 2.0437e-04 Min: 2 Max: 100970
Index Stats::
  Index: SALES_CUST_BMP_NDX  Col#: 2
    LVLS: 1  #LB: 358  #DK: 7005  LB/K: 1.00  DB/K: 1.00  CLUF: 7005.00
***************************************
```

There are statistics on the CUST_ID_PK index but not on the CUSTOMER_DIM table and CUST_ID column. (CREATE INDEX had included “COMPUTE STATISTICS”)

Example -- 2

- How does Explain Plan handle BINDs ?
- It doesn't ! Explain Plan does *not* peek binds.
- See http://tkyte.blogspot.com/2007/04/when-explanation-doesn-sound-quite.html
- ***Explain plan is blind to the bind***

```
SQL> create table Explain_Binds (owner varchar2(30) not null, object_name varchar2(30));
Table created.

SQL> insert into Explain_Binds select owner, object_name from dba_objects ;
46170 rows created.

SQL> create index Explain_Binds_NDX_1 on Explain_Binds(owner);
Index created.

SQL>
SQL> select owner, count(*) from dba_objects group by owner order by owner ;

OWNER                          COUNT(*)
------------------------------ ----------
DBSNMP                             46
EXFSYS                            281
HEMANT                            135
ORDPLUGINS                         10
ORDSYS                           1720
OUTLN                               8
PUBLIC                          18548
SH                                296
SI_INFORMTN_SCHEMA                  8
SYS                             22432
SYSMAN                           1321
SYSTEM                            449
TSMSYS                              3
WMSYS                             242
XDB                               672

15 rows selected.

SQL> exec dbms_stats.gather_table_stats('','EXPLAIN_BINDS',-
>                 estimate_percent=>100,method_opt=>'FOR COLUMNS OWNER SIZE 250');

PL/SQL procedure successfully completed.

SQL>
```

```
SQL> -- explain for Literal
SQL> explain plan for select object_name from explain_binds where owner = 'TSMSYS';
Explained.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
Plan hash value: 3312994892
-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |     3 |    69 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| EXPLAIN_BINDS       |     3 |    69 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | EXPLAIN_BINDS_NDX_1 |     3 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("OWNER"='TSMSYS')
14 rows selected.

SQL>

SQL> explain plan for select object_name from explain_binds where owner = 'SYS';
Explained.

SQL> select * from table(dbms_xplan.display);
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
Plan hash value: 2144353690

-------------------------------------------------------------------------------------
| Id  | Operation         | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |               | 22432 |   503K|    56   (2)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EXPLAIN_BINDS | 22432 |   503K|    56   (2)| 00:00:01 |
-------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("OWNER"='SYS')
13 rows selected.

SQL>
```

```
SQL> -- explain for Bind
SQL> variable  ow varchar2(30);
SQL> exec :ow := 'TSMSYS';
PL/SQL procedure successfully completed.

SQL> explain plan for select object_name from explain_binds where owner = :ow;
Explained.

SQL> select * from table(dbms_xplan.display);
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
Plan hash value: 3312994892

-----------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                     |  3078 | 70794 |    39   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| EXPLAIN_BINDS       |  3078 | 70794 |    39   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | EXPLAIN_BINDS_NDX_1 |  3078 |       |     8   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("OWNER"=:OW)
14 rows selected.

SQL>
SQL> exec :ow := 'SYS';
PL/SQL procedure successfully completed.

SQL> explain plan for select object_name from explain_binds where owner = :ow;
Explained.

SQL> select * from table(dbms_xplan.display);
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------
Plan hash value: 3312994892
-----------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                     |  3078 | 70794 |    39   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| EXPLAIN_BINDS       |  3078 | 70794 |    39   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | EXPLAIN_BINDS_NDX_1 |  3078 |       |     8   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("OWNER"=:OW)
14 rows selected.

SQL>
```

Observations -- 2

- Explain Plan with Bind assumed the same Cardinality for different Binds
- It did a Hard Parse without peeking the Bind

Other Methods -- 1

```
SQL> l
  1* select * from table(dbms_xplan.display_cursor('&sql_id'))
SQL> /
Enter value for sql_id: 1qgc21cjkb741

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
     --------------------------
SQL_ID  1qgc21cjkb741, child number 0
-------------------------------------
select c.cust_id, c.cust_name, f.prod_id, f.time_id, f.quantity_sold from sales_fact f,
customer_dim c where f.cust_id = c.cust_id and f.cust_id = 2 order by f.prod_id, f.time_id
Plan hash value: 2760194159
-----------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                    |       |       |    45 (100)|          |
|   1 |  SORT ORDER BY                 |                    |   197 | 12017 |    45   (3)| 00:00:01 |
|*  2 |   HASH JOIN                    |                    |   197 | 12017 |    44   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID | CUSTOMER_DIM       |     1 |    42 |     2   (0)| 00:00:01 |
|*  4 |     INDEX UNIQUE SCAN          | CUST_ID_PK         |     1 |       |     1   (0)| 00:00:01 |
|   5 |    TABLE ACCESS BY INDEX ROWID | SALES_FACT         |   197 |  3743 |    42   (0)| 00:00:01 |
|   6 |     BITMAP CONVERSION TO ROWIDS|                    |       |       |            |          |
|*  7 |      BITMAP INDEX SINGLE VALUE | SALES_CUST_BMP_NDX |       |       |            |          |
-----------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("F"."CUST_ID"="C"."CUST_ID")
   4 - access("C"."CUST_ID"=2)
   7 - access("F"."CUST_ID"=2)
27 rows selected.

SQL>
```

Other Methods -- 2

```
SQL> select * from table(dbms_xplan.display_awr('&sql_id'));
Enter value for sql_id: 1qgc21cjkb741

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------
     ---------------------------
SQL_ID 1qgc21cjkb741
--------------------
select c.cust_id, c.cust_name, f.prod_id, f.time_id, f.quantity_sold from sales_fact f,
customer_dim c where f.cust_id = c.cust_id and f.cust_id = 2 order by f.prod_id, f.time_id

Plan hash value: 2760194159

-------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                     |       |       |    45 (100)|          |
|   1 |  SORT ORDER BY                     |                     |   197 | 12017 |    45   (3)| 00:00:01 |
|   2 |   HASH JOIN                        |                     |   197 | 12017 |    44   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID     | CUSTOMER_DIM        |     1 |    42 |     2   (0)| 00:00:01 |
|   4 |     INDEX UNIQUE SCAN              | CUST_ID_PK          |     1 |       |     1   (0)| 00:00:01 |
|   5 |    TABLE ACCESS BY INDEX ROWID     | SALES_FACT          |   197 |  3743 |    42   (0)| 00:00:01 |
|   6 |     BITMAP CONVERSION TO ROWIDS    |                     |       |       |            |          |
|   7 |      BITMAP INDEX SINGLE VALUE     | SALES_CUST_BMP_NDX  |       |       |            |          |
-------------------------------------------------------------------------------------------------------


20 rows selected.

SQL>
```

Unfortunately, the PREDICATES section is missing.

(did you notice the transitive closure ?)

Other Methods -- 3

```
SQL> set serveroutput off
SQL> select /*+ gather_plan_statistics */
c.cust_id, c.cust_name, f.prod_id, f.time_id, f.quantity_sold
from sales_fact f, customer_dim c
where f.cust_id = c.cust_id
and f.cust_id = 2
order by f.prod_id, f.time_id
/

178 rows selected.

SQL>
```

```
SQL> select * from table(dbms_xplan.display_cursor('','','ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------------------------------------
--
SQL_ID  36cf1j4n2wcw3, child number 0
-------------------------------------
select /*+ gather_plan_statistics */ c.cust_id, c.cust_name, f.prod_id, f.time_id, f.quantity_sold from sales_fact f,
customer_dim c where f.cust_id = c.cust_id and f.cust_id = 2 order by f.prod_id, f.time_id

Plan hash value: 2760194159

------------------------------------------------------------------------------------------------------------------------------------------------------
--
------
| Id  | Operation                     | Name               | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  OMem |  1Mem |
Used
-Mem |
------------------------------------------------------------------------------------------------------------------------------------------------------
--
------
|   1 |  SORT ORDER BY                |                    |      1 |    197 |    178 |00:00:00.01 |     157 | 13312 | 13312
|12288
  (0)|
|*  2 |   HASH JOIN                   |                    |      1 |    197 |    178 |00:00:00.01 |     157 |   788K|   788K|
306
K (0)|
|   3 |    TABLE ACCESS BY INDEX ROWID | CUSTOMER_DIM       |      1 |      1 |      1 |00:00:00.01 |       4 |       |       |
     |
|*  4 |     INDEX UNIQUE SCAN         | CUST_ID_PK         |      1 |      1 |      1 |00:00:00.01 |       2 |       |       |
     |
|   5 |    TABLE ACCESS BY INDEX ROWID | SALES_FACT         |      1 |    197 |    178 |00:00:00.01 |     153 |       |       |
     |
|   6 |     BITMAP CONVERSION TO ROWIDS|                    |      1 |        |    178 |00:00:00.01 |       2 |       |       |
     |
|*  7 |      BITMAP INDEX SINGLE VALUE | SALES_CUST_BMP_NDX |      1 |        |      1 |00:00:00.01 |       2 |       |       |
     |
------------------------------------------------------------------------------------------------------------------------------------------------------
--
------

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

   2 - access("F"."CUST_ID"="C"."CUST_ID")
   4 - access("C"."CUST_ID"=2)
   7 - access("F"."CUST_ID"=2)

26 rows selected.
SQL>
```

Options to XPLAN

```
SQL> select * from table(dbms_xplan.display_cursor('&sql_id','','BASIC'));
Enter value for sql_id: 1qgc21cjkb741

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------
     --------------------------
EXPLAINED SQL STATEMENT:
------------------------
select c.cust_id, c.cust_name, f.prod_id, f.time_id, f.quantity_sold
from sales_fact f, customer_dim c where f.cust_id = c.cust_id and
f.cust_id = 2 order by f.prod_id, f.time_id

Plan hash value: 2760194159

-------------------------------------------------------------
| Id  | Operation                     | Name               |
-------------------------------------------------------------
|   0 | SELECT STATEMENT              |                    |
|   1 |  SORT ORDER BY                |                    |
|   2 |   HASH JOIN                   |                    |
|   3 |    TABLE ACCESS BY INDEX ROWID | CUSTOMER_DIM       |
|   4 |     INDEX UNIQUE SCAN         | CUST_ID_PK         |
|   5 |    TABLE ACCESS BY INDEX ROWID | SALES_FACT         |
|   6 |     BITMAP CONVERSION TO ROWIDS|                    |
|   7 |      BITMAP INDEX SINGLE VALUE | SALES_CUST_BMP_NDX |
-------------------------------------------------------------

21 rows selected.
SQL>
```

More Options

- For more options to DBMS_XPLAN see $ORACLE_HOME/rdbms/admin/dbmsxpln.sql

More examples

From https://forums.oracle.com/forums/thread.jspa?messageID=9881406

```
SELECT Max(logId) AS logId FROM online_users_t
  WHERE online_users_date >= to_date('2011-09-19 10:00:00') - 3.2 AND online_users_date <= to_date('2011-
    09-19 10:00:00') AND online_users_result in (1, -1)
  GROUP BY online_users_user
```

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                    | 24800 |   629K|  1336   (1)| 00:00:17 |
|   1 |  HASH GROUP BY                |                    | 24800 |   629K|  1336   (1)| 00:00:17 |
|*  2 |   TABLE ACCESS BY INDEX ROWID| ONLINE_USERS_T     | 38833 |   985K|  1334   (1)| 00:00:17 |
|*  3 |    INDEX RANGE SCAN           | ONLINE_USERS_T_IDX |   116K|       |   313   (1)| 00:00:04 |
-----------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("ONLINE_USERS_RESULT"=(-1) OR "ONLINE_USERS_RESULT"=1)
   3 - access("ONLINE_USERS_DATE">=TO_DATE(' 2011-09-16 05:12:00', 'syyyy-mm-dd
              hh24:mi:ss') AND "ONLINE_USERS_DATE"<=TO_DATE(' 2011-09-19 10:00:00', 'syyyy-mm-dd
              hh24:mi:ss'))
```

Assuming that the “Rows” (cardinality) figures are correct, what is noticeable in this plan ?

Answer : **“Throwaway”**

11g

- Real Time monitoring using V$SQL_MONITOR and V$SQL_PLAN_MONITOR

Quiz

- What’s the difference between “Index Unique Scan” and “Index Range Scan” when the target *is* a Unique Index ?
- What’s the difference between “Index Full Scan” and “Index Fast Full Scan” ?
- What are the number of “rows” when a BitMap index appears ?
- What is View Merging ? What is SubQuery UnNesting ?