```
------------ Is the RBO really predictable?
Connected.
------------ Create a table to play with
SQL> create table rbo(c1 number(8), c2 number(8), c3 number(8), c4
number(8));

Table created.

SQL> insert into rbo select level, level, level, level from dual connect
by level <= 1000;

1000 rows created.

------------ Create an index
SQL> create index rbo_c1c2_idx on rbo(c1,c2);

Index created.

------------ Perform a RBO optimized query
SQL> select /*+ rule */ * from rbo where c1 = 1;

        C1         C2         C3         C4
---------- ---------- ---------- ----------
         1          1          1          1

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
SQL_ID  0xgwprdfsawp5, child number 0
-------------------------------------
select /*+ rule */ * from rbo where c1 = 1

Plan hash value: 616638904

-------------------------------------------------
| Id  | Operation                   | Name         |
-------------------------------------------------
|   0 | SELECT STATEMENT            |              |
|   1 |  TABLE ACCESS BY INDEX ROWID| RBO          |
|*  2 |   INDEX RANGE SCAN          | RBO_C1C2_IDX |
-------------------------------------------------

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

   2 - access("C1"=1)

Note
-----
   - rule based optimizer used (consider using cbo)


------------ Create another index
```

```
SQL> create index rbo_c1c3c4_idx on rbo(c1,c3,c4);

Index created.

------------ Re-run our query
SQL> select /*+ rule */ * from rbo where c1 = 1;

        C1         C2         C3         C4
---------- ---------- ---------- ----------
         1          1          1          1

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
SQL_ID  0xgwprdfsawp5, child number 0
-------------------------------------
select /*+ rule */ * from rbo where c1 = 1

Plan hash value: 3111888641

-----------------------------------------------------
| Id  | Operation                    | Name           |
-----------------------------------------------------
|   0 | SELECT STATEMENT             |                |
|   1 |  TABLE ACCESS BY INDEX ROWID| RBO            |
|*  2 |   INDEX RANGE SCAN           | RBO_C1C3C4_IDX |
-----------------------------------------------------

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

   2 - access("C1"=1)

Note
-----
   - rule based optimizer used (consider using cbo)


------------ Re-create our first index
SQL> drop index rbo_c1c2_idx;

Index dropped.

SQL> create index rbo_c1c2_idx on rbo(c1,c2);

Index created.

------------ Re-run our query
SQL> select /*+ rule */ * from rbo where c1 = 1;

        C1         C2         C3         C4
---------- ---------- ---------- ----------
         1          1          1          1
```

```
------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor());

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
--------
SQL_ID  0xgwprdfsawp5, child number 0
-------------------------------------
select /*+ rule */ * from rbo where c1 = 1

Plan hash value: 616638904

-------------------------------------------------
| Id  | Operation                   | Name         |
-------------------------------------------------
|   0 | SELECT STATEMENT            |              |
|   1 |  TABLE ACCESS BY INDEX ROWID| RBO          |
|*  2 |   INDEX RANGE SCAN          | RBO_C1C2_IDX |
-------------------------------------------------

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

   2 - access("C1"=1)

Note
-----
   - rule based optimizer used (consider using cbo)


------------ Cleanup
SQL> set echo off

Table dropped.

Table dropped.
```

```
------------ Optimizer Plan Stability
Connected.
------------ Create a table to play with
SQL> create table foo(c1 number(8), c2 number(8), c3 char(50));

Table created.

SQL> insert into foo select level, 1, 1 from dual connect by level <=
100000;

100000 rows created.

SQL> exec dbms_stats.gather_table_stats(null,'foo')

PL/SQL procedure successfully completed.

------------ Perform a query
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 1342139204

-----------------------------------
| Id  | Operation          | Name |
-----------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  SORT AGGREGATE    |      |
|   2 |   TABLE ACCESS FULL| FOO  |
-----------------------------------


------------ Give the optimizer some choice
SQL> create index i1 on foo(c1);

Index created.

SQL> exec dbms_stats.gather_table_stats(null,'foo')

PL/SQL procedure successfully completed.

------------ Run the query again
SQL> select sum(c2) from foo where c1 <= 100;
```

```
   SUM(C2)
----------
       100

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 2639956353

---------------------------------------------
| Id  | Operation                     | Name |
---------------------------------------------
|   0 | SELECT STATEMENT              |      |
|   1 |  SORT AGGREGATE               |      |
|   2 |   TABLE ACCESS BY INDEX ROWID| FOO  |
|   3 |    INDEX RANGE SCAN           | I1   |
---------------------------------------------


------------ Capture this plan in a stored outline
SQL> alter session set create_stored_outlines = Harald;

Session altered.

------------ Run the query again
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> alter session set create_stored_outlines = false;

Session altered.

------------ Succeeded?
SQL> select name,category,used from dba_outlines;

NAME                           CATEGORY                       USED
------------------------------ ------------------------------ ------
SYS_OUTLINE_09120516133528901  HARALD                         UNUSED

------------ Run the query again
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
```

```
       100

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 2639956353

-------------------------------------------
| Id  | Operation                    | Name |
-------------------------------------------
|   0 | SELECT STATEMENT             |      |
|   1 |  SORT AGGREGATE              |      |
|   2 |   TABLE ACCESS BY INDEX ROWID| FOO  |
|   3 |    INDEX RANGE SCAN          | I1   |
-------------------------------------------


------------ Why is the stored outline not used?
SQL> alter session set use_stored_outlines = Harald;

Session altered.

------------ Run the query again
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 2639956353

-------------------------------------------
| Id  | Operation                    | Name |
-------------------------------------------
|   0 | SELECT STATEMENT             |      |
|   1 |  SORT AGGREGATE              |      |
|   2 |   TABLE ACCESS BY INDEX ROWID| FOO  |
```

```
|   3 |    INDEX RANGE SCAN          | I1   |
---------------------------------------------

Note
-----
   - outline "SYS_OUTLINE_09120516133528901" used for this statement


------------ See the outline status
SQL> select name,category,used from dba_outlines;

NAME                           CATEGORY                       USED
------------------------------ ------------------------------ ------
SYS_OUTLINE_09120516133528901  HARALD                         USED

------------ Give the optimizer some more choice
SQL> create index i2 on foo(c1,c2);

Index created.

SQL> exec dbms_stats.gather_table_stats(null,'foo')

PL/SQL procedure successfully completed.

------------ Run the query again
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 2639956353

---------------------------------------------
| Id  | Operation                     | Name |
---------------------------------------------
|   0 | SELECT STATEMENT              |      |
|   1 |  SORT AGGREGATE               |      |
|   2 |   TABLE ACCESS BY INDEX ROWID| FOO  |
|   3 |    INDEX RANGE SCAN           | I1   |
---------------------------------------------

Note
-----
   - outline "SYS_OUTLINE_09120516133528901" used for this statement
```

```
------------ Create a new stored outline
SQL> create or replace outline for category Harald on
  2  select sum(c2) from foo where c1 <= 100;

Outline created.

------------ Run the query again
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 2639956353

---------------------------------------------
| Id  | Operation                     | Name |
---------------------------------------------
|   0 | SELECT STATEMENT              |      |
|   1 |  SORT AGGREGATE               |      |
|   2 |   TABLE ACCESS BY INDEX ROWID| FOO  |
|   3 |    INDEX RANGE SCAN           | I1   |
---------------------------------------------

Note
-----
   - outline "SYS_OUTLINE_09120516133528901" used for this statement


------------ Why did the plan not change?
SQL> alter session set use_stored_outlines = false;

Session altered.

------------ Try again to create a new stored outline
SQL> create or replace outline for category Harald on
  2  select sum(c2) from foo where c1 <= 100;

Outline created.

------------ Run the query again
SQL> alter session set use_stored_outlines = Harald;
```

```
Session altered.

SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

------------ See the execution plan used
SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 2583336616

-----------------------------------
| Id  | Operation         | Name |
-----------------------------------
|   0 | SELECT STATEMENT  |      |
|   1 |  SORT AGGREGATE   |      |
|   2 |   INDEX RANGE SCAN| I2   |
-----------------------------------

Note
-----
   - outline "SYS_OUTLINE_09120516133528901" used for this statement


------------ Drop all outlines in category Harald
SQL> exec dbms_outln.drop_by_cat('HARALD')

PL/SQL procedure successfully completed.

------------ Cleanup
SQL> set echo off

Table dropped.

Table dropped.
```

```
------------ SQL Plan Management
Connected.

PL/SQL procedure successfully completed.

------------ Create a table to play with
SQL> create table foo(c1 number(8), c2 number(8), c3 char(50));

Table created.

SQL> insert into foo select level, 1, 1 from dual connect by level <=
100000;

100000 rows created.

SQL> exec dbms_stats.gather_table_stats(null,'foo')

PL/SQL procedure successfully completed.

------------ Show the SQL Plan Management parameters
SQL> show parameter baseline

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------
-------
optimizer_capture_sql_plan_baselines boolean     FALSE
optimizer_use_sql_plan_baselines     boolean     TRUE
------------ Create our first SQL plan baseline
SQL> alter session set optimizer_capture_sql_plan_baselines = true;

Session altered.

------------ Perform a query
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> alter session set optimizer_capture_sql_plan_baselines = false;

Session altered.

------------ Succeeded?
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select sum(c2) from foo%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
```

```
SQL_PLAN_b6nynw6ffmn4b8e276424 AUTO-CAPTURE   YES YES NO              244

------------ See the complete DBA_SQL_PLAN_BASELINES view
SQL> desc dba_sql_plan_baselines
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
--------
 SIGNATURE                                 NOT NULL NUMBER
 SQL_HANDLE                                NOT NULL VARCHAR2(30)
 SQL_TEXT                                  NOT NULL CLOB
 PLAN_NAME                                 NOT NULL VARCHAR2(30)
 CREATOR                                            VARCHAR2(30)
 ORIGIN                                             VARCHAR2(14)
 PARSING_SCHEMA_NAME                                VARCHAR2(30)
 DESCRIPTION                                        VARCHAR2(500)
 VERSION                                            VARCHAR2(64)
 CREATED                                   NOT NULL TIMESTAMP(6)
 LAST_MODIFIED                                      TIMESTAMP(6)
 LAST_EXECUTED                                      TIMESTAMP(6)
 LAST_VERIFIED                                      TIMESTAMP(6)
 ENABLED                                            VARCHAR2(3)
 ACCEPTED                                           VARCHAR2(3)
 FIXED                                              VARCHAR2(3)
 AUTOPURGE                                          VARCHAR2(3)
 OPTIMIZER_COST                                     NUMBER
 MODULE                                             VARCHAR2(48)
 ACTION                                             VARCHAR2(32)
 EXECUTIONS                                         NUMBER
 ELAPSED_TIME                                       NUMBER
 CPU_TIME                                           NUMBER
 BUFFER_GETS                                        NUMBER
 DISK_READS                                         NUMBER
 DIRECT_WRITES                                      NUMBER
 ROWS_PROCESSED                                     NUMBER
 FETCHES                                            NUMBER
 END_OF_FETCH_COUNT                                 NUMBER

SQL> -- Note: The last executed timestamp will only be updated once per
week
------------ Show the captured execution plan
SQL> select sql_handle from dba_sql_plan_baselines
  2  where sql_text like 'select sum(c2) from foo%';

SQL_HANDLE
------------------------------
SYS_SQL_b353d4e19ce9d08b

------------ Press ENTER to continue.
SQL> select * from table(
  2  dbms_xplan.display_sql_plan_baseline(sql_handle=>'&sql_handle',
  3  format=>'basic +cost')
  4  );

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
```

```
--------------------------------------------------------------------------------
--------
SQL handle: SYS_SQL_b353d4e19ce9d08b
SQL text: select sum(c2) from foo where c1 <= 100
--------------------------------------------------------------------------------
--------

--------------------------------------------------------------------------------
--------
Plan name: SQL_PLAN_b6nynw6ffmn4b8e276424         Plan id: 2384946212
Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
--------------------------------------------------------------------------------
--------

Plan hash value: 1342139204

-----------------------------------------------
| Id  | Operation          | Name | Cost (%CPU)|
-----------------------------------------------
|   0 | SELECT STATEMENT   |      |   244   (1)|
|   1 |  SORT AGGREGATE    |      |            |
|   2 |   TABLE ACCESS FULL| FOO  |   244   (1)|
-----------------------------------------------

------------ Run the query again and show the execution plan
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 1342139204

-----------------------------------
| Id  | Operation          | Name |
-----------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  SORT AGGREGATE    |      |
|   2 |   TABLE ACCESS FULL| FOO  |
-----------------------------------

Note
-----
   - SQL plan baseline SQL_PLAN_b6nynw6ffmn4b8e276424 used for this
statement
```

```
------------ Give the optimizer some choice
SQL> create index i1 on foo(c1);

Index created.

SQL> exec dbms_stats.gather_table_stats(null,'foo')

PL/SQL procedure successfully completed.

------------ Run the query again and show the execution plan
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
--------
SQL_ID  f7xrmkmxmcnww, child number 5

An uncaught error happened in prepare_sql_statement : ORA-01403: no data
found

NOTE: cannot fetch plan for SQL_ID: f7xrmkmxmcnww, CHILD_NUMBER: 5
      Please verify value of SQL_ID and CHILD_NUMBER;
      It could also be that the plan is no longer in cursor cache (check
v$sql_p
lan)



------------ Huh?
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 1342139204
```

```
---------------------------------
| Id  | Operation          | Name |
---------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  SORT AGGREGATE    |      |
|   2 |   TABLE ACCESS FULL| FOO  |
---------------------------------

Note
-----
   - SQL plan baseline SQL_PLAN_b6nynw6ffmn4b8e276424 used for this
statement


------------ Have a look in the SMB
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select sum(c2) from foo%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_b6nynw6ffmn4b8e276424 AUTO-CAPTURE   YES YES NO             244
SQL_PLAN_b6nynw6ffmn4be47a8c11 AUTO-CAPTURE   YES NO  NO               3

SQL> -- Note: We did not set the capture parameter to true to catch the
2nd plan
------------ And show their execution plans
SQL> select * from table(
  2  dbms_xplan.display_sql_plan_baseline(sql_handle=>'&sql_handle',
  3  format=>'basic +cost')
  4  );

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
--------

--------------------------------------------------------------------------
--------
SQL handle: SYS_SQL_b353d4e19ce9d08b
SQL text: select sum(c2) from foo where c1 <= 100
--------------------------------------------------------------------------
--------

--------------------------------------------------------------------------
--------
Plan name: SQL_PLAN_b6nynw6ffmn4b8e276424         Plan id: 2384946212
Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
--------------------------------------------------------------------------
--------

Plan hash value: 1342139204

-----------------------------------------------
| Id  | Operation          | Name | Cost (%CPU)|
-----------------------------------------------
|   0 | SELECT STATEMENT   |      |   244   (1)|
```

```
|   1 |  SORT AGGREGATE    |      |            |
|   2 |   TABLE ACCESS FULL| FOO  |   244   (1)|
-----------------------------------------------

-------------------------------------------------------------------------
--------
Plan name: SQL_PLAN_b6nynw6ffmn4be47a8c11         Plan id: 3833236497
Enabled: YES     Fixed: NO      Accepted: NO      Origin: AUTO-CAPTURE
-------------------------------------------------------------------------
--------

Plan hash value: 2639956353

-----------------------------------------------------------
| Id  | Operation                    | Name | Cost (%CPU)|
-----------------------------------------------------------
|   0 | SELECT STATEMENT             |      |     3   (0)|
|   1 |  SORT AGGREGATE              |      |            |
|   2 |   TABLE ACCESS BY INDEX ROWID| FOO  |     3   (0)|
|   3 |    INDEX RANGE SCAN          | I1   |     2   (0)|
-----------------------------------------------------------

------------ Evolve the second plan
SQL> select dbms_spm.evolve_sql_plan_baseline(NULL,'&plan_name') from
dual;

DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(NULL,'SQL_PLAN_B6NYNW6FFMN4BE47A8C11')
-------------------------------------------------------------------------
--------

-------------------------------------------------------------------------
-------
                        Evolve SQL Plan Baseline Report
-------------------------------------------------------------------------
-------

Inputs:
-------
  SQL_HANDLE =
  PLAN_NAME  = SQL_PLAN_b6nynw6ffmn4be47a8c11
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = YES
  COMMIT     = YES

Plan: SQL_PLAN_b6nynw6ffmn4be47a8c11
------------------------------------
  Plan was verified: Time used .06 seconds.
  Plan passed performance criterion: 286.86 times better than baseline
plan.
  Plan was changed to an accepted plan.

                            Baseline Plan      Test Plan       Stats
Ratio
                            -------------      ---------       ---------
--
  Execution Status:              COMPLETE       COMPLETE
```

```
  Rows Processed:                         1              1
  Elapsed Time(ms):                   4.109           .134
30.66
  CPU Time(ms):                       4.444          1.111
4
  Buffer Gets:                          892              3
297.33
  Physical Read Requests:                 0              0
  Physical Write Requests:                0              0
  Physical Read Bytes:                    0              0
  Physical Write Bytes:                   0              0
  Executions:                             1              1

-------------------------------------------------------------------------
-------
                                  Report Summary
-------------------------------------------------------------------------
-------
Number of plans verified: 1
Number of plans accepted: 1


------------ check the dictionary
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select sum(c2) from foo%';

PLAN_NAME                      ORIGIN          ENA ACC FIX OPTIMIZER_COST
------------------------------ --------------- --- --- --- --------------
SQL_PLAN_b6nynw6ffmn4b8e276424 AUTO-CAPTURE    YES YES NO             244
SQL_PLAN_b6nynw6ffmn4be47a8c11 AUTO-CAPTURE    YES YES NO               3

------------ Run the query again and show the execution plan
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------
--------
SQL_ID  f7xrmkmxmcnww, child number 4

An uncaught error happened in prepare_sql_statement : ORA-01403: no data
found

NOTE: cannot fetch plan for SQL_ID: f7xrmkmxmcnww, CHILD_NUMBER: 4
      Please verify value of SQL_ID and CHILD_NUMBER;
      It could also be that the plan is no longer in cursor cache (check
v$sql_p
lan)
```

```
------------ Huh?
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 2639956353

---------------------------------------------
| Id  | Operation                    | Name |
---------------------------------------------
|   0 | SELECT STATEMENT             |      |
|   1 |  SORT AGGREGATE              |      |
|   2 |   TABLE ACCESS BY INDEX ROWID| FOO  |
|   3 |    INDEX RANGE SCAN          | I1   |
---------------------------------------------

Note
-----
   - SQL plan baseline SQL_PLAN_b6nynw6ffmn4be47a8c11 used for this
statement


------------ See what happens after changing the FTS plan to FIXED
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select sum(c2) from foo%' and OPTIMIZER_COST >
100;

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_b6nynw6ffmn4b8e276424 AUTO-CAPTURE   YES YES NO             244

SQL> exec :cnt := dbms_spm.alter_sql_plan_baseline( -
>                 plan_name         => '&plan_name', -
>                 attribute_name    => 'FIXED', -
>                 attribute_value   => 'YES');

PL/SQL procedure successfully completed.

------------ Have a look in the SMB
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
```

```
  3  where sql_text like 'select sum(c2) from foo%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_b6nynw6ffmn4b8e276424 AUTO-CAPTURE   YES YES YES             244
SQL_PLAN_b6nynw6ffmn4be47a8c11 AUTO-CAPTURE   YES YES NO                3

------------ Run the query again and show the execution plan
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 1342139204

-----------------------------------
| Id  | Operation          | Name |
-----------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  SORT AGGREGATE    |      |
|   2 |   TABLE ACCESS FULL| FOO  |
-----------------------------------

Note
-----
   - SQL plan baseline SQL_PLAN_b6nynw6ffmn4b8e276424 used for this
statement


------------ Give the optimizer some more choice
SQL> create index i2 on foo(c1,c2);

Index created.

SQL> exec dbms_stats.gather_table_stats(null,'foo')

PL/SQL procedure successfully completed.

------------ Run the query again and show the execution plan
SQL> select sum(c2) from foo where c1 <= 100;

   SUM(C2)
----------
       100
```

```
SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select sum(c2) from foo where c1 <= 100

Plan hash value: 1342139204

-----------------------------------
| Id  | Operation          | Name |
-----------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  SORT AGGREGATE    |      |
|   2 |   TABLE ACCESS FULL| FOO  |
-----------------------------------

Note
-----
   - SQL plan baseline SQL_PLAN_b6nynw6ffmn4b8e276424 used for this
statement


------------ Have a look in the SMB
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select sum(c2) from foo%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_b6nynw6ffmn4b8e276424 AUTO-CAPTURE   YES YES YES            244
SQL_PLAN_b6nynw6ffmn4be47a8c11 AUTO-CAPTURE   YES YES NO               3

------------ Configure the SMB
SQL> exec dbms_spm.configure('space_budget_percent',10)

PL/SQL procedure successfully completed.

SQL> exec dbms_spm.configure('plan_retention_weeks',53)

PL/SQL procedure successfully completed.

------------ Show this in the dictionary
SQL> select * from dba_sql_management_config;

PARAMETER_NAME       PARAMETER_VALUE LAST_MODIFIED
MODIFIED_B
-------------------- --------------- --------------------------- ------
----
SPACE_BUDGET_PERCENT              10 05-DEC-09 04.14.30.000000 PM SYSTEM
PLAN_RETENTION_WEEKS              53 05-DEC-09 04.14.30.000000 PM SYSTEM

------------ Create a SMB staging table
```

```
SQL> exec DBMS_SPM.CREATE_STGTAB_BASELINE('MYSMB')

PL/SQL procedure successfully completed.

------------ See it
SQL> desc mysmb
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
--------
 VERSION                                            NUMBER
 SIGNATURE                                          NUMBER
 SQL_HANDLE                                         VARCHAR2(30)
 OBJ_NAME                                           VARCHAR2(30)
 OBJ_TYPE                                           VARCHAR2(30)
 PLAN_ID                                            NUMBER
 SQL_TEXT                                           CLOB
 CREATOR                                            VARCHAR2(30)
 ORIGIN                                             VARCHAR2(30)
 DESCRIPTION                                        VARCHAR2(500)
 DB_VERSION                                         VARCHAR2(64)
 CREATED                                            TIMESTAMP(6)
 LAST_MODIFIED                                      TIMESTAMP(6)
 LAST_EXECUTED                                      TIMESTAMP(6)
 LAST_VERIFIED                                      TIMESTAMP(6)
 STATUS                                             NUMBER
 OPTIMIZER_COST                                     NUMBER
 MODULE                                             VARCHAR2(48)
 ACTION                                             VARCHAR2(32)
 EXECUTIONS                                         NUMBER
 ELAPSED_TIME                                       NUMBER
 CPU_TIME                                           NUMBER
 BUFFER_GETS                                        NUMBER
 DISK_READS                                         NUMBER
 DIRECT_WRITES                                      NUMBER
 ROWS_PROCESSED                                     NUMBER
 FETCHES                                            NUMBER
 END_OF_FETCH_COUNT                                 NUMBER
 CATEGORY                                           VARCHAR2(30)
 SQLFLAGS                                           NUMBER
 TASK_ID                                            NUMBER
 TASK_EXEC_NAME                                     VARCHAR2(30)
 TASK_OBJ_ID                                        NUMBER
 TASK_FND_ID                                        NUMBER
 TASK_REC_ID                                        NUMBER
 INUSE_FEATURES                                     NUMBER
 PARSE_CPU_TIME                                     NUMBER
 PRIORITY                                           NUMBER
 OPTIMIZER_ENV                                      RAW(2000)
 BIND_DATA                                          RAW(2000)
 PARSING_SCHEMA_NAME                                VARCHAR2(30)
 COMP_DATA                                          CLOB

------------ Export the SMB to the staging table
SQL> exec :cnt := DBMS_SPM.PACK_STGTAB_BASELINE('MYSMB')

PL/SQL procedure successfully completed.
```

```
SQL> print :cnt

       CNT
----------
         2

------------ Create another table to play with
SQL> create table bar(c1 number(8), c2 number(8), c3 char(50));

Table created.

SQL> insert into bar select level, 1, 1 from dual connect by level <=
100;

100 rows created.

SQL> insert into bar select * from bar;

100 rows created.

SQL> /

200 rows created.

SQL> exec dbms_stats.gather_table_stats(null,'bar')

PL/SQL procedure successfully completed.

------------ Perform a query with different optimizer modes
SQL> alter session set optimizer_mode = first_rows_1;

Session altered.

SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

------------ Get the SQL_ID, switch optimizer mode and exec again
SQL> select prev_sql_id SQL_ID from v$session where sid =
sys_context('userenv', 'SID');

SQL_ID
-------------
1bcpkywprtdrh

SQL> alter session set optimizer_mode = all_rows;

Session altered.
```

```
SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

------------ Load their execution plans into the SMB
SQL> exec :cnt := dbms_spm.load_plans_from_cursor_cache('&sql_id')

PL/SQL procedure successfully completed.

SQL> print :cnt

       CNT
----------
         2

------------ See them in the SMB
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_7zqfaxj4fd787e6a36028 MANUAL-LOAD    YES YES NO               4
SQL_PLAN_7zqfaxj4fd787fc9e05d1 MANUAL-LOAD    YES YES NO               6

------------ Exec again and see the plan
SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
--------
SQL_ID  1bcpkywprtdrh, child number 2

An uncaught error happened in prepare_sql_statement : ORA-01403: no data
found

NOTE: cannot fetch plan for SQL_ID: 1bcpkywprtdrh, CHILD_NUMBER: 2
      Please verify value of SQL_ID and CHILD_NUMBER;
```

```
      It could also be that the plan is no longer in cursor cache (check
v$sql_p
lan)



SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and f.c1=1

Plan hash value: 1207490703

-----------------------------------
| Id  | Operation          | Name |
-----------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  HASH JOIN         |      |
|   2 |   INDEX RANGE SCAN | I2   |
|   3 |   TABLE ACCESS FULL| BAR  |
-----------------------------------

Note
-----
   - SQL plan baseline SQL_PLAN_7zqfaxj4fd787fc9e05d1 used for this
statement


------------ What happens if we switch optimizer mode?
SQL> alter session set optimizer_mode = first_rows_1;

Session altered.

SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
```

```
         1          1          1

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
--------
SQL_ID  1bcpkywprtdrh, child number 3

An uncaught error happened in prepare_sql_statement : ORA-01403: no data
found

NOTE: cannot fetch plan for SQL_ID: 1bcpkywprtdrh, CHILD_NUMBER: 3
      Please verify value of SQL_ID and CHILD_NUMBER;
      It could also be that the plan is no longer in cursor cache (check
v$sql_p
lan)



SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> select * from table(dbms_xplan.display_cursor(format => 'basic
note'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
--------
EXPLAINED SQL STATEMENT:
------------------------
select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and f.c1=1

Plan hash value: 3834250642

-----------------------------------
| Id  | Operation          | Name |
-----------------------------------
|   0 | SELECT STATEMENT   |      |
|   1 |  NESTED LOOPS      |      |
|   2 |   INDEX RANGE SCAN | I2   |
|   3 |   TABLE ACCESS FULL| BAR  |
-----------------------------------

Note
-----
   - SQL plan baseline SQL_PLAN_7zqfaxj4fd787e6a36028 used for this
statement
```

```
------------ Empty the SMB
SQL> declare
  2          cnt number;
  3  begin
  4          for r_sql in ( select distinct sql_handle
  5                  from dba_sql_plan_baselines)
  6          loop
  7                  begin
  8                          cnt :=
dbms_spm.drop_sql_plan_baseline(r_sql.sql_handle);
  9                  exception
 10                  when others
 11                          then
 12                                  dbms_output.put_line( 'Problem
dropping: ' ||
 13                                  r_sql.sql_handle );
 14                  end;
 15          end loop;
 16  end;
 17  /

PL/SQL procedure successfully completed.

------------ Create some new SQL plan baselines
SQL> alter session set optimizer_mode = all_rows;

Session altered.

SQL> drop index i1;

Index dropped.

SQL> drop index i2;

Index dropped.

SQL> alter session set optimizer_capture_sql_plan_baselines = true;

Session altered.

SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> alter session set optimizer_capture_sql_plan_baselines = false;

Session altered.
```

```
SQL> create index i2 on foo(c1,c2);

Index created.

SQL> exec dbms_stats.gather_table_stats(null,'foo')


PL/SQL procedure successfully completed.

SQL> --create index b1 on bar(c1);
SQL> --exec dbms_stats.gather_table_stats(null,'bar')
SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> /

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> /

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> /

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> /

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1
```

```
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_7zqfaxj4fd787f02813bf AUTO-CAPTURE   YES YES NO             248
SQL_PLAN_7zqfaxj4fd787fc9e05d1 AUTO-CAPTURE   YES NO  NO               6

SQL> select name from dba_sql_profilesvailable?
  2  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

no rows selected

------------ Before running the SQL tuning advisor enable profile
acceptance
SQL> @sysdba
SQL> connect / as sysdba
Connected.
SQL> exec dbms_workload_repository.create_snapshot

PL/SQL procedure successfully completed.

SQL> exec  dbms_sqltune.reset_tuning_task('SYS_AUTO_SQL_TUNING_TASK')

PL/SQL procedure successfully completed.

SQL> exec
dbms_sqltune.set_tuning_task_parameter('SYS_AUTO_SQL_TUNING_TASK','ACCEP
T_SQL_PROFILES','true')

PL/SQL procedure successfully completed.

------------ See this in the dictionary
SQL> col parameter_name for a30
SQL> col parameter_value for a30
SQL> select parameter_name,parameter_value from DBA_ADVISOR_PARAMETERS
  2  where parameter_name like '%PROF%';

PARAMETER_NAME                 PARAMETER_VALUE
------------------------------ ------------------------------
MAX_AUTO_SQL_PROFILES          10000
MAX_SQL_PROFILES_PER_EXEC      20
ACCEPT_SQL_PROFILES            true

------------ Now run the SQL Tuning advisor
SQL> exec  dbms_sqltune.execute_tuning_task('SYS_AUTO_SQL_TUNING_TASK')

PL/SQL procedure successfully completed.

------------ See the SQL tuning advisor report
SQL> select dbms_sqltune.report_auto_tuning_task from dual;

REPORT_AUTO_TUNING_TASK
```

```
-------------------------------------------------------------------------------
--------
GENERAL INFORMATION SECTION
-------------------------------------------------------------------------------
-------
Tuning Task Name                          : SYS_AUTO_SQL_TUNING_TASK
Tuning Task Owner                         : SYS
Workload Type                             : Automatic High-Load SQL
Workload
Scope                                     : COMPREHENSIVE
Global Time Limit(seconds)                : 3600
Per-SQL Time Limit(seconds)               : 1200
Completion Status                         : COMPLETED
Started at                                : 12/05/2009 16:14:49
Completed at                              : 12/05/2009 16:14:52
Number of Candidate SQLs                  : 7
Cumulative Elapsed Time of SQL (s)        : 32

-------------------------------------------------------------------------------
-------
SUMMARY SECTION
-------------------------------------------------------------------------------
-------
                      Global SQL Tuning Result Statistics
-------------------------------------------------------------------------------
-------
Number of SQLs Analyzed                      : 7
Number of SQLs in the Report                 : 2
Number of SQLs with Findings                 : 2
Number of SQLs with Alternative Plan Findings: 2
Number of SQLs with SQL profiles implemented : 1

-------------------------------------------------------------------------------
-------
    SQLs with Findings Ordered by Maximum (Profile/Index) Benefit,
Object ID
-------------------------------------------------------------------------------
-------
object ID  SQL ID        statistics profile(benefit) index(benefit)
restructure
---------- ------------- ---------- --------------- -------------- ----
-------
         6 1bcpkywprtdrh                      99.13%
         9 f7xrmkmxmcnww

-------------------------------------------------------------------------------
-------
DETAILS SECTION
-------------------------------------------------------------------------------
-------
 Statements with Results Ordered by Maximum (Profile/Index) Benefit,
Object ID
-------------------------------------------------------------------------------
-------
Object ID  : 6
Schema Name: SYSTEM
```

```
SQL ID      : 1bcpkywprtdrh
SQL Text    : select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1
and
              f.c1=1

--------------------------------------------------------------------------
-------
FINDINGS SECTION (2 findings)
--------------------------------------------------------------------------
-------

1- SQL Profile Finding (see explain plans section below)
--------------------------------------------------------
  A potentially better execution plan was found for this statement.
  SQL profile "SYS_SQLPROF_01255f6859f50000" and SQL plan baseline
  "SQL_PLAN_7zqfaxj4fd787fc9e05d1" were created automatically for this
  statement.

  Recommendation (estimated benefit: 99.13%)
  ------------------------------------------
  - An automatically-created SQL profile is present on the system.
    Name:   SYS_SQLPROF_01255f6859f50000
    Status: ENABLED

  Validation results
  ------------------
  The SQL profile was tested by executing both its plan and the original
plan
  and measuring their respective execution statistics. A plan may have
been
  only partially executed if the other could be run to completion in
less time.

                          Original Plan  With SQL Profile  % Improved
                          -------------  ----------------  ----------
  Completion Status:           COMPLETE          COMPLETE
  Elapsed Time(us):               3721               147      96.04 %
  CPU Time(us):                   4000                 0        100 %
  User I/O Time(us):                 0                 0
  Buffer Gets:                     808                 7      99.13 %
  Physical Read Requests:            0                 0
  Physical Write Requests:           0                 0
  Physical Read Bytes:               0                 0
  Physical Write Bytes:              0                 0
  Rows Processed:                    4                 4
  Fetches:                           4                 4
  Executions:                        1                 1

  Notes
  -----
  1. The original plan was first executed to warm the buffer cache.
  2. Statistics for original plan were averaged over next 9 executions.
  3. The SQL profile plan was first executed to warm the buffer cache.
  4. Statistics for the SQL profile plan were averaged over next 9
executions.
```

```
2- Alternative Plan Finding
---------------------------
  Some alternative execution plans for this statement were found by
searching
  the system's real-time and historical performance data.

  The following table lists these plans ranked by their average elapsed
time.
  See section "ALTERNATIVE PLANS SECTION" for detailed information on
each
  plan.

  id plan hash  last seen            elapsed (s)  origin          note

  -- ---------- -------------------- ------------ --------------- ------
--------
--
   1 3834250642  2009-12-05/16:14:34        0.002 Cursor Cache

   2 1207490703  2009-12-05/16:14:35        0.003 Cursor Cache

   3 2094252384  2009-12-05/16:14:42        0.005 Cursor Cache
original plan

   4 2194434350  2009-12-05/15:59:47        0.147 AWR


  Recommendation
  --------------
  - Consider creating a SQL plan baseline for the plan with the best
average
    elapsed time.
    execute dbms_sqltune.create_sql_plan_baseline(task_name =>
            'SYS_AUTO_SQL_TUNING_TASK', object_id => 6, plan_hash_value
=>
            3834250642);

-------------------------------------------------------------------------
-------
EXPLAIN PLANS SECTION
-------------------------------------------------------------------------
-------

1- Original With Adjusted Cost
------------------------------
Plan hash value: 2094252384

-------------------------------------------------------------------------
---
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time
|
-------------------------------------------------------------------------
---
|   0 | SELECT STATEMENT   |      |     4 |    56 |   248   (2)|
00:00:03 |
```

```
|*  1 |  HASH JOIN         |      |     4 |    56 |   248   (2)|
00:00:03 |
|*  2 |   TABLE ACCESS FULL| FOO  |     1 |     8 |   244   (1)|
00:00:03 |
|*  3 |   TABLE ACCESS FULL| BAR  |     4 |    24 |     3   (0)|
00:00:01 |
--------------------------------------------------------------------------
---

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

   1 - access("F"."C1"="B"."C1")
   2 - filter("F"."C1"=1)
   3 - filter("B"."C1"=1)

2- Using SQL Profile
--------------------
Plan hash value: 1207490703

--------------------------------------------------------------------------
---
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time
|
--------------------------------------------------------------------------
---
|   0 | SELECT STATEMENT   |      |     4 |    56 |     6  (17)|
00:00:01 |
|*  1 |  HASH JOIN         |      |     4 |    56 |     6  (17)|
00:00:01 |
|*  2 |   INDEX RANGE SCAN | I2   |     1 |     8 |     2   (0)|
00:00:01 |
|*  3 |   TABLE ACCESS FULL| BAR  |     4 |    24 |     3   (0)|
00:00:01 |
--------------------------------------------------------------------------
---

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

   1 - access("F"."C1"="B"."C1")
   2 - access("F"."C1"=1)
   3 - filter("B"."C1"=1)

--------------------------------------------------------------------------
-------
ALTERNATIVE PLANS SECTION
--------------------------------------------------------------------------
-------

Plan 2
------

  Plan Origin                 :Cursor Cache
  Plan Hash Value             :1207490703
  Executions                  :1
```

```
  Elapsed Time                :0.003 sec
  CPU Time                    :0.000 sec
  Buffer Gets                 :9
  Disk Reads                  :0
  Disk Writes                 :0

Notes:
  1. Statistics shown are averaged over multiple executions.

-------------------------------------------------------------------------
---
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time
|
-------------------------------------------------------------------------
---
|   0 | SELECT STATEMENT   |      |     4 |    56 |     6  (17)|
00:00:01 |
|*  1 |  HASH JOIN         |      |     4 |    56 |     6  (17)|
00:00:01 |
|*  2 |   INDEX RANGE SCAN | I2   |     1 |     8 |     2   (0)|
00:00:01 |
|*  3 |   TABLE ACCESS FULL| BAR  |     4 |    24 |     3   (0)|
00:00:01 |
-------------------------------------------------------------------------
---

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

   1 - access("F"."C1"="B"."C1")
   2 - access("F"."C1"=1)
   3 - filter("B"."C1"=1)

Plan 1
------

  Plan Origin                 :Cursor Cache
  Plan Hash Value             :3834250642
  Executions                  :1
  Elapsed Time                :0.002 sec
  CPU Time                    :0.000 sec
  Buffer Gets                 :10
  Disk Reads                  :0
  Disk Writes                 :0

Notes:
  1. Statistics shown are averaged over multiple executions.

-------------------------------------------------------------------------
---
------------ See the SQL plan baselines
SQL> @system
SQL> connect system/oracle
Connected.
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
```

```
  3  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_7zqfaxj4fd787f02813bf AUTO-CAPTURE   YES YES NO             248
SQL_PLAN_7zqfaxj4fd787fc9e05d1 AUTO-CAPTURE   YES YES NO               6

------------ See the created profile
SQL> select name from dba_sql_profiles
  2  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

NAME
------------------------------
SYS_SQLPROF_01255f6859f50000

------------ Delete the SQL plan baselines
SQL> select sql_handle from dba_sql_plan_baselines
  2  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

SQL_HANDLE
------------------------------
SYS_SQL_7fd9caec48e69d07
SYS_SQL_7fd9caec48e69d07

SQL> exec :cnt := dbms_spm.drop_sql_plan_baseline('&sql_handle')

PL/SQL procedure successfully completed.

SQL> print :cnt

       CNT
----------
         2

------------ Drop the SQL profile
SQL> exec dbms_sqltune.drop_sql_profile('&name')

PL/SQL procedure successfully completed.

------------ Create some new SQL plan baselines
SQL> alter session set optimizer_mode = first_rows_1;

Session altered.

SQL> drop index i2;

Index dropped.

SQL> alter session set optimizer_capture_sql_plan_baselines = true;

Session altered.

SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
```

```
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> alter session set optimizer_capture_sql_plan_baselines = false;

Session altered.

SQL> create index i1 on foo(c1);

Index created.

SQL> exec dbms_stats.gather_table_stats(null,'foo')

PL/SQL procedure successfully completed.

SQL> select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1 and
f.c1=1;

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> /

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> /

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> /

        C1         C2         C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> /
```

```
        C1          C2          C2
---------- ---------- ----------
         1          1          1
         1          1          1
         1          1          1
         1          1          1

SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_7zqfaxj4fd78743ac900d AUTO-CAPTURE   YES NO  NO               4
SQL_PLAN_7zqfaxj4fd787e6003f7b AUTO-CAPTURE   YES YES NO             125

------------ Now run the SQL Tuning advisor
SQL> @sysdba
SQL> connect / as sysdba
Connected.
SQL> exec dbms_workload_repository.create_snapshot

PL/SQL procedure successfully completed.

SQL> exec  dbms_sqltune.reset_tuning_task('SYS_AUTO_SQL_TUNING_TASK')

PL/SQL procedure successfully completed.

SQL> exec  dbms_sqltune.execute_tuning_task('SYS_AUTO_SQL_TUNING_TASK')

PL/SQL procedure successfully completed.

------------ See the SQL tuning advisor report
SQL> select dbms_sqltune.report_auto_tuning_task from dual;

REPORT_AUTO_TUNING_TASK
--------------------------------------------------------------------------------
--------
GENERAL INFORMATION SECTION
--------------------------------------------------------------------------------
-------
Tuning Task Name                          : SYS_AUTO_SQL_TUNING_TASK
Tuning Task Owner                         : SYS
Workload Type                             : Automatic High-Load SQL
Workload
Scope                                     : COMPREHENSIVE
Global Time Limit(seconds)                : 3600
Per-SQL Time Limit(seconds)               : 1200
Completion Status                         : COMPLETED
Started at                                : 12/05/2009 16:15:08
Completed at                              : 12/05/2009 16:15:11
Number of Candidate SQLs                  : 7
Cumulative Elapsed Time of SQL (s)        : 35
```

```
--------------------------------------------------------------------------------
-------
SUMMARY SECTION
--------------------------------------------------------------------------------
-------
                      Global SQL Tuning Result Statistics
--------------------------------------------------------------------------------
-------
Number of SQLs Analyzed                       : 7
Number of SQLs in the Report                  : 2
Number of SQLs with Findings                  : 2
Number of SQLs with Alternative Plan Findings: 2
Number of SQLs with SQL profiles implemented : 1

--------------------------------------------------------------------------------
-------
    SQLs with Findings Ordered by Maximum (Profile/Index) Benefit,
Object ID
--------------------------------------------------------------------------------
-------
object ID  SQL ID        statistics profile(benefit) index(benefit)
restructure
---------- ------------- ---------- ---------------- -------------- ----
-------
         6 1bcpkywprtdrh                      99.01%
         9 f7xrmkmxmcnww

--------------------------------------------------------------------------------
-------
DETAILS SECTION
--------------------------------------------------------------------------------
-------
 Statements with Results Ordered by Maximum (Profile/Index) Benefit,
Object ID
--------------------------------------------------------------------------------
-------
Object ID  : 6
Schema Name: SYSTEM
SQL ID     : 1bcpkywprtdrh
SQL Text   : select f.c1,f.c2,b.c2 from foo f, bar b where f.c1 = b.c1
and
             f.c1=1

--------------------------------------------------------------------------------
-------
FINDINGS SECTION (2 findings)
--------------------------------------------------------------------------------
-------

1- SQL Profile Finding (see explain plans section below)
--------------------------------------------------------
  A potentially better execution plan was found for this statement.
  SQL profile "SYS_SQLPROF_01255f68a3330001" and SQL plan baseline
  "SQL_PLAN_7zqfaxj4fd787e53205ee" were created automatically for this
  statement.
```

```
  Recommendation (estimated benefit: 99.01%)
  ------------------------------------------
  - An automatically-created SQL profile is present on the system.
    Name:   SYS_SQLPROF_01255f68a3330001
    Status: ENABLED

  Validation results
  ------------------
  The SQL profile was tested by executing both its plan and the original
plan
  and measuring their respective execution statistics. A plan may have
been
  only partially executed if the other could be run to completion in
less time.

                          Original Plan  With SQL Profile  % Improved
                          -------------  ----------------  ----------
  Completion Status:           COMPLETE          COMPLETE
  Elapsed Time(us):               5884               155      97.36 %
  CPU Time(us):                   6000                 0        100 %
  User I/O Time(us):                 0                 0
  Buffer Gets:                     808                 8         99 %
  Physical Read Requests:            0                 0
  Physical Write Requests:           0                 0
  Physical Read Bytes:               0                 0
  Physical Write Bytes:              0                 0
  Rows Processed:                    4                 4
  Fetches:                           4                 4
  Executions:                        1                 1

  Notes
  -----
  1. The original plan was first executed to warm the buffer cache.
  2. Statistics for original plan were averaged over next 9 executions.
  3. The SQL profile plan was first executed to warm the buffer cache.
  4. Statistics for the SQL profile plan were averaged over next 9
executions.

2- Alternative Plan Finding
---------------------------
  Some alternative execution plans for this statement were found by
searching
  the system's real-time and historical performance data.

  The following table lists these plans ranked by their average elapsed
time.
  See section "ALTERNATIVE PLANS SECTION" for detailed information on
each
  plan.

  id plan hash  last seen            elapsed (s)  origin          note

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

```
   1 3834250642  2009-12-05/16:14:34        0.002 Cursor Cache    not
reproducib
le
   2 1207490703  2009-12-05/16:14:35        0.003 Cursor Cache    not
reproducib
le
   3 2194434350  2009-12-05/16:15:06        0.006 Cursor Cache
original plan

   4 2094252384  2009-12-05/16:13:57        0.007 AWR


  Information
  -----------
  - All alternative plans other than the Original Plan could not be
    reproduced in the current environment.
  - The plan with id 1 could not be reproduced in the current
environment.
    For this reason, a SQL plan baseline cannot be created to instruct
the
    Oracle optimizer to pick this plan in the future.
  - The plan with id 2 could not be reproduced in the current
environment.
    For this reason, a SQL plan baseline cannot be created to instruct
the
    Oracle optimizer to pick this plan in the future.

--------------------------------------------------------------------------
-------
EXPLAIN PLANS SECTION
--------------------------------------------------------------------------
-------

1- Original With Adjusted Cost
------------------------------
Plan hash value: 2194434350

--------------------------------------------------------------------------
----
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time
|
--------------------------------------------------------------------------
----
|   0 | SELECT STATEMENT    |      |     2 |    40 |   247   (2)|
00:00:03 |
|   1 |  MERGE JOIN         |      |     2 |    40 |   247   (2)|
00:00:03 |
|*  2 |   TABLE ACCESS FULL | BAR  |     2 |    12 |     2   (0)|
00:00:01 |
|*  3 |   SORT JOIN         |      |     1 |     8 |   245   (2)|
00:00:03 |
|*  4 |    TABLE ACCESS FULL| FOO  |     1 |     8 |   244   (1)|
00:00:03 |
--------------------------------------------------------------------------
----
```

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

   2 - filter("B"."C1"=1)
   3 - access("F"."C1"="B"."C1")
       filter("F"."C1"="B"."C1")
   4 - filter("F"."C1"=1)

2- Using SQL Profile
--------------------
Plan hash value: 2556729538

-------------------------------------------------------------------------------
--------
-----
| Id  | Operation                     | Name | Rows  | Bytes | Cost
(%CPU)| Time
    |
-------------------------------------------------------------------------------
--------
-----
|   0 | SELECT STATEMENT              |      |     4 |    56 |     6
(17)| 00:00
:01 |
|*  1 |  HASH JOIN                    |      |     4 |    56 |     6
(17)| 00:00
:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| FOO  |     1 |     8 |     2
(0)| 00:00
:01 |
|*  3 |    INDEX RANGE SCAN           | I1   |     1 |       |     1
(0)| 00:00
:01 |
|*  4 |   TABLE ACCESS FULL           | BAR  |     4 |    24 |     3
(0)| 00:00
:01 |
-------------------------------------------------------------------------------
--------
-----

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

   1 - access("F"."C1"="B"."C1")
   3 - access("F"."C1"=1)
   4 - filter("B"."C1"=1)

-------------------------------------------------------------------------------
-------
ALTERNATIVE PLANS SECTION
-------------------------------------------------------------------------------
-------

Plan 2
------
```

```
  Plan Origin                  :Cursor Cache
  Plan Hash Value              :1207490703
  Executions                   :1
  Elapsed Time                 :0.003 sec
  CPU Time                     :0.000 sec
  Buffer Gets                  :9
  Disk Reads                   :0
  Disk Writes                  :0

Notes:
  1. Statistics shown are averaged over multiple executions.
  2. The plan with id  could not be reproduced in the current
environment. For t
his reason, a SQL plan baseline cannot be created to instruct the Oracle
optimiz
er to pick this plan in the future.

--------------------------------------------------------------------------
---
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time
|
--------------------------------------------------------------------------
---
|   0 | SELECT STATEMENT   |      |       |       |     6 (100)|
|
|   1 |  HASH JOIN         |      |     4 |    56 |     6  (17)|
00:00:01 |
|   2 |   INDEX RANGE SCAN | I2   |     1 |     8 |     2   (0)|
00:00:01 |
|   3 |   TABLE ACCESS FULL| BAR  |     4 |    24 |     3   (0)|
00:00:01 |
--------------------------------------------------------------------------
---

Plan 1
------

  Plan Origin                  :Cursor Cache
  Plan Hash Value              :3834250642
  Executions                   :1
  Elapsed Time                 :0.002 sec


------------ See the SQL plan baselines
SQL> @system
SQL> connect system/oracle
Connected.
SQL> select plan_name, origin, enabled, accepted , fixed,optimizer_cost
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

PLAN_NAME                      ORIGIN         ENA ACC FIX OPTIMIZER_COST
------------------------------ -------------- --- --- --- --------------
SQL_PLAN_7zqfaxj4fd78743ac900d AUTO-CAPTURE   YES NO  NO               4
SQL_PLAN_7zqfaxj4fd787e53205ee AUTO-SQLTUNE   YES YES NO             247
SQL_PLAN_7zqfaxj4fd787e6003f7b AUTO-CAPTURE   YES YES NO             125
```

```
------------ See the created profile
SQL> select name from dba_sql_profiles
  2  where sql_text like 'select f.c1,f.c2,b.c2 from foo f, bar b%';

NAME
------------------------------
SYS_SQLPROF_01255f68a3330001

------------ Cleanup
SQL> set echo off

Table dropped.

Table dropped.

Table dropped.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```