SQL Plan Management

January 2014

Prepared for XXXX

Primary Author: Karl Arao

Table of Contents

I. Introduction

One of the biggest challenges in SQL Performance is the sudden change in execution plan and this happens whenever there are changes in optimizer environment, statistics, and schema changes.

SQL Plan Management (SPM) solves this issue by providing a mechanism to maintain consistent SQL performance regardless of the changes mentioned above. To guarantee performance, only accepted execution plans will be used. The new plans will be tracked and only be accepted when it is evolved.

For more details on how SPM works and documentation check the Appendix A.

II. How to use this document

This document is a HOWTO on SPM. The doc starts with the essential concepts and then diving into the common scenarios of SPM. A test case data is provided on Appendix B and will be first mentioned on the "Evolution" section of concepts and used throughout this doc. Having consistent and repeatable test cases is essential for finding out the ins and outs of a technology that's why on each section there's a brief and to the point explanation followed by set of scripts detailed on Appendix C to help administer the SPM without difficulty.

A test database is a must to create the test data and run the scripts. You can extract the attached spm.zip on any directory of your test machine or can even be executed from SQL*Developer.

III. Views

Here are the two common views used in SPM

DBA_SQL_PLAN_BASELINES displays information about the SQL plan baselines currently created for specific SQL statements.
DBA_SQL_MANAGEMENT_CONFIG displays the configuration parameters of the SQL management base.

IV. Parameters

Here are the two database parameters used in SPM

OPTIMIZER_USE_SQL_PLAN_BASELINES (default TRUE) enables or disables the use of SQL plan baselines stored in SQL Management Base. When enabled, the optimizer looks for a SQL plan

baseline for the SQL statement being compiled. If one is found in SQL Management Base, then the optimizer will cost each of the baseline plans and pick one with the lowest cost.
OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES (default FALSE) enables or disables the automatic recognition of repeatable SQL statements, as well as the generation of SQL plan baselines for such statements.

V. Package

The package used for managing SQL Plan Management is DBMS_SPM. The package is owned by SYS. The EXECUTE package privilege is required to execute its procedures. Any user granted the ADMINISTER SQL MANAGEMENT OBJECT privilege is able to execute the DBMS_SPM package.

VI. Concepts

Enabling and Capture

By default the OPTIMIZER_USE_SQL_PLAN_BASELINES is set to TRUE and any baselines created for a particular SQL will be evaluated and used. Loading the execution plans to SPM is another thing; you can do automatic or manual capture as shown below:

For **automatic capture**, set the OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES to TRUE on system or session level. The SQL has to be **executed twice** for it to be captured by SPM.

A logon trigger can also be created to do auto capture on just specific sessions

```
DROP TRIGGER SYS.SESSION_OPTIMIZATIONS;
CREATE OR REPLACE TRIGGER SYS.session_optimizations after logon on database
begin
   if (user in ('HR','KARLARAO')) then
      execute immediate('ALTER SESSION SET optimizer_capture_sql_plan_baselines=TRUE');
   end if;
end;
/
```

For **manual capture**, you can use the procedures below

- DBMS_SPM.UNPACK_STGTAB_BASELINE - Unpacks (imports) SQL plan baselines from a staging table into SQL management base (the SPM repository)
- DBMS_SPM.LOAD_PLANS_FROM_SQLSET - Loads plans stored in a SQL tuning set (STS) into SQL plan baselines
- DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE - Loads one or more plans present in the cursor cache for a SQL statement

The automatic capture logon trigger is the preferred way.

Evolution

When the optimizer finds a new plan for a SQL statement, the database adds the plan to the plan history as a non-accepted plan. The DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE evolves the non-accepted plan by comparing the performance of the non-accepted SQL against the baseline. After the evolution process the SQL with the lowest cost will be used as a new baseline. There are three options to do the evolution process:

- Run Evolve and accept if performance is better (default)
  - parameters: VERIFY=YES, COMMIT=YES
- Run Evolve and report only (do not accept)
  - parameters: VERIFY=YES, COMMIT=NO
- Run Evolve and accept without testing performance
  - parameters: VERIFY=NO, COMMIT=YES

Do the following to demo the evolution process and the rest of the concepts:

1) Create the logon trigger from the "Enabling and Capture" section above, change the parsing schema accordingly
2) Go to Appendix B and execute the step 1 (spm_demo_gendata.sql) to create the table and generate the data
3) Execute the step 3 (spm_demo_query.sql) twice to automatically create the 1st baseline
4) Execute the step 2 (spm_demo_createindex.sql) to create the index and gather the stats
5) Execute the step 3 (spm_demo_query.sql) twice to automatically create the 2nd baseline

```
KARLARAO@orcl> @spm_baselines

PARSING_ CREATED              PLAN_NAME                      SQL_HANDLE                SQL_TEXT                            OPTIMIZER_COST ENA ACC FIX REP ORIGIN
-------- -------------------- ------------------------------ ------------------------- ----------------------------------- -------------- --- --- --- --- -------
KARLARAO 01/12/14 18:15:37    SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                 7 YES YES NO  YES AUTO-CA
                                                                                                                                                 TURE

KARLARAO 01/12/14 18:17:47    SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                 2 YES NO  NO  YES AUTO-CA
                                                                                                                                                 TURE


KARLARAO@orcl> @spm_evolve
Enter value for sql_handle: SQL_e543035defc5f593
Enter value for verify: YES
Enter value for commit: YES

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

Inputs:
-------
  SQL_HANDLE = SQL_e543035defc5f593
  PLAN_NAME  =
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = YES
  COMMIT     = YES

Plan: SQL_PLAN_fahs3brrwbxcm65b66921
------------------------------------
  Plan was verified: Time used .13 seconds.
  Plan passed performance criterion: 6.34 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):                    .265
.02             13.25
  CPU Time(ms):                        .222              0
  Buffer Gets:                          19               3               6.33
  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


PL/SQL procedure successfully completed.

KARLARAO@orcl> @spm_baselines

PARSING_ CREATED              PLAN_NAME                      SQL_HANDLE                SQL_TEXT                            OPTIMIZER_COST ENA ACC FIX REP ORIGIN
-------- -------------------- ------------------------------ ------------------------- ----------------------------------- -------------- --- --- --- --- --------
KARLARAO 01/12/14 18:15:37    SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                7 YES YES NO  YES AUTO-CAP
                                                                                                                                           TURE

KARLARAO 01/12/14 18:17:47    SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                2 YES YES NO  YES AUTO-CAP
                                                                                                                                           TURE
```

Display baselines of SQL_ID

To display the baselines associated by a SQL_ID make use of the following views

- V$SQL
- DBA_SQL_PLAN_BASELINES

```
19:14:22 SYS@orcl> @spm_find_sql
Enter value for sql_text: %skew%
Enter value for sql_id:

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE     EXACT_MATCHING_SIGNATURE     FORCE_MATCHING_SIGNATURE SQL_TEXT
------------- ------------ --------------- ---------------------------- ---------------------------- ----------------------------------------
--------------------------------------------------------------------------------
6vbwwqumt7ns7            0      2836784050          141451454813021375          8837324131823865881 select sql_id, child_number, plan_h
a5jq5khm9w64n            0      3844559076         16520051560114681235          2058880940649185083 select * from skew where skew=3
ctak9z6zcf4gx            0      2836784050         17267678976023022210         11747550215032002950 select sql_id, child_number, plan_h

19:14:41 SYS@orcl> @spm_sqlid
Enter value for sql_id: a5jq5khm9w64n

PARSING_ CREATED    PLAN_NAME                      SQL_HANDLE                SQL_TEXT
OPTIMIZER_COST ENA ACC FIX ORIGIN
-------- ---------- ------------------------------ ------------------------- ------------------------------------------------------------------------------- -------
-------- --- --- --- --------------
KARLARAO 12-JAN-14  SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                                                  2 YES
YES NO AUTO-CAPTURE
         06.17.47.0
         00000 PM

KARLARAO 12-JAN-14  SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                                                  7 YES
YES NO  AUTO-CAPTURE
         06.15.37.0
         00000 PM
```

Display execution plans of baselines

To display the plans of baselines make use of the DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE

```
SYS@orcl> @spm_plans
Enter value for sql_handle: SQL_e543035defc5f593
```

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

     ---------------------------------------------------------------------------
     SQL handle: SQL_e543035defc5f593
     SQL text: select * from skew where skew=3
     ---------------------------------------------------------------------------

     ---------------------------------------------------------------------------
     Plan name: SQL_PLAN_fahs3brrwbxcm65b66921         Plan id: 1706453281
     Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
     ---------------------------------------------------------------------------


     PLAN_TABLE_OUTPUT
     ------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------
     Plan hash value: 3844559076

     ---------------------------------------------
     | Id  | Operation                   | Name     |
     ---------------------------------------------
     |   0 | SELECT STATEMENT            |          |
     |   1 |  TABLE ACCESS BY INDEX ROWID| SKEW     |
     |   2 |   INDEX RANGE SCAN          | SKEW_IDX |
     ---------------------------------------------

     ---------------------------------------------------------------------------

     PLAN_TABLE_OUTPUT
     ------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------
     Plan name: SQL_PLAN_fahs3brrwbxcm950a48a8         Plan id: 2500479144
     Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
     ---------------------------------------------------------------------------

     Plan hash value: 246648590

     ----------------------------------
     | Id  | Operation         | Name |
     ----------------------------------
     |   0 | SELECT STATEMENT  |      |
     |   1 |  TABLE ACCESS FULL| SKEW |

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

Fixed

FIXED is an attribute of a baseline (default=NO), if set to YES then new plans will not be considered for this baseline and will not be evolved over time. A fixed plan takes precedence over a non-fixed plan.

```
19:04:04 SYS@orcl> @spm_baselines

PARSING_ CREATED             PLAN_NAME                      SQL_HANDLE                SQL_TEXT                           OPTIMIZER_COST ENA ACC FIX REP ORIGIN
-------- ------------------- ------------------------------ ------------------------- ---------------------------------- -------------- --- --- --- --- ---------
KARLARAO 01/12/14 18:15:37   SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                7 YES YES NO  YES AUTO-CAP
                                                                                                                                                    TURE
KARLARAO 01/12/14 18:17:47   SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                2 YES YES NO  YES AUTO-CAP
                                                                                                                                                    TURE


19:04:07 SYS@orcl> @spm_fixed
Enter value for sql_handle: SQL_e543035defc5f593
Enter value for plan_name: SQL_PLAN_fahs3brrwbxcm65b66921
Enter value for yes_or_no: YES

PL/SQL procedure successfully completed.


19:04:35 SYS@orcl> @spm_baselines

PARSING_ CREATED             PLAN_NAME                      SQL_HANDLE                SQL_TEXT                           OPTIMIZER_COST ENA ACC FIX REP ORIGIN
-------- ------------------- ------------------------------ ------------------------- ---------------------------------- -------------- --- --- --- --- ---------
KARLARAO 01/12/14 18:15:37   SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                7 YES YES NO  YES AUTO-CAP
                                                                                                                                                    TURE
KARLARAO 01/12/14 18:17:47   SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                2 YES YES YES YES AUTO-CAP
                                                                                                                                                    TURE
```

Enable/Disable

ENABLED is an attribute of a baseline (default=YES), if set to NO then the execution plan will not be used and will not be evaluated for the evolution process

```
19:29:54 SYS@orcl> @spm_baselines

PARSING_ CREATED              PLAN_NAME                      SQL_HANDLE                SQL_TEXT                            OPTIMIZER_COST ENA ACC FIX REP ORIGIN
-------- -------------------- ------------------------------ ------------------------- ----------------------------------- -------------- --- --- --- --- ---------
KARLARAO 01/12/14 18:15:37    SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                 7 YES YES NO  YES AUTO-CAP
                                                                                                                                                    TURE
KARLARAO 01/12/14 18:17:47    SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                 2 YES YES NO  YES AUTO-CAP
                                                                                                                                                    TURE


19:29:58 SYS@orcl> @spm_plans
Enter value for sql_handle: SQL_e543035defc5f593

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

--------------------------------------------------------------------------------
SQL handle: SQL_e543035defc5f593
SQL text: select * from skew where skew=3
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Plan name: SQL_PLAN_fahs3brrwbxcm65b66921         Plan id: 1706453281
Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
--------------------------------------------------------------------------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------
Plan hash value: 3844559076

---------------------------------------------
| Id  | Operation                   | Name     |
---------------------------------------------
|   0 | SELECT STATEMENT            |          |
|   1 |  TABLE ACCESS BY INDEX ROWID| SKEW     |
|   2 |   INDEX RANGE SCAN          | SKEW_IDX |
---------------------------------------------

--------------------------------------------------------------------------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------
Plan name: SQL_PLAN_fahs3brrwbxcm950a48a8         Plan id: 2500479144
Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
--------------------------------------------------------------------------------

Plan hash value: 246648590

----------------------------------
| Id  | Operation         | Name |
----------------------------------
|   0 | SELECT STATEMENT  |      |
|   1 |  TABLE ACCESS FULL| SKEW |

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

34 rows selected.


19:35:11 SYS@orcl> @spm_enable
Enter value for sql_handle: SQL_e543035defc5f593
Enter value for plan_name: SQL_PLAN_fahs3brrwbxcm950a48a8
Enter value for yes_or_no: NO

PL/SQL procedure successfully completed.


19:35:58 SYS@orcl> @spm_baselines

PARSING_ CREATED              PLAN_NAME                      SQL_HANDLE                SQL_TEXT                            OPTIMIZER_COST ENA
ACC FIX REP ORIGIN
-------- -------------------- ------------------------------ ------------------------- ----------------------------------- -------------- ---
--- --- --- ---------
KARLARAO 01/12/14 18:15:37    SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                 7 NO
YES NO  YES AUTO-CAP

TURE
KARLARAO 01/12/14 18:17:47    SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                 2 YES
YES NO  YES AUTO-CAP

TURE
```

Drop

To drop a baseline or all baselines make use of the DBMS_SPM.DROP_SQL_PLAN_BASELINE

```
19:54:42 SYS@orcl> @spm_baselines

PARSING_ CREATED             PLAN_NAME                      SQL_HANDLE                SQL_TEXT                            OPTIMIZER_COST ENA ACC FIX REP ORIGIN
-------- ------------------- ------------------------------ ------------------------- ----------------------------------- -------------- --- --- --- --- --------
KARLARAO 01/12/14 18:15:37    SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                 7 NO  YES NO  YES AUTO-CAP
                                                                                                                                                    TURE

KARLARAO 01/12/14 18:17:47    SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                 2 YES YES NO  YES AUTO-CAP
                                                                                                                                                    TURE


19:54:47 SYS@orcl> ! ls spm_drop*
spm_drop_all_baseline.sql  spm_drop_baseline.sql


19:55:05 SYS@orcl> @spm_drop_baseline.sql
Enter value for sql_handle: SQL_e543035defc5f593
Enter value for plan_name: SQL_PLAN_fahs3brrwbxcm950a48a8

PL/SQL procedure successfully completed.

19:55:23 SYS@orcl> @spm_baselines

PARSING_ CREATED             PLAN_NAME                      SQL_HANDLE                SQL_TEXT                            OPTIMIZER_COST ENA
ACC FIX REP ORIGIN
-------- ------------------- ------------------------------ ------------------------- ----------------------------------- -------------- ---
--- --- --- --------
KARLARAO 01/12/14 18:17:47    SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                 2 YES
YES NO  YES AUTO-CAP

TURE
```

Dropping all baselines means looping through the distinct sql_handle and plan_name of DBA_SQL_PLAN_BASELINES view and executing DBMS_SPM.DROP_SQL_PLAN_BASELINE on each of them.

```
20:08:00 SYS@orcl> @spm_drop_all_baseline.sql
plans deleted: 6

PL/SQL procedure successfully completed.

20:08:07 SYS@orcl> @spm_baselines

no rows selected
```

Configure

The baselines are stored in the SQL management base (SMB) and reside in the SYSAUX tablespace. The space usage on the SMB is controlled by two attributes and can be modified by making use of DBMS_SPM.CONFIGURE

- space_budget_percent (default 10): Maximum size as a percentage of SYSAUX space. Allowable values 1-50
- plan_retention_weeks (default 53): Number of weeks unused plans are retained before being purged. Allowable values 5-523 weeks

```
20:24:20 SYS@orcl> @spm_smb_configure.sql

PARAMETER_NAME                 PARAMETER_VALUE
```

```
------------------------------ ---------------
SPACE_BUDGET_PERCENT                        10
PLAN_RETENTION_WEEKS                        53

Enter value for space_budget_percent: 9
Enter value for plan_retention_weeks: 52

PL/SQL procedure successfully completed.


PARAMETER_NAME                 PARAMETER_VALUE
------------------------------ ---------------
SPACE_BUDGET_PERCENT                         9
PLAN_RETENTION_WEEKS                        52
```

VII. Scenarios

Below are the common scenarios for SPM

Gather Stats

The new plans created as a result of gathering statistics are stored as non-accepted plans. These plans have to be manually evolved to be used.

@spm_demo_cleanup.sql
@spm_drop_all_baseline.sql
@spm_demo_createindex.sql
@spm_demo_fudgestats.sql
@spm_demo_query.sql
@spm_demo_query.sql <-- first baseline created (full scan)
@spm_baselines.sql
@spm_demo_createindex.sql <-- gather stats, index add
@spm_demo_query.sql <-- new baseline created (index scan), but still first baseline is used
@spm_baselines.sql
@spm_evolve.sql

Index Add

When adding indexes, new plans will be stored as non-accepted plans. Same as when gathering statistics, these plans have to be manually evolved to be used.

@spm_demo_cleanup.sql
@spm_drop_all_baseline.sql
@spm_demo_query.sql
@spm_demo_query.sql <-- first baseline created (full scan)
@spm_baselines.sql
@spm_demo_createindex.sql <-- gather stats, index add
@spm_demo_query.sql <-- new baseline created (index scan), but still first baseline is used
@spm_baselines.sql

@spm_evolve.sql

Index Drop

If the execution plan cannot be reproduced due to an index drop, then the baselines cannot be used even if it's ACCEPTED. If there are no other ACCEPTED plans that suits the new execution plan. Then a new baseline will be created and used (with full scan). If the index gets recreated then the old baseline (index scan) will be used.

```
@spm_demo_cleanup.sql
@spm_drop_all_baseline.sql
@spm_demo_createindex.sql
@spm_demo_query.sql
@spm_demo_query.sql      <-- first baseline created (index scan)
@spm_baselines.sql
@spm_plans.sql
@spm_demo_cleanup.sql
@spm_demo_query.sql
@spm_baselines.sql
```

Alter object – add/drop/rename/modify column

When new columns are added or dropped then it has no effect on the baselines.
When the columns and objects that are being referenced on the query are renamed,dropped, or modified (change of data type) then the baselines will not be used.

```
-- ADD COLUMN
@spm_demo_cleanup.sql
@spm_drop_all_baseline.sql
@spm_demo_createindex.sql
@spm_demo_query.sql
@spm_demo_query.sql                    <-- first baseline create (index scan)
@spm_baselines.sql
ALTER TABLE skew ADD skew2 varchar2(50);       <-- no effect
@spm_demo_query.sql
@spm_baselines.sql

-- DROP COLUMN
ALTER TABLE skew DROP COLUMN skew2;          <-- no effect
@spm_demo_query.sql
@spm_baselines.sql

-- RENAME COLUMN
ALTER TABLE skew RENAME COLUMN skew to skew2;
```

```
select * from skew where skew2=3;
select * from skew where skew2=3;            <-- baseline will not be used, a new one will be created
@spm_baselines.sql

-- RENAME TABLE
ALTER TABLE skew RENAME TO skew2;
select * from skew2 where skew2=3;
select * from skew2 where skew2=3;           <-- baseline will not be used, a new one will be created
@spm_baselines.sql

-- REVERT TO OLD COLUMN/TABLE NAMES
ALTER TABLE skew2 RENAME COLUMN skew2 to skew;
ALTER TABLE skew2 RENAME TO skew;

-- MODIFY COLUMN FROM NUMBER TO VARCHAR2
truncate table skew;
ALTER TABLE skew MODIFY skew varchar2(100) not null;
insert into skew select rownum all_distinct, 10000 skew from dual connect by level <= 10000;
update skew set skew=all_distinct where all_distinct in (1,2,3,4,5,6,7,8,9,10);
select skew, count(*) from skew group by skew order by skew;
@spm_demo_query.sql                  <-- this created a new baseline (full scan)
@spm_baselines.sql

-- MODIFY COLUMN - REVERT TO NUMBER
truncate table skew;
ALTER TABLE skew MODIFY skew number;
insert into skew select rownum all_distinct, 10000 skew from dual connect by level <= 10000;
update skew set skew=all_distinct where all_distinct in (1,2,3,4,5,6,7,8,9,10);
select skew, count(*) from skew group by skew order by skew;
@spm_demo_query.sql                  <-- reverted back to the old baseline (index scan)
@spm_baselines.sql

-- MODIFY COLUMN - ALTER NUMBER COLUMN
truncate table skew;
ALTER TABLE skew MODIFY skew number(30);
insert into skew select rownum all_distinct, 10000 skew from dual connect by level <= 10000;
update skew set skew=all_distinct where all_distinct in (1,2,3,4,5,6,7,8,9,10);
select skew, count(*) from skew group by skew order by skew;
@spm_demo_query.sql                  <-- still used the old baseline (index scan)
@spm_baselines.sql
```

Drop and recreate a table

When a table is dropped the ACCEPTED plans still remains in the repository. On the recreation of the table all the corresponding objects (indexes) that are referenced on the baseline also has to be recreated, else the old baseline will not be used and a new one will be created (and used).

```
@spm_demo_cleanup.sql
@spm_drop_all_baseline.sql
@spm_demo_createindex.sql
@spm_demo_query.sql
@spm_demo_query.sql      <-- first baseline create (index scan)
@spm_baselines.sql
drop table skew cascade constraints;
@spm_demo_gendata.sql
@spm_demo_query.sql      <-- without creating the indexes a new baseline is created (with full scan)
@spm_demo_createindex.sql <-- recreate the indexes
@spm_demo_query.sql      <-- old baseline used (index scan)
```

Truncate a table

When a table is truncated, the baselines will still be used

```
@spm_demo_cleanup.sql
@spm_drop_all_baseline.sql
@spm_demo_createindex.sql
@spm_demo_query.sql
@spm_demo_query.sql      <-- first baseline create (index scan)
@spm_baselines.sql
truncate table skew;
@spm_demo_query.sql      <-- the baseline is still used (index scan)
@spm_baselines.sql
```

Change optimizer environment

When there is a change in the optimizer environment then the new plans will be stored as non-accepted plans. Same as when gathering statistics and adding indexes, these plans have to be manually evolved to be used.

```
@spm_demo_cleanup.sql
@spm_drop_all_baseline.sql
@spm_demo_createindex.sql
@spm_demo_query.sql
@spm_demo_query.sql      <-- first baseline create (index scan)
@spm_baselines.sql
alter session set optimizer_index_cost_adj=10000;   <-- influence full scans
@spm_demo_query.sql      <-- new baseline create (full scan)
```

@spm_baselines.sql

Backup, Drop, Restore baselines on the same database

In the event of a bug that's being caused by baselines there may be a case where we need to disable the SPM and drop and recreate all the baselines. Below are the steps to do that:

1) Create the staging table and pack the baselines into the staging table

```
var n NUMBER
EXEC dbms_spm.create_stgtab_baseline('SPM_STAGE');
EXEC :n := dbms_spm.pack_stgtab_baseline('SPM_STAGE');
```

2) Query the packed baselines

```
SET long 1000000
SET longchunksize 30
colu sql_text format a30
colu optimizer_cost format 999,999 heading 'Cost'
colu buffer_gets    format 999,999 heading 'Gets'
SELECT sql_text, OPTIMIZER_COST, CPU_TIME, BUFFER_GETS, COMP_DATA FROM SPM_STAGE;
```

3) Export the table and backup the dump file

4) Drop all baselines

```
SET SERVEROUT ON;
DECLARE
 x NUMBER;
 y NUMBER := 0;
BEGIN
 FOR i IN (SELECT DISTINCT sql_handle, plan_name FROM dba_sql_plan_baselines)
 LOOP
   x := DBMS_SPM.DROP_SQL_PLAN_BASELINE(i.sql_handle, i.plan_name);
   y := y + x;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('plans deleted: '||y);
END;
/
SET SERVEROUT OFF;
```

5) To restore, locate the backup dump file and import the staging table then unpack the baselines

```
var n NUMBER
exec :n:=DBMS_SPM.UNPACK_STGTAB_BASELINE('SPM_STAGE');
```

6) Verify

```
col parsing_schema format a8
col created format a20
col sql_handle format a25
col sql_text format a35
col origin format a8
SELECT parsing_schema_name parsing_schema, TO_CHAR(created,'MM/DD/YY HH24:MI:SS') created, plan_name, sql_handle, substr(sql_text,1,35) sql_text, optimizer_cost, enabled, accepted, fixed, reproduced, origin
FROM dba_sql_plan_baselines order by 2,4 asc;
```

Move baselines to another database

To move baselines from one database to another follow the steps below, for more details check this MOS note "Transporting SQL PLAN Baselines from One Database to Another. (Doc ID 880485.1)"

1) Create the staging table and pack the baselines into the staging table

```
var n NUMBER
EXEC dbms_spm.create_stgtab_baseline('SPM_STAGE');
EXEC :n := dbms_spm.pack_stgtab_baseline('SPM_STAGE');
```

2) Query the packed baselines

```
SET long 1000000
SET longchunksize 30
colu sql_text format a30
colu optimizer_cost format 999,999 heading 'Cost'
colu buffer_gets    format 999,999 heading 'Gets'
SELECT sql_text, OPTIMIZER_COST, CPU_TIME, BUFFER_GETS, COMP_DATA FROM SPM_STAGE;
```

3) Export the table and copy the dump file to the destination environment

4) Import the table on the destination environment, and unpack the baselines

```
var n NUMBER
exec :n:=DBMS_SPM.UNPACK_STGTAB_BASELINE('SPM_STAGE');
```

5) Verify

```
col parsing_schema format a8
col created format a20
col sql_handle format a25
col sql_text format a35
```

```
col origin format a8
SELECT parsing_schema_name parsing_schema, TO_CHAR(created,'MM/DD/YY HH24:MI:SS') created, plan_name, sql_handle, substr(sql_text,1,35) sql_text, optimizer_cost, enabled, accepted, fixed, reproduced, origin
FROM dba_sql_plan_baselines order by 2,4 asc;
```

Loading Hinted Execution Plans into SQL Plan Baseline

This step by step is very useful if the SQL coming from the application can't be modified and needed hints to run a good execution plan. For more details check this MOS note “Loading Hinted Execution Plans into SQL Plan Baseline. (Doc ID 787692.1)”

1) Execute the hinted SQL and run DBMS_XPLAN.DISPLAY_CURSOR

```
03:09:01 KARLARAO@orcl> select /* test1 */ /*+ FULL (a) */ * from skew a where skew=3;

ALL_DISTINCT       SKEW
------------ ----------
           3          3

03:09:16 KARLARAO@orcl> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------
SQL_ID  3247jpkdf62tw, child number 0
-------------------------------------
select /* test1 */ /*+ FULL (a) */ * from skew a where skew=3

Plan hash value: 246648590

-------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       |     7 (100)|          |
|*  1 |  TABLE ACCESS FULL| SKEW |     1 |     7 |     7   (0)| 00:00:01 |

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

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

   1 - filter("SKEW"=3)


18 rows selected.
```

2) Get the V$SQL details of the hinted SQL, keep note of SQL_ID and PLAN_HASH_VALUE

```
03:09:24 KARLARAO@orcl> @spm_find_sql
Enter value for sql_text: %test1%
Enter value for sql_id:

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE     EXACT_MATCHING_SIGNATURE     FORCE_MATCHING_SIGNATURE SQL_TEXT
------------- ------------ --------------- ---------------------------- ---------------------------- ----------------------------------
3247jpkdf62tw            0       246648590          5981025266313531807          4160911647290526840 select /* test1 */ /*+ FULL (a) */
```

3) If the original SQL is not yet captured as a baseline then load it using DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE

```
03:10:35 KARLARAO@orcl> @spm_find_sql
Enter value for sql_text: %select * from skew where skew=3%
Enter value for sql_id:

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE     EXACT_MATCHING_SIGNATURE     FORCE_MATCHING_SIGNATURE SQL_TEXT
------------- ------------ --------------- ---------------------------- ---------------------------- ----------------------------------
a5jq5khm9w64n            0      3844559076         16520051560114681235          2058880940649185083 select * from skew where skew=3


var res number ;
exec :res := dbms_spm.load_plans_from_cursor_cache(sql_id => '&original_sql_id', plan_hash_value => '&original_plan_hash_value' );


03:10:56 KARLARAO@orcl> var res number ;
exec :res := dbms_spm.load_plans_from_cursor_cache(sql_id => '&original_sql_id', plan_hash_value => '&original_plan_hash_value' );03:11:19
KARLARAO@orcl>
Enter value for original_sql_id: a5jq5khm9w64n
Enter value for original_plan_hash_value: 3844559076

PL/SQL procedure successfully completed.
```

4) Verify the baseline exist for the original SQL

```
03:11:25 KARLARAO@orcl> @spm_baselines
Enter value for sql_text:
Enter value for exact_matching_signature:

PARSING_ CREATED              PLAN_NAME                      SQL_HANDLE                SQL_TEXT                            OPTIMIZER_COST ENA
ACC FIX REP ORIGIN
-------- -------------------- ------------------------------ ------------------------- ----------------------------------- -------------- ---
--- --- --- --------

KARLARAO 01/13/14 03:11:25     SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                 2 YES
YES NO  YES MANUAL-L

OAD
```

5) Associate the hinted execution plan to the original sql_handle.

```
var res number
exec :res := dbms_spm.load_plans_from_cursor_cache(sql_id => '&hinted_SQL_ID', plan_hash_value => &hinted_plan_hash_value, sql_handle =>
'&sql_handle_for_original');

03:11:52 KARLARAO@orcl> var res number
exec :res := dbms_spm.load_plans_from_cursor_cache(sql_id => '&hinted_SQL_ID', plan_hash_value => &hinted_plan_hash_value, sql_handle =>
'&sql_handle_for_original');03:13:23 KARLARAO@orcl>
Enter value for hinted_sql_id: 3247jpkdf62tw
Enter value for hinted_plan_hash_value: 246648590
Enter value for sql_handle_for_original: SQL_e543035defc5f593

PL/SQL procedure successfully completed.
```

6) Verify

```
03:13:52 KARLARAO@orcl> @spm_baselines
Enter value for sql_text:
Enter value for exact_matching_signature:

PARSING_ CREATED              PLAN_NAME                      SQL_HANDLE                SQL_TEXT                            OPTIMIZER_COST ENA
ACC FIX REP ORIGIN
-------- -------------------- ------------------------------ ------------------------- ----------------------------------- -------------- ---
--- --- --- --------
KARLARAO 01/13/14 03:11:25     SQL_PLAN_fahs3brrwbxcm65b66921 SQL_e543035defc5f593      select * from skew where skew=3                 2 YES
YES NO  YES MANUAL-L

OAD

KARLARAO 01/13/14 03:13:52     SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                 7 YES
YES NO  YES MANUAL-L

OAD
```

7) Drop the original baseline

```
03:15:19 KARLARAO@orcl> @spm_drop_baseline.sql
Enter value for sql_handle: SQL_e543035defc5f593
Enter value for plan_name: SQL_PLAN_fahs3brrwbxcm65b66921

PL/SQL procedure successfully completed.



03:15:56 KARLARAO@orcl>
03:15:56 KARLARAO@orcl> @spm_baselines
Enter value for sql_text:
Enter value for exact_matching_signature:

PARSING_ CREATED              PLAN_NAME                      SQL_HANDLE                SQL_TEXT                             OPTIMIZER_COST ENA
ACC FIX REP ORIGIN
-------- -------------------- ------------------------------ ------------------------- ---------------------------------- -------------- ---
--- --- --- --------
KARLARAO 01/13/14 03:13:52    SQL_PLAN_fahs3brrwbxcm950a48a8 SQL_e543035defc5f593      select * from skew where skew=3                 7 YES
YES NO  YES MANUAL-L

OAD
```

Appendix A: Useful Documentation

- Master Note: Plan Stability Features (Including SQL Plan Management (SPM)) (Doc ID 1359841.1)
- How to Use SQL Plan Management (SPM) - Example Usage (Doc ID 456518.1)
- Oracle WhitePaper - SPM in 11g
  http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-sql-plan-management-11gr2-133099.pdf
- Using SQL Plan Management
  http://docs.oracle.com/cd/E11882_01/server.112/e41573/optplanmgmt.htm#PFGRF007
- DBMS_SPM http://docs.oracle.com/cd/E11882_01/appdev.112/e40758/d_spm.htm#ARPLS150

Appendix B: Test Case Data

Step 1) generate data (spm_demo_gendata.sql)

```
create table skew as select rownum all_distinct, 10000 skew from dual connect by level <= 10000;
update skew set skew=all_distinct where rownum<=10;
select skew, count(*) from skew group by skew order by skew;
```

Step 2) create index gather stats (spm_demo_createindex.sql)

```
create index skew_idx on skew(skew);
exec dbms_stats.gather_index_stats(user,'SKEW_IDX', no_invalidate => false);
exec dbms_stats.gather_table_stats(user,'SKEW', no_invalidate => false);
```

Step 3) query (spm_demo_query.sql)

```
select * from skew where skew=3;
select * from table(dbms_xplan.display_cursor);
```

Step 4) set stats to make full scan on skew=1 (spm_demo_fudgestats.sql)

```
exec dbms_stats.set_table_stats(user, 'SKEW', numrows => 1, numblks => 1, avgrlen => 1, no_invalidate
=> false);
```

Step 5) cleanup (spm_demo_cleanup.sql)

```
drop index skew_idx;
exec dbms_stats.delete_table_stats(user,'SKEW');
exec dbms_stats.gather_table_stats(user,'SKEW',method_opt=>'for columns skew size 1');
```

Appendix C: Scripts

Below are the scripts:

spm.zip

```
-- get the signature of SQL_ID, baselines use EXACT_MATCHING_SIGNATURE - (spm_find_sql.sql)
set verify off
col exact_matching_signature format 99999999999999999999999999
col force_matching_signature format 99999999999999999999999999
select sql_id, child_number, plan_hash_value, exact_matching_signature, force_matching_signature,
substr(sql_text,1,35) sql_text
from v$sql
where upper(sql_text) like upper(nvl('&sql_text',sql_text))
and sql_text not like '%from v$sql where sql_text like nvl(%'
and sql_id like nvl('&sql_id',sql_id)
order by 1,2;


-- query all baselines - (spm_baselines.sql)
set verify off
col parsing_schema format a8
col created format a20
col sql_handle format a25
col sql_text format a35
col origin format a8
SELECT parsing_schema_name parsing_schema, TO_CHAR(created,'MM/DD/YY HH24:MI:SS')
created, plan_name, sql_handle, substr(sql_text,1,35) sql_text, optimizer_cost, enabled, accepted, fixed,
reproduced, origin
FROM dba_sql_plan_baselines
where upper(sql_text) like upper(nvl('&sql_text',sql_text))
and sql_text not like '%from v$sql where sql_text like nvl(%'
and signature like nvl('&exact_matching_signature',signature)
order by 2,4 asc;


-- find baselines used by SQL_ID - (spm_sqlid.sql)
set verify off
col parsing_schema format a8
col created format a10
SELECT parsing_schema_name parsing_schema, created, plan_name, sql_handle, sql_text,
optimizer_cost, enabled, accepted, fixed, origin
FROM dba_sql_plan_baselines
WHERE signature IN (SELECT exact_matching_signature FROM v$sql WHERE sql_id like
nvl('&sql_id',sql_id))
```

```
/


-- View the exectution plan stored in baselines (format options - basic, typical, all) - (spm_plans.sql)
set lines 200
set verify off
SELECT * FROM
TABLE(DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE(sql_handle=>'&sql_handle',
format=>'basic'));


-- View the execution plan of SQL_ID - (spm_dplan.sql)
set lines 200
set verify off
select * from table(dbms_xplan.display_cursor('&sql_id','&child_no','advanced +peeked_binds'))
/


-- evolve - (spm_evolve.sql)
set verify off
SET SERVEROUTPUT ON
SET long 1000000
SET longchunksize 300
set lines 900
DECLARE
report clob;
BEGIN
report := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(
sql_handle => '&sql_handle',
verify => '&verify',
commit => '&commit');
DBMS_OUTPUT.PUT_LINE(report);
END;
/


-- alter baseline, FIXED - (spm_fixed.sql)
set verify off
declare
myplan pls_integer;
begin
myplan:=DBMS_SPM.ALTER_SQL_PLAN_BASELINE (sql_handle => '&sql_handle',plan_name  =>
'&plan_name',attribute_name => 'FIXED',   attribute_value => '&YES_OR_NO');
end;
/


-- alter baseline, DISABLE - (spm_enable.sql)
set verify off
declare
```

```
myplan pls_integer;
begin
myplan:=DBMS_SPM.ALTER_SQL_PLAN_BASELINE (sql_handle => '&sql_handle',plan_name  =>
'&plan_name',attribute_name => 'ENABLED',   attribute_value => '&YES_OR_NO');
end;
/


-- drop specific baseline - (spm_drop_baseline.sql)
set verify off
DECLARE
  plans_dropped    PLS_INTEGER;
BEGIN
  plans_dropped := DBMS_SPM.drop_sql_plan_baseline (
sql_handle => '&sql_handle',
plan_name  => '&plan_name');
DBMS_OUTPUT.put_line(plans_dropped);
END;
 /


-- drop all baselines - (spm_drop_all_baseline.sql)
SET SERVEROUT ON;
DECLARE
  x NUMBER;
  y NUMBER := 0;
BEGIN
  FOR i IN (SELECT DISTINCT sql_handle, plan_name FROM dba_sql_plan_baselines)
  LOOP
    x := DBMS_SPM.DROP_SQL_PLAN_BASELINE(i.sql_handle, i.plan_name);
    y := y + x;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('plans deleted: '||y);
END;
/
SET SERVEROUT OFF;


-- spm config parameters (spm_smb_configure.sql)
set lines 300
set verify off
SELECT PARAMETER_NAME, PARAMETER_VALUE FROM
DBA_SQL_MANAGEMENT_CONFIG;
BEGIN
  DBMS_SPM.configure('space_budget_percent', &space_budget_percent);
  DBMS_SPM.configure('plan_retention_weeks', &plan_retention_weeks);
END;
/
SELECT PARAMETER_NAME, PARAMETER_VALUE FROM
DBA_SQL_MANAGEMENT_CONFIG;
```