In the last couple of years I have seen several very good references for the DBMS_XPLAN parameters, but it seems that those references are typically hard to locate when needed. The documentation, while good, is a little confusing because few example outputs are included. From the documentation:

"format: Controls the level of details for the plan. It accepts four values:

BASIC: Displays the minimum information in the plan—the operation ID, the operation name and its option.

TYPICAL: This is the default. Displays the most relevant information in the plan (operation id, name and option, #rows, #bytes and optimizer cost). Pruning, parallel and predicate information are only displayed when applicable. Excludes only PROJECTION, ALIAS and REMOTE SQL information (see below).

SERIAL: Like TYPICAL except that the parallel information is not displayed, even if the plan executes in parallel.

ALL: Maximum user level. Includes information displayed with the TYPICAL level with additional information (PROJECTION, ALIAS and information about REMOTE SQL if the operation is distributed).

Format keywords must be separated by either a comma or a space:

ROWS – if relevant, shows the number of rows estimated by the optimizer

BYTES – if relevant, shows the number of bytes estimated by the optimizer

COST – if relevant, shows optimizer cost information

PARTITION – if relevant, shows partition pruning information

PARALLEL – if relevant, shows PX information (distribution method and table queue information)

PREDICATE – if relevant, shows the predicate section

PROJECTION –if relevant, shows the projection section

ALIAS – if relevant, shows the "Query Block Name / Object Alias" section

REMOTE – if relevant, shows the information for distributed query (for example, remote from serial distribution and remote SQL)

NOTE – if relevant, shows the note section of the explain plan

IOSTATS – assuming that basic plan statistics are collected when SQL statements are executed (either by using the gather_plan_statistics hint or by setting the parameter statistics_level to ALL), this format shows IO statistics for ALL (or only for the LAST as shown below) executions of the cursor.

MEMSTATS – Assuming that PGA memory management is enabled (that is, pga_aggregate_target parameter is set to a non 0 value), this format allows to display memory management statistics (for example, execution mode of the operator, how much memory was used, number of bytes spilled to disk, and so on). These statistics only apply to memory intensive operations like hash-joins, sort or some bitmap operators.

ALLSTATS – A shortcut for 'IOSTATS MEMSTATS'

LAST – By default, plan statistics are shown for all executions of the cursor. The keyword LAST can be specified to see only the statistics for the last execution.

The following two formats are deprecated but supported for backward compatibility:

RUNSTATS_TOT – Same as IOSTATS, that is, displays IO statistics for all executions of the specified cursor.

RUNSTATS_LAST – Same as IOSTATS LAST, that is, displays the runtime statistics for the last execution of the cursor

Format keywords can be prefixed by the sign '-' to exclude the specified information. For example, '-PROJECTION' excludes projection information."

This blog article will attempt to demonstrate using Oracle Database 11.2.0.1 as many of the FORMAT parameters for DBMS_XPLAN.DISPLAY_CURSOR as is possible. We will use four test tables for the demonstration. The test table definitions follow (warning – creating table T1 could require an hour or longer):

```
CREATE TABLE T1 (
  ID NUMBER,
  DESCRIPTION VARCHAR2(80));

INSERT INTO T1
SELECT
  CEIL(ABS(SIN(ROWNUM/9.9999)*10000)),
  'This is the long description for this number '|| TO_CHAR(CEIL(ABS(SIN(ROWNUM/9.9999)*10000)))
FROM
  (SELECT
    ROWNUM RN
  FROM
    DUAL
  CONNECT BY
    LEVEL<=10000),
  (SELECT
    ROWNUM RN
  FROM
    DUAL
  CONNECT BY
    LEVEL<=10000);

CREATE INDEX IND_T1 ON T1(ID);

CREATE TABLE T2 AS
SELECT
  ROWNUM C1,
  LPAD('A',100,'A') C2
FROM
  DUAL
CONNECT BY
  LEVEL<=10000;

CREATE TABLE T3 AS
```

```
SELECT
  ROWNUM C1,
  LPAD('A',100,'A') C2
FROM
  DUAL
CONNECT BY
  LEVEL<=10000;

CREATE TABLE T4 AS
SELECT
  ROWNUM C1,
  LPAD('A',100,'A') C2
FROM
  DUAL
CONNECT BY
  LEVEL<=10000;

CREATE INDEX IND_T4 ON T4(C1);

COMMIT;

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T1',CASCADE=>TRUE,METHOD_OPT=>'FOR ALL COLUMNS SIZE 1')
EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T3',CASCADE=>TRUE)
EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T4',CASCADE=>TRUE)
```

I will start by setting the STATISTICS_LEVEL parameter to ALL at the session level. In general, this parameter should be set to TYPICAL (edit March 5, 2010: *a /*+ GATHER_PLAN_STATISTICS */ hint may be used immediately after the SELECT keyword in the SQL statement to provide almost the same level of detail as would be available when setting the STATISTICS_LEVEL parameter to ALL, without as significant of a negative performance impact – see the Related Blog Articles links below for examples that use the hint*). I will also disable the output of rows to the SQL*Plus window:

```
ALTER SESSION SET STATISTICS_LEVEL='ALL';
SET AUTOTRACE TRACEONLY STATISTICS
```

The following SQL statement is executed twice in session 1:

```
SELECT /*+ PARALLEL(T1 8 ) */
  *
FROM
  T1;

Statistics
----------------------------------------------------------
         24  recursive calls
```

```
           0  db block gets
      815350  consistent gets
      813217  physical reads
           0  redo size
  5509985356  bytes sent via SQL*Net to client
     1100512  bytes received via SQL*Net from client
      100001  SQL*Net roundtrips to/from client
           0  sorts (memory)

           0  sorts (disk)
   100000000  rows processed
```

In session 2 we determine the SQL_ID and CHILD_NUMBER of the SQL statement that is executing in session 1:

```
SELECT
  SQL_ID,
  CHILD_NUMBER
FROM
  V$SQL
WHERE
  SQL_TEXT LIKE 'SELECT /*+ PARALLEL(T1 8 ) */%';

SQL_ID        CHILD_NUMBER
------------- ------------
6kd5fkqdjb8fu            0
```

BASIC Format Parameter Value:

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('6kd5fkqdjb8fu',0,'BASIC'));

EXPLAINED SQL STATEMENT:
------------------------
SELECT /*+ PARALLEL(T1 8 ) */   * FROM   T1

Plan hash value: 2494645258
```

```
----------------------------------------
| Id  | Operation            | Name     |
----------------------------------------
|   0 | SELECT STATEMENT     |          |
|   1 |  PX COORDINATOR      |          |
|   2 |   PX SEND QC (RANDOM)| :TQ10000 |
|   3 |    PX BLOCK ITERATOR |          |
|   4 |     TABLE ACCESS FULL| T1       |
----------------------------------------
```

SERIAL Format Parameter Value:

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('6kd5fkqdjb8fu',0,'SERIAL'));

SQL_ID  6kd5fkqdjb8fu, child number 0
-------------------------------------
SELECT /*+ PARALLEL(T1 8 ) */   * FROM   T1

Plan hash value: 2494645258

-------------------------------------------------------------------------------------
| Id  | Operation            | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |          |       |       | 30907 (100)|          |
|   1 |  PX COORDINATOR      |          |       |       |            |          |
|   2 |   PX SEND QC (RANDOM)| :TQ10000 |   100M|  5149M| 30907   (1)| 00:06:11 |
|   3 |    PX BLOCK ITERATOR |          |   100M|  5149M| 30907   (1)| 00:06:11 |
|*  4 |     TABLE ACCESS FULL| T1       |   100M|  5149M| 30907   (1)| 00:06:11 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
 4 - access(:Z>=:Z AND :Z<=:Z
```

TYPICAL Format Parameter Value:

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('6kd5fkqdjb8fu',0,'TYPICAL'));

SQL_ID  6kd5fkqdjb8fu, child number 0
-------------------------------------
SELECT /*+ PARALLEL(T1 8 ) */   * FROM   T1

Plan hash value: 2494645258

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name     | Rows  | Bytes | Cost (%CPU)|  Time    |    TQ  |IN-OUT| PQ Distrib |
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |          |       |       | 30907 (100)|          |        |      |            |
|   1 |  PX COORDINATOR      |          |       |       |            |          |        |      |            |
|   2 |   PX SEND QC (RANDOM)| :TQ10000 |   100M|  5149M| 30907   (1)| 00:06:11 |  Q1,00 | P->S | QC (RAND)  |
|   3 |    PX BLOCK ITERATOR |          |   100M|  5149M| 30907   (1)| 00:06:11 |  Q1,00 | PCWC |            |
|*  4 |     TABLE ACCESS FULL| T1       |   100M|  5149M| 30907   (1)| 00:06:11 |  Q1,00 | PCWP |            |
-----------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
 4 - access(:Z>=:Z AND :Z<=:Z)
```

ALL Format Parameter Value

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('6kd5fkqdjb8fu',0,'ALL'));

SQL_ID  6kd5fkqdjb8fu, child number 0
-------------------------------------
```

```
SELECT /*+ PARALLEL(T1 8 ) */   * FROM   T1

Plan hash value: 2494645258

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name     | Rows  | Bytes | Cost (%CPU)|   Time   |    TQ  |IN-OUT| PQ Distrib |
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |          |       |       | 30907 (100)|          |        |      |            |
|   1 |  PX COORDINATOR      |          |       |       |            |          |        |      |            |
|   2 |   PX SEND QC (RANDOM)| :TQ10000 |   100M|  5149M| 30907   (1)| 00:06:11 |  Q1,00 | P->S | QC (RAND)  |
|   3 |    PX BLOCK ITERATOR |          |   100M|  5149M| 30907   (1)| 00:06:11 |  Q1,00 | PCWC |            |
|*  4 |     TABLE ACCESS FULL| T1       |   100M|  5149M| 30907   (1)| 00:06:11 |  Q1,00 | PCWP |            |
-----------------------------------------------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------
   1 - SEL$1
   4 - SEL$1 / T1@SEL$1

Predicate Information (identified by operation id):
---------------------------------------------------
   4 - access(:Z>=:Z AND :Z<=:Z)

Column Projection Information (identified by operation id):
-----------------------------------------------------------
   1 - "T1"."ID"[NUMBER,22], "T1"."DESCRIPTION"[VARCHAR2,80]
   2 - (#keys=0) "T1"."ID"[NUMBER,22], "T1"."DESCRIPTION"[VARCHAR2,80]
   3 - "T1"."ID"[NUMBER,22], "T1"."DESCRIPTION"[VARCHAR2,80]
   4 - "T1"."ID"[NUMBER,22], "T1"."DESCRIPTION"[VARCHAR2,80]
```

ALLSTATS Format Parameter Value

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('6kd5fkqdjb8fu',0,'ALLSTATS'));

SQL_ID  6kd5fkqdjb8fu, child number 0
```

```
-------------------------------------
SELECT /*+ PARALLEL(T1 8 ) */   * FROM   T1
Plan hash value: 2494645258
-------------------------------------------------------------------------------------------------------------
| Id  | Operation            | Name     | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------------------

|   0 | SELECT STATEMENT     |          |      2 |        |    200M|00:00:47.47 |      52 |     12 |
|   1 |  PX COORDINATOR      |          |      2 |        |    200M|00:00:47.47 |      52 |     12 |
|   2 |   PX SEND QC (RANDOM)| :TQ10000 |      0 |    100M|      0 |00:00:00.01 |       0 |      0 |
|   3 |    PX BLOCK ITERATOR |          |     16 |    100M|    200M|00:01:56.71 |    1630K|   1626K|
|*  4 |     TABLE ACCESS FULL| T1       |    248 |    100M|    200M|00:00:53.14 |    1528K|   1524K|
-------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
4 - access(:Z>=:Z AND :Z<=:Z)
```

ALLSTATS LAST Format Parameter Value

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('6kd5fkqdjb8fu',0,'ALLSTATS LAST'));

SQL_ID  6kd5fkqdjb8fu, child number 0
-------------------------------------
SELECT /*+ PARALLEL(T1 8 ) */   * FROM   T1

Plan hash value: 2494645258

-----------------------------------------------------------------------------------------------
| Id  | Operation            | Name     | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |          |      1 |        |    100M|00:00:23.61 |      25 |
|   1 |  PX COORDINATOR      |          |      1 |        |    100M|00:00:23.61 |      25 |
|   2 |   PX SEND QC (RANDOM)| :TQ10000 |      0 |    100M|      0 |00:00:00.01 |       0 |
|   3 |    PX BLOCK ITERATOR |          |      0 |    100M|      0 |00:00:00.01 |       0 |
|*  4 |     TABLE ACCESS FULL| T1       |      0 |    100M|      0 |00:00:00.01 |       0 |
```

```
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
 4 - access(:Z>=:Z AND :Z<=:Z)
```

Next SQL Statement Executed in Session 1:

```
VARIABLE N1 NUMBER
VARIABLE N2 NUMBER
EXEC :N1:=1
EXEC :N2:=100

SELECT
  *
FROM
  T2,
  T4
WHERE
  T2.C1 BETWEEN :N1 AND :N2
  AND T2.C1=T4.C1;

Statistics
----------------------------------------------------
   340  recursive calls
     0  db block gets
   294  consistent gets
   171  physical reads
     0  redo size
  1994  bytes sent via SQL*Net to client
   360  bytes received via SQL*Net from client
     2  SQL*Net roundtrips to/from client

     6  sorts (memory)
     0  sorts (disk)
   100  rows processed
```

In session 2 we determine the SQL_ID and CHILD_NUMBER of the SQL statement that is executing in session 1:

```
SELECT
  SQL_ID,
  CHILD_NUMBER
```

```
FROM
  V$SQL
WHERE
  SQL_TEXT LIKE 'SELECT%T2.C1 BETWEEN :N1 AND :N2%'
  AND SQL_TEXT NOT LIKE '%V$SQL%';

SQL_ID        CHILD_NUMBER
------------- ------------
75chksrfa5fbt            0
```

Starting Point, Viewing the Last Execution Statistics:

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('75chksrfa5fbt',0,'ALLSTATS LAST'));

SQL_ID  75chksrfa5fbt, child number 0
-------------------------------------
SELECT   * FROM   T2,   T4 WHERE   T2.C1 BETWEEN :N1 AND :N2   AND
T2.C1=T4.C1

Plan hash value: 3771400634

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name    | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |         |      1 |        |    100 |00:00:00.03 |     171 |     29 |
|*  1 |  FILTER                       |         |      1 |        |    100 |00:00:00.03 |     171 |     29 |
|   2 |   NESTED LOOPS                |         |      1 |        |    100 |00:00:00.03 |     171 |     29 |
|   3 |    NESTED LOOPS               |         |      1 |      2 |    100 |00:00:00.02 |     168 |     21 |
|*  4 |     TABLE ACCESS FULL         | T2      |      1 |      2 |    100 |00:00:00.01 |     159 |     13 |
|*  5 |     INDEX RANGE SCAN          | IND_T4  |    100 |      1 |    100 |00:00:00.01 |       9 |      8 |
|   6 |    TABLE ACCESS BY INDEX ROWID| T4      |    100 |      1 |    100 |00:00:00.01 |       3 |      8 |
-----------------------------------------------------------------------------------------------------------

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

```
---------------------------------------------------
   1 - filter(:N1<=:N2)
   4 - filter(("T2"."C1">=:N1 AND "T2"."C1"<=:N2))
   5 - access("T2"."C1"="T4"."C1")
       filter(("T4"."C1"<=:N2 AND "T4"."C1">=:N1))

Note
-----
   - dynamic sampling used for this statement (level=2)
```

Enabling Extra Output:

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('75chksrfa5fbt',0,'ALLSTATS LAST +PEEKED_BINDS +PROJECTION +ALIAS +PREDICATE +COST +BYTES'));

SQL_ID  75chksrfa5fbt, child number 0
-------------------------------------
SELECT   * FROM   T2,   T4 WHERE   T2.C1 BETWEEN :N1 AND :N2   AND  T2.C1=T4.C1

Plan hash value: 3771400634


-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name    | Starts | E-Rows |E-Bytes| Cost(%CPU) | A-Rows |   A-Time   | Buffers | Reads  |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |         |      1 |        |       |    51 (100)|    100 |00:00:00.03 |     171 |     29 |
|*  1 |  FILTER                       |         |      1 |        |       |            |    100 |00:00:00.03 |     171 |     29 |
|   2 |   NESTED LOOPS                |         |      1 |        |       |            |    100 |00:00:00.03 |     171 |     29 |
|   3 |    NESTED LOOPS               |         |      1 |      2 |   340 |    51   (0)|    100 |00:00:00.02 |     168 |     21 |
|*  4 |     TABLE ACCESS FULL         | T2      |      1 |      2 |   130 |    47   (0)|    100 |00:00:00.01 |     159 |     13 |
|*  5 |     INDEX RANGE SCAN          | IND_T4  |    100 |      1 |       |     1   (0)|    100 |00:00:00.01 |       9 |      8 |
|   6 |    TABLE ACCESS BY INDEX ROWID| T4      |    100 |      1 |   105 |     2   (0)|    100 |00:00:00.01 |       3 |      8 |
-----------------------------------------------------------------------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------
   1 - SEL$1
```

```
   1 - SEL$1
   4 - SEL$1 / T2@SEL$1
   5 - SEL$1 / T4@SEL$1
   6 - SEL$1 / T4@SEL$1

Peeked Binds (identified by position):
--------------------------------------
   1 - (NUMBER): 1
   2 - (NUMBER): 100

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(:N1<=:N2)
   4 - filter(("T2"."C1">=:N1 AND "T2"."C1"<=:N2))
   5 - access("T2"."C1"="T4"."C1")
       filter(("T4"."C1"<=:N2 AND "T4"."C1">=:N1))

Column Projection Information (identified by operation id):
-----------------------------------------------------------
   1 - "T2"."C1"[NUMBER,22], "T2"."C2"[VARCHAR2,100], "T4"."C1"[NUMBER,22], "T4"."C2"[VARCHAR2,100]
   2 - "T2"."C1"[NUMBER,22], "T2"."C2"[VARCHAR2,100], "T4"."C1"[NUMBER,22], "T4"."C2"[VARCHAR2,100]
   3 - "T2"."C1"[NUMBER,22], "T2"."C2"[VARCHAR2,100], "T4".ROWID[ROWID,10], "T4"."C1"[NUMBER,22]
   4 - "T2"."C1"[NUMBER,22], "T2"."C2"[VARCHAR2,100]
   5 - "T4".ROWID[ROWID,10], "T4"."C1"[NUMBER,22]
   6 - "T4"."C2"[VARCHAR2,100]

Note
-----
   - dynamic sampling used for this statement (level=2)
```

Removing Output Sections:

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('75chksrfa5fbt',0,'ALLSTATS LAST -NOTE -ROWS -PREDICATE'));

SQL_ID  75chksrfa5fbt, child number 0
```

```
-------------------------------------
SELECT   * FROM   T2,   T4 WHERE   T2.C1 BETWEEN :N1 AND :N2   AND
T2.C1=T4.C1

Plan hash value: 3771400634


-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name   | Starts | A-Rows |   A-Time   | Buffers | Reads  |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |        |      1 |    100 |00:00:00.03 |     171 |     29 |
|   1 |  FILTER                       |        |      1 |    100 |00:00:00.03 |     171 |     29 |
|   2 |   NESTED LOOPS                |        |      1 |    100 |00:00:00.03 |     171 |     29 |
|   3 |    NESTED LOOPS               |        |      1 |    100 |00:00:00.02 |     168 |     21 |
|   4 |     TABLE ACCESS FULL         | T2     |      1 |    100 |00:00:00.01 |     159 |     13 |
|   5 |     INDEX RANGE SCAN          | IND_T4 |    100 |    100 |00:00:00.01 |       9 |      8 |
|   6 |    TABLE ACCESS BY INDEX ROWID| T4     |    100 |    100 |00:00:00.01 |       3 |      8 |
-----------------------------------------------------------------------------------------------------
```

A More Complicated Example

The previous examples were too simple, so let's look at something that is a bit more interesting. We will introduce partitioning, parallel execution, and remote databases. First, let's create a larger version of table T3 with 1,000,000 rows rather than 10,000 rows:

```
DROP TABLE T3 PURGE;

CREATE TABLE T3
  PARTITION BY RANGE(C1)
   (PARTITION P1 VALUES LESS THAN (5),
    PARTITION P2 VALUES LESS THAN (10),
    PARTITION P3 VALUES LESS THAN (20),
    PARTITION P4 VALUES LESS THAN (40),
    PARTITION P5 VALUES LESS THAN (80),
    PARTITION P6 VALUES LESS THAN (160),
    PARTITION P7 VALUES LESS THAN (320),
    PARTITION P8 VALUES LESS THAN (640),
    PARTITION P9 VALUES LESS THAN (1280),
    PARTITION P10 VALUES LESS THAN (2560),
    PARTITION P11 VALUES LESS THAN (5120),
    PARTITION P12 VALUES LESS THAN (10240),
    PARTITION P20 VALUES LESS THAN (MAXVALUE))
AS
```

```
AS
SELECT
  ROWNUM C1,
  LPAD('A',100,'A') C2
FROM
  DUAL
CONNECT BY
  LEVEL<=1000000;

COMMIT;

EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T3',CASCADE=>TRUE)
```

Now let's connect to the database in another session as the SYS user and create a database link to an Oracle 11.1.0.6 database (global names are not used in the database, otherwise we would would need a specific name for the database link, as mentioned here), and then flush the buffer cache:

```
CREATE PUBLIC DATABASE LINK TEST_LINK CONNECT TO TESTUSER IDENTIFIED BY MY_PASS_HERE USING 'o11106';

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;
```

In the remote 11.1.0.6 database, the TESTUSER creates the following tables, and then the SYS user flushes the buffer cache:

```
CREATE TABLE T3 AS
SELECT
  ROWNUM C1,
  LPAD('A',100,'A') C2
FROM
  DUAL
CONNECT BY
  LEVEL<=10000;

CREATE TABLE T4 AS
SELECT
  ROWNUM C1,
  LPAD('A',100,'A') C2
FROM
  DUAL
CONNECT BY
  LEVEL<=10000;

CREATE INDEX IND_T4 ON T4(C1);

COMMIT;
```

```
EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T3',CASCADE=>TRUE)
EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>USER,TABNAME=>'T4',CASCADE=>TRUE)

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH BUFFER_CACHE;
```

Back in the 11.2.0.1 database as our test user, we create a SQL statement to access table T1, T2, T3, T4, and the remote tables T3 and T4:

```
VARIABLE N1 NUMBER
VARIABLE N2 NUMBER
EXEC :N1:=1
EXEC :N2:=200
ALTER SESSION SET STATISTICS_LEVEL='ALL';
SET AUTOTRACE TRACEONLY STATISTICS

SELECT /*+ PARALLEL(8) */
  T2.C1 T2_C1,
  SUBSTR(T2.C2,1,10) T2_C2,
  T3.C1 T3_C1,
  SUBSTR(T3.C2,1,10) T3_C2,
  T4.C1 T4_C1,
  SUBSTR(T4.C2,1,10) T4_C2,
  TL_T3.C1 TL_T3_C1,
  SUBSTR(TL_T3.C2,1,10) TL_T3_C2,
  TL_T4.C1 TL_T4_C1,
  SUBSTR(TL_T4.C2,1,10) TL_T4_C2,
  V_T1.C
FROM
  T2,
  T3,
  T4,
  T3@TEST_LINK TL_T3,
  T4@TEST_LINK TL_T4,
  (SELECT
    ID,
    COUNT(*) C
  FROM
    T1
  GROUP BY
    ID) V_T1
WHERE
```

```
  T2.C1 BETWEEN :N1 AND :N2
  AND T2.C1=T3.C1
  AND T2.C1=T4.C1
  AND T2.C1=TL_T3.C1
  AND T2.C1=TL_T4.C1
  AND T2.C1=V_T1.ID(+)
ORDER BY
  T2.C1;

Statistics
----------------------------------------------------------
       2855  recursive calls
         12  db block gets
       3979  consistent gets
       2893  physical reads
       1324  redo size
       9145  bytes sent via SQL*Net to client
        667  bytes received via SQL*Net from client
         15  SQL*Net roundtrips to/from client
         95  sorts (memory)
          0  sorts (disk)
        200  rows processed
```

Now let's check the execution plan:

```
SET AUTOTRACE OFF

SELECT
  SQL_ID,
  CHILD_NUMBER
FROM
  V$SQL
WHERE
  SQL_TEXT LIKE '%T3@TEST_LINK TL_T3,%'
  AND SQL_TEXT NOT LIKE '%V$SQL%';

SQL_ID        CHILD_NUMBER
------------- ------------
dkmcbpadz15w1            0
dkmcbpadz15w1            1
```

Interesting, two child cursors. Let's see the execution plans:

```
SELECT
  *
FROM
  TABLE(DBMS_XPLAN.DISPLAY_CURSOR('dkmcbpadz15w1',NULL,'ALLSTATS LAST'));

SQL_ID  dkmcbpadz15w1, child number 0
-------------------------------------
SELECT /*+ PARALLEL(8) */   T2.C1 T2_C1,   SUBSTR(T2.C2,1,10) T2_C2,
T3.C1 T3_C1,   SUBSTR(T3.C2,1,10) T3_C2,   T4.C1 T4_C1,
SUBSTR(T4.C2,1,10) T4_C2,   TL_T3.C1 TL_T3_C1,   SUBSTR(TL_T3.C2,1,10)
TL_T3_C2,   TL_T4.C1 TL_T4_C1,   SUBSTR(TL_T4.C2,1,10) TL_T4_C2,
V_T1.C FROM   T2,   T3,   T4,   T3@TEST_LINK TL_T3,   T4@TEST_LINK
TL_T4,   (SELECT     ID,     COUNT(*) C   FROM     T1   GROUP BY
ID) V_T1 WHERE   T2.C1 BETWEEN :N1 AND :N2   AND T2.C1=T3.C1   AND
T2.C1=T4.C1   AND T2.C1=TL_T3.C1   AND T2.C1=TL_T4.C1   AND
T2.C1=V_T1.ID(+) ORDER BY   T2.C1

Plan hash value: 453902047

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                     | Name     | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  |  OMem |  1Mem |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                              |          |      1 |        |    200 |00:00:02.49 |    2609 |   2655 |       |       |
|*  1 |  PX COORDINATOR                               |          |      1 |        |    200 |00:00:02.49 |    2609 |   2655 |       |       |
|   2 |   PX SEND QC (ORDER)                          | :TQ10008 |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |       |       |
|   3 |    SORT ORDER BY                              |          |      0 |     17 |      0 |00:00:00.01 |       0 |      0 | 73728 | 73728 |
|   4 |     PX RECEIVE                                |          |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |       |       |
|   5 |      PX SEND RANGE                            | :TQ10007 |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |       |       |
|*  6 |       FILTER                                  |          |      0 |        |      0 |00:00:00.01 |       0 |      0 |       |       |
|*  7 |        HASH JOIN OUTER                        |          |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |   697K|   697K|
|   8 |         PX RECEIVE                            |          |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |       |       |
|   9 |          PX SEND HASH                         | :TQ10006 |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |       |       |
|* 10 |           HASH JOIN                           |          |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |   705K|   705K|
|  11 |            PART JOIN FILTER CREATE            | :BF0000  |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |       |       |
|  12 |             PX RECEIVE                        |          |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |       |       |
|  13 |              PX SEND BROADCAST                | :TQ10005 |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |       |       |
|* 14 |               HASH JOIN                       |          |      0 |     17 |      0 |00:00:00.01 |       0 |      0 |   713K|   713K|
|* 15 |                HASH JOIN                      |          |      0 |     21 |      0 |00:00:00.01 |       0 |      0 |   727K|   727K|
|* 16 |                 HASH JOIN                     |          |      0 |     25 |      0 |00:00:00.01 |       0 |      0 |   757K|   757K|
|  17 |                  BUFFER SORT                  |          |      0 |        |      0 |00:00:00.01 |       0 |      0 | 73728 | 73728 |
|  18 |                   PX RECEIVE                  |          |      0 |     25 |      0 |00:00:00.01 |       0 |      0 |       |       |
|  19 |                    PX SEND HASH               | :TQ10000 |      0 |     25 |      0 |00:00:00.01 |       0 |      0 |       |       |
```

```
|  20 |                    TABLE ACCESS BY INDEX ROWID| T4       |      1 |     25 |    200 |00:00:00.02 |       6 |     16 |        |        |
|* 21 |                     INDEX RANGE SCAN          | IND_T4   |      1 |     45 |    200 |00:00:00.01 |       2 |      8 |        |        |
|  22 |                  PX RECEIVE                   |          |      0 |     30 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  23 |                   PX SEND HASH                | :TQ10004 |      0 |     30 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  24 |                    PX BLOCK ITERATOR          |          |      0 |     30 |      0 |00:00:00.01 |       0 |      0 |        |        |
|* 25 |                     TABLE ACCESS FULL         | T2       |      0 |     30 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  26 |                BUFFER SORT                    |          |      0 |        |      0 |00:00:00.01 |       0 |      0 | 73728  | 73728  |
|  27 |                 PX RECEIVE                    |          |      0 |     25 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  28 |                  PX SEND HASH                 | :TQ10001 |      0 |     25 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  29 |                   REMOTE                      | T3       |      1 |     25 |    200 |00:00:00.09 |       0 |      0 |        |        |
|  30 |               BUFFER SORT                     |          |      0 |        |      0 |00:00:00.01 |       0 |      0 | 73728  | 73728  |
|  31 |                PX RECEIVE                     |          |      0 |     25 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  32 |                 PX SEND HASH                  | :TQ10002 |      0 |     25 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  33 |                  REMOTE                       | T4       |      1 |     25 |    200 |00:00:00.01 |       0 |      0 |        |        |
|  34 |            PX BLOCK ITERATOR                  |          |      0 |   2500 |      0 |00:00:00.01 |       0 |      0 |        |        |
|* 35 |             TABLE ACCESS FULL                 | T3       |      0 |   2500 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  36 |         BUFFER SORT                           |          |      0 |        |      0 |00:00:00.01 |       0 |      0 | 73728  | 73728  |
|  37 |          PX RECEIVE                           |          |      0 |  10000 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  38 |           PX SEND HASH                        | :TQ10003 |      0 |  10000 |      0 |00:00:00.01 |       0 |      0 |        |        |
|  39 |            VIEW                              |          |      1 |  10000 |    200 |00:00:01.03 |    2577 |   2632 |        |        |
|  40 |             HASH GROUP BY                     |          |      1 |  10000 |    200 |00:00:01.03 |    2577 |   2632 |  1115K|  1115K|
|* 41 |              FILTER                           |          |      1 |        |   1273K|00:00:00.78 |    2577 |   2632 |        |        |
|* 42 |               INDEX RANGE SCAN                | IND_T1   |      1 |    250K|   1273K|00:00:00.43 |    2577 |   2632 |        |        |
-------------------------------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(:N1<=:N2)
   6 - filter(:N1<=:N2)
   7 - access("T2"."C1"="V_T1"."ID")
  10 - access("T2"."C1"="T3"."C1")
  14 - access("T2"."C1"="TL_T4"."C1")
  15 - access("T2"."C1"="TL_T3"."C1")
  16 - access("T2"."C1"="T4"."C1")
  21 - access("T4"."C1">=:N1 AND "T4"."C1"<=:N2)
  25 - access(:Z>=:Z AND :Z<=:Z)
       filter(("T2"."C1">=:N1 AND "T2"."C1"<=:N2))
  35 - access(:Z>=:Z AND :Z<=:Z)
       filter(("T3"."C1">=:N1 AND "T3"."C1"<=:N2))
  41 - filter(:N1<=:N2)
  42 - access("ID">=:N1 AND "ID"<=:N2)
```

```
Note
-----
   - dynamic sampling used for this statement (level=5)
   - Degree of Parallelism is 8 because of hint
```

```
SQL_ID  dkmcbpadz15w1, child number 1
-------------------------------------
SELECT /*+ PARALLEL(8) */   T2.C1 T2_C1,   SUBSTR(T2.C2,1,10) T2_C2,
T3.C1 T3_C1,   SUBSTR(T3.C2,1,10) T3_C2,   T4.C1 T4_C1,
SUBSTR(T4.C2,1,10) T4_C2,   TL_T3.C1 TL_T3_C1,   SUBSTR(TL_T3.C2,1,10)
TL_T3_C2,   TL_T4.C1 TL_T4_C1,   SUBSTR(TL_T4.C2,1,10) TL_T4_C2,
V_T1.C FROM   T2,   T3,   T4,   T3@TEST_LINK TL_T3,   T4@TEST_LINK
TL_T4,   (SELECT     ID,     COUNT(*) C   FROM     T1   GROUP BY
ID) V_T1 WHERE   T2.C1 BETWEEN :N1 AND :N2   AND T2.C1=T3.C1   AND
T2.C1=T4.C1   AND T2.C1=TL_T3.C1   AND T2.C1=TL_T4.C1   AND
T2.C1=V_T1.ID(+) ORDER BY   T2.C1

Plan hash value: 453902047

-----------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                      | Name     | Starts | E-Rows | A-Rows |   A-Time   |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                               |          |      0 |        |      0 |00:00:00.01 |       |       |          |
|*  1 |  PX COORDINATOR                                |          |      0 |        |      0 |00:00:00.01 |       |       |          |
|   2 |   PX SEND QC (ORDER)                           | :TQ10008 |      0 |     17 |      0 |00:00:00.01 |       |       |          |
|   3 |    SORT ORDER BY                               |          |      0 |     17 |      0 |00:00:00.01 | 36864 | 36864 | 4096  (0)|
|   4 |     PX RECEIVE                                 |          |      0 |     17 |      0 |00:00:00.01 |       |       |          |
|   5 |      PX SEND RANGE                             | :TQ10007 |      0 |     17 |      0 |00:00:00.01 |       |       |          |
|*  6 |       FILTER                                   |          |      1 |        |     25 |00:00:01.03 |       |       |          |
|*  7 |        HASH JOIN OUTER                         |          |      1 |     17 |     25 |00:00:01.03 |   693K|   693K| 1286K (0)|
|   8 |         PX RECEIVE                             |          |      1 |     17 |     25 |00:00:00.01 |       |       |          |
|   9 |          PX SEND HASH                          | :TQ10006 |      0 |     17 |      0 |00:00:00.01 |       |       |          |
|* 10 |           HASH JOIN                            |          |      0 |     17 |      0 |00:00:00.01 |  1278K|   930K| 1260K (0)|
|  11 |            PART JOIN FILTER CREATE             | :BF0000  |      0 |     17 |      0 |00:00:00.01 |       |       |          |
|  12 |             PX RECEIVE                         |          |      0 |     17 |      0 |00:00:00.01 |       |       |          |
|  13 |              PX SEND BROADCAST                 | :TQ10005 |      0 |     17 |      0 |00:00:00.01 |       |       |          |
|* 14 |               HASH JOIN                        |          |      1 |     17 |     25 |00:00:00.14 |   705K|   705K| 1139K (0)|
|* 15 |                HASH JOIN                       |          |      1 |     21 |     25 |00:00:00.12 |   720K|   720K| 1129K (0)|
|* 16 |                 HASH JOIN                      |          |      1 |     25 |     25 |00:00:00.03 |   763K|   763K| 1128K (0)|
|  17 |                  BUFFER SORT                   |          |      1 |        |     25 |00:00:00.02 |  4096 |  4096 | 4096  (0)|
```

```
|  18 |                   PX RECEIVE                   |          |      1 |     25 |     25 |00:00:00.02 |       |       |          |
|  19 |                    PX SEND HASH                | :TQ10000 |      0 |     25 |      0 |00:00:00.01 |       |       |          |
|  20 |                     TABLE ACCESS BY INDEX ROWID| T4       |      0 |     25 |      0 |00:00:00.01 |       |       |          |
|* 21 |                      INDEX RANGE SCAN          | IND_T4   |      0 |     45 |      0 |00:00:00.01 |       |       |          |
|  22 |                  PX RECEIVE                    |          |      1 |     30 |     25 |00:00:00.01 |       |       |          |
|  23 |                   PX SEND HASH                 | :TQ10004 |      0 |     30 |      0 |00:00:00.01 |       |       |          |
|  24 |                    PX BLOCK ITERATOR           |          |      0 |     30 |      0 |00:00:00.01 |       |       |          |
|* 25 |                     TABLE ACCESS FULL          | T2       |      0 |     30 |      0 |00:00:00.01 |       |       |          |
|  26 |                 BUFFER SORT                    |          |      1 |        |     25 |00:00:00.09 | 36864 | 36864 | 4096  (0)|
|  27 |                  PX RECEIVE                    |          |      1 |     25 |     25 |00:00:00.09 |       |       |          |
|  28 |                   PX SEND HASH                 | :TQ10001 |      0 |     25 |      0 |00:00:00.01 |       |       |          |
|  29 |                    REMOTE                      | T3       |      0 |     25 |      0 |00:00:00.01 |       |       |          |
|  30 |                BUFFER SORT                     |          |      1 |        |     25 |00:00:00.01 | 36864 | 36864 | 4096  (0)|
|  31 |                 PX RECEIVE                     |          |      1 |     25 |     25 |00:00:00.01 |       |       |          |
|  32 |                  PX SEND HASH                  | :TQ10002 |      0 |     25 |      0 |00:00:00.01 |       |       |          |
|  33 |                   REMOTE                       | T4       |      0 |     25 |      0 |00:00:00.01 |       |       |          |
|  34 |            PX BLOCK ITERATOR                   |          |      0 |   2500 |      0 |00:00:00.01 |       |       |          |
|* 35 |             TABLE ACCESS FULL                  | T3       |      0 |   2500 |      0 |00:00:00.01 |       |       |          |
|  36 |         BUFFER SORT                            |          |      1 |        |     25 |00:00:01.03 | 18432 | 18432 | 2048  (0)|
|  37 |          PX RECEIVE                            |          |      1 |  10000 |     25 |00:00:01.03 |       |       |          |
|  38 |           PX SEND HASH                         | :TQ10003 |      0 |  10000 |      0 |00:00:00.01 |       |       |          |
|  39 |            VIEW                               |          |      0 |  10000 |      0 |00:00:00.01 |       |       |          |
|  40 |             HASH GROUP BY                      |          |      0 |  10000 |      0 |00:00:00.01 |    10M|  3024K|          |
|* 41 |              FILTER                            |          |      0 |        |      0 |00:00:00.01 |       |       |          |
|* 42 |               INDEX RANGE SCAN                 | IND_T1   |      0 |    250K|      0 |00:00:00.01 |       |       |          |
-------------------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(:N1<=:N2)
   6 - filter(:N1<=:N2)
   7 - access("T2"."C1"="V_T1"."ID")
  10 - access("T2"."C1"="T3"."C1")
  14 - access("T2"."C1"="TL_T4"."C1")
  15 - access("T2"."C1"="TL_T3"."C1")
  16 - access("T2"."C1"="T4"."C1")
  21 - access("T4"."C1">=:N1 AND "T4"."C1"<=:N2)
  25 - access(:Z>=:Z AND :Z<=:Z)
       filter(("T2"."C1">=:N1 AND "T2"."C1"<=:N2))
  35 - access(:Z>=:Z AND :Z<=:Z)
       filter(("T3"."C1">=:N1 AND "T3"."C1"<=:N2))
  41 - filter(:N1<=:N2)
```

```
   42 - access("ID">=:N1 AND "ID"<=:N2)

Note
-----
   - dynamic sampling used for this statement (level=5)
   - automatic DOP: Computed Degree of Parallelism is 8
```

There are a couple of interesting items that you might notice, maybe someone can explain why:

The execution plan for the first child cursor shows on line ID 25 that 0 rows were retrieved from table T2, yet the execution plan shows that 200 rows were retrieved.

SHOW PARAMETER OPTIMIZER_DYNAMIC_SAMPLING shows that dynamic sampling is set to 2, yet the Note section of the execution plans show that dynamic sampling at level 5 was performed (statistics were not collected for table T2).

The Note section of the first child cursor shows that the degree of parallelism is 8 because of a hint, while the Note section of the second child cursor shows that the degree of parallelism was automatically computed as 8.

What was the purpose of the second child cursor? No rows were returned, yet some lines in that plan show that 25 rows were retrieved.

Did I miss something?

Adding and Removing Items from the DBMS_XPLAN Output

The following execution plan was created by specifying the format parameters displayed in the blue box. The yellow boxes indicate where those items appear in the execution plan, and how to remove other items that appear by default when the ALLSTATS LAST format parameter is provided.