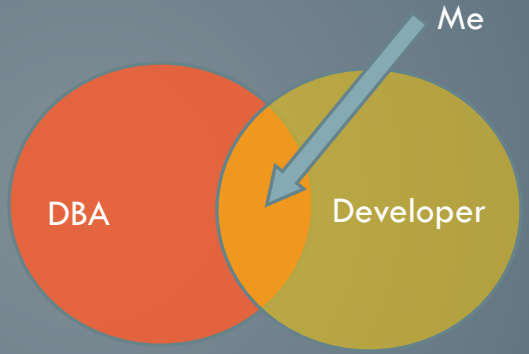
SQL Plan Management

(v1.4)

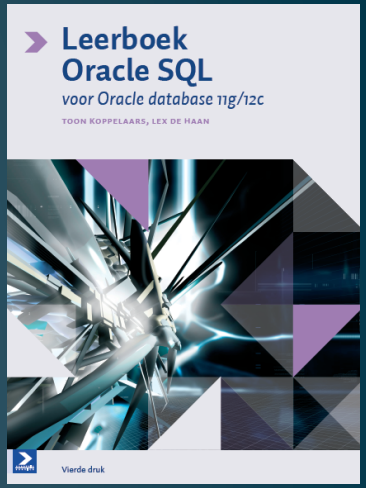
Toon Koppelaars, Netherlands

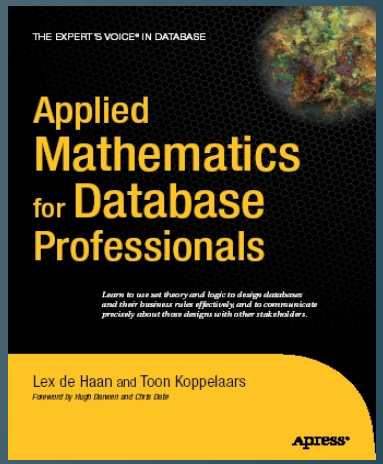
The speaker

- Oracle technology since 1987



- Twitter: @ToonKoppelaars
- Blogs
  - Thehelsinkideclaration.blogspot.com
  - Harmfultriggers.blogspot.com





RULE·GEN
data quality assurance

OakTable.net

Today's topic

- SQL Plan Management (aka SPM)
- This will be a tutorial
  - Not a 'deep dive' or 'hacking session'
- But will show interesting oddities

Contents

- SPM:
  - What is it and why would you want to use it?
  - How does it work?
    - Baseline repository
    - Selection, evolution, capture
- Researching common questions on SPM

Explain it

Demo it

Break it

What and Why?

- Plan stability
  - SPM is mechanism to provide sql execution plan stability

> "Allows execution plans for SQL to be stored so that plan remains consistent throughout schema changes, database reorganizations, and data volume changes."

- Like stored outlines, but more sophisticated
  - [Note: outlines have been deprecated in 11.1]
  - [Note: outlines (still) have precedence over SPM]
  - [Note: outline technology is used under-the-covers by SPM]

SPM this is an EE feature, i.e. does not require tuning pack

What and Why?

- Why would you want this?
  - SQL execution plan depends on many things
    - If these change, plan can change
- Couple of use-cases:
  1. New/patched database (ie. optimizer) version
  2. Changes to optimizer statistics can break good SQL
  3. Other changes:
     - System statistics and system settings
     - Optimizer related changes in parameter file
     - Schema and metadata definitions
     - SQL profile creation
     - Adaptive cursor sharing, cardinality feedback, …

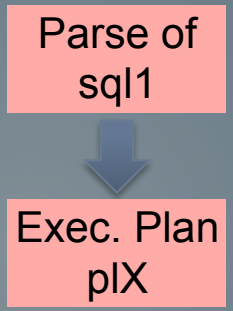
How does it work?

- In summary:
  1. For every SQL statement, repository of (accepted) execution plans is held in:
     - **`DBA_SQL_PLAN_BASELINES`**
  2. On hard parse:
     - Repository is searched for that execution plan
     - If found *or repository is empty*, then that plan is used
     - If not found *and other accepted plans exist*, then one of these accepted plans is used instead

How does it work?

- Feature controlled by init.ora parameter
  - `Optimizer_use_sql_plan_baselines = true/false`
  - Alter system + session modifiable
- Management via supplied package DBMS_SPM

Vizualized



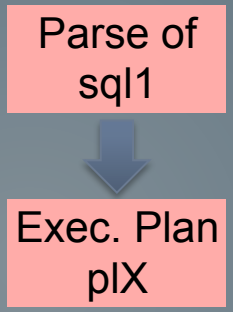
- If plX = plA1 then use it
- else force use of plA1

➔ Always use plA1…

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	
sql1	plA1	Y	
sql2	plB1	Y	
sql3	plC1	Y	

More than one plan can exist...



DBA_SQL_PLAN_BASELINES

SQL	plan	acc	
sql1	plA1	Y	
sql1	plA2	Y	
sql1	plA3	Y	
sql2	plB1	Y	
sql3	plC1	Y	

- If plX in (plA1,plA2,plA3) then use plX
- else choose one from plA1, plA2, plA3

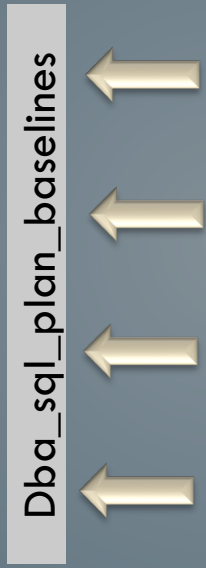
How “choose” is done, not well documented. Assumption: 1) same optimizer env, 2) least cost

SQL Plan Management concepts

- Three concepts
  - SQL plan baseline capture
  - SQL plan baseline selection
  - SQL plan baseline evolution

Baseline capture

- Four ways to load execution plans into repository



1. Capture all hard parses in sessions
   - `Optimizer_capture_sql_plan_baselines = true/false`
2. Pre-deliver/import from known set
   - `DBMS_SPM.Unpack_Stgtab_Baseline`
3. Import from SQL Tuning Set (requires Tuning Pack)
   - `DBMS_SPM.Load_Plans_From_Sqlset`
4. Pre-load from shared-pool (v$sql_plan)
   - `DBMS_SPM.Load_Plans_From_Sql_Cache`

(Migrate stored_outlines: `DBMS_SPM.Migrate_Stored_Outline`)

Baseline capture

- Capture in sessions works somewhat sophisticated
  - SQL needs to be *executed* twice in order to be captured
  - To prevent lots of useless plans for SQL with literals
  - SYS.SQLLOG$ keeps log of first executions

- During capture baseline properties are set:
  - Enabled true/false
  - Accepted true/false
  - Fixed true/false

Auto-capture

Parse of sql1

Exec. Plan plA1

- Sql1 parsed 1st time
- Not yet baselined
- “Marked”: added to sqllog$
- plA1 used

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed

Auto-capture

Parse of sql1

- Sql1 parsed 1st time
- Not yet baselined
- “Marked”: added to sqllog$
- plA1 used

Exec. Plan plA1

2nd exec of sql1

- 2nd execution detected
- plA1 now stored as accepted plan

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed

Auto-capture

Parse of sql1

- Sql1 parsed 1st time
- Not yet baselined
- "Marked": added to sqllog$
- plA1 used

Exec. Plan plA1

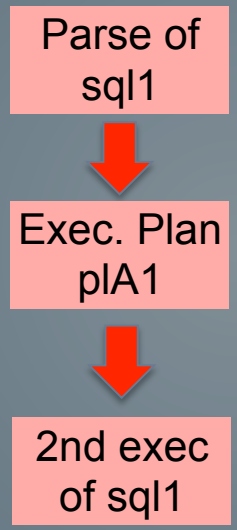
2nd exec of sql1

- 2nd execution detected
- plA1 now stored as accepted plan

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed
sql1	plA1	Y	Y	N

Auto-capture



- Sql1 parsed 1st time
- Not yet baselined
- "Marked": added to sqllog$
- plA1 used

- 2nd execution detected
- plA1 now stored as accepted plan

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed
sql1	plA1	Y	Y	N

- From now on: future parses of sql1 will use plA1

Auto-capture: introducing more plans…

Parse of sql1

- Sql1 parsed
- Already baselined

Exec. Plan plA2

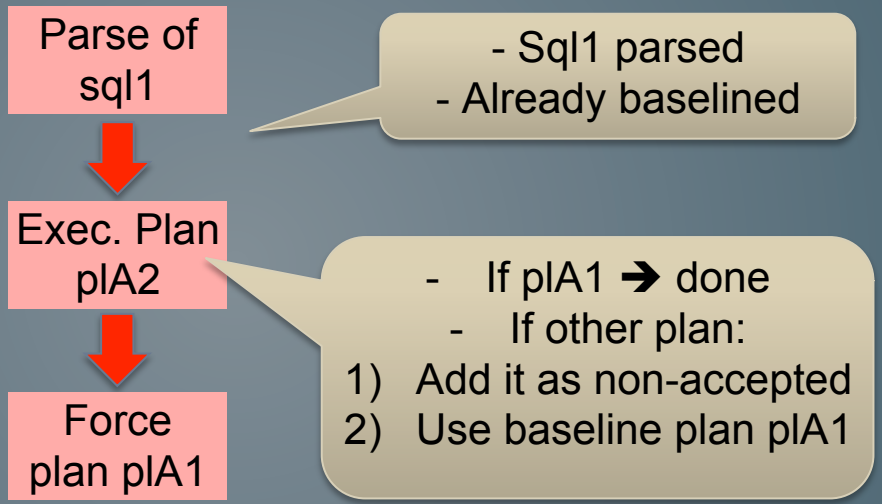
- If plA1 ➔ done
- If other plan:

1) Add it as non-accepted
2) Use baseline plan plA1

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed
sql1	plA1	Y	Y	N

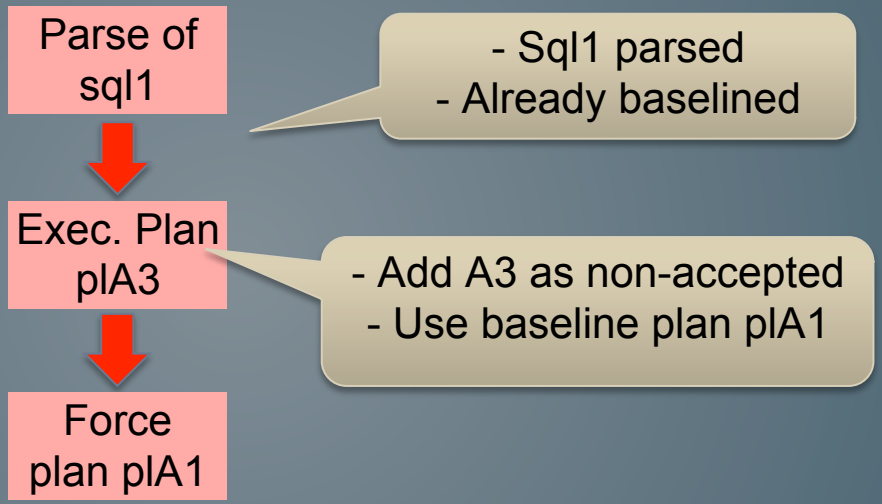
Auto-capture: introducing more plans…



DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed
sql1	plA1	Y	Y	N
sql1	plA2	**N**	Y	N

Auto-capture: introducing more plans…



DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed
sql1	plA1	Y	Y	N
sql1	plA2	**N**	Y	N
sql1	plA3	**N**	Y	N

Baseline evolution

- Evolution = find out if not yet accepted plans can be turned into accepted plans
- How?
  - By executing them and comparing performance with already accepted plans in the baseline
  - `DBMS_SPM.Evolve_Sql_Plan_Baseline`

Again, not well documented. How is comparison done in case of multiple already accepted plans available?

Baseline evolution

- `DBMS_SPM.Evolve_Sql_Plan_Baseline`
- Options:
  - Run Evolve and accept if performance is better
  - Run Evolve and report only (do not accept)
  - Run Evolve and accept without testing performance

Will show Evolve run later on…

Baseline selection

- **`Fixed = true`** changes selection and capture process
  - You can set this manually
  - If exists a fixed & enabled plan for the SQL
    1. Auto-capture will not add new not-accepted plans for this SQL
    2. Selection will choose fixed/enabled plan for this SQL
       - Again can be multiple plans…

Not well documented. How is selection performed in this case?

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed
sql1	plA1	Y	Y	N
sql1	plA2	Y	Y	**Y**
sql1	plA3	Y	Y	N

Baseline selection

- `Enabled = false` changes selection and evolution process
  - You can set this manually
  - Disabled plans not considered for
    1. Selection
    2. Evolution

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	ena	fixed
sql1	plA1	Y	Y	N
sql1	plA2	Y	Y	N
sql1	plA3	Y	**N**	N

SPM specific configuration

- Repository storage in SYSAUX
- Two additional (not init.ora) parameters:
  - **Space_budget_percent = 30**
  - **Plan_retention_weeks = 53**
- To monitor storage usage in SYSAUX tablespace
- To purge baseline plans not used x weeks
- Set via DBMS_SPM.**Configure**
- Query via **DBA_SQL_MANAGEMENT_CONFIG**

Any questions sofar?

Popular questions

1. What if an accepted baseline plan is “no longer valid”?
2. How is baseline plan chosen in case multiple plans exist?
3. How is baseline plan evolution performed?

Q1: Baseline invalidation

- What if:
  - Accepted plan relies on index
  - Index no longer exists due to software changes
- How does SPM deal with this situation?

Q1: Baseline invalidation

- A baseline plan does *not* hold execution plan, but
  - set of outline hints
    - When injected in sql-text, are supposed to produce actual execution plan
  - plan-hash-id
    - Of plan supposed to be produced by outline hints
      - Ie. plan that was produced when this baseline was captured
- Upon plan selection
  - Hints are applied to reproduce plan
  - This plan is hashed
  - This hash is compared against stored hash
  - If unequal ➔ Plan is discarded

Q1: Baseline invalidation

Parse of sql1

Exec. Plan plA2

- This plan relies on some index
- Index no longer exists

- Hard parse of sql1 now results in new plan (without the index)

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	hsh	repro
sql1	plA1	Y	x1	Y

Q1: Baseline invalidation

Parse of sql1

↓

Exec. Plan plA2

↓

Exec. Plan plA1

- As usual, new plan is added as non-accepted plan
- And SPM forces use of plA1

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	hsh	repro
sql1	plA1	Y	x1	Y
sql1	plA2	N	x2	Y

Q1: Baseline invalidation

Parse of sql1

↓

Exec. Plan plA2

↓

Exec. Plan plA1

↓

Exec. Plan plA2

- SPM tries to reproduce plan plA1, and fails. This is detected by mismatch in hash
- SPM marks reproduced = 'N' and switches back to CBO plan

DBA_SQL_PLAN_BASELINES

SQL	plan	acc	hsh	repro
sql1	plA1	Y	x1	N
sql1	plA2	N	x2	Y

Popular questions

1. What if an accepted baseline plan is “no longer valid”?
2. How is baseline plan chosen in case multiple plans exist?
3. How is baseline plan evolution performed?

Q2: Baseline selection

- Reminder:
  - If hard parse results in plan X, and X is one of accepted (and reproducable!) plans, then plan X is obviously used
- But, what if:
  - Hard parse produces new plan, and
  - Multiple (other) accepted and reproducable plans exist
- How does SPM choose which plan to use in these cases?
  - Does it take different opt-env into account?
  - [Does it take bind-var values into account?]

Q2: Baseline selection

- Experiment specification:
  - ALL_ROWS plan
  - FIRST_ROWS_1 plan

```
select pk,vc1
from spm_test
order by vc1;
```

```
SQL> desc spm_test
 Name                      Null?    Type
 ------------------------- -------- -------------
 PK                        NOT NULL NUMBER
 VC1                       NOT NULL VARCHAR2(4)
 VC2                       NOT NULL VARCHAR2(4)
 PADDING                   NOT NULL VARCHAR2(100)

Two indexes: (PK), (VC1)
```

```
-------------------------------
| Operation          | Name   |
-------------------------------
| SELECT STATEMENT   |        |
|  SORT ORDER BY     |        |
|   TABLE ACCESS FULL| SPM_TEST |
-------------------------------
```

```
-----------------------------------------------
| Operation                    | Name         |
-----------------------------------------------
| SELECT STATEMENT             |              |
|  TABLE ACCESS BY INDEX ROWID| SPM_TEST     |
|   INDEX FULL SCAN            | SPM_TEST_VC1 |
-----------------------------------------------
```

Q2: Baseline selection

```
select pk,vc1
from spm_test
order by vc1;
```

- Both plans are available in baseline (accepted)
- New index I3 on (VC1,PK)
- Parse will now produce new plan ➔
  - In both modes (AR/FR1)

```
--------------------------------
| Operation         | Name       |
--------------------------------
| SELECT STATEMENT  |            |
|   INDEX FULL SCAN | SPM_TEST_I3 |
--------------------------------
```

- Hypothesis: we have plan stability, so …
  - SPM chooses first_rows_1 baseline, when in first_rows_1 mode
  - SPM chooses all_rows baseline, when in all_rows mode

Following test performed on 11.2.0.2 and 11.2.0.3

Q2: Baseline selection (script: demo01.txt)

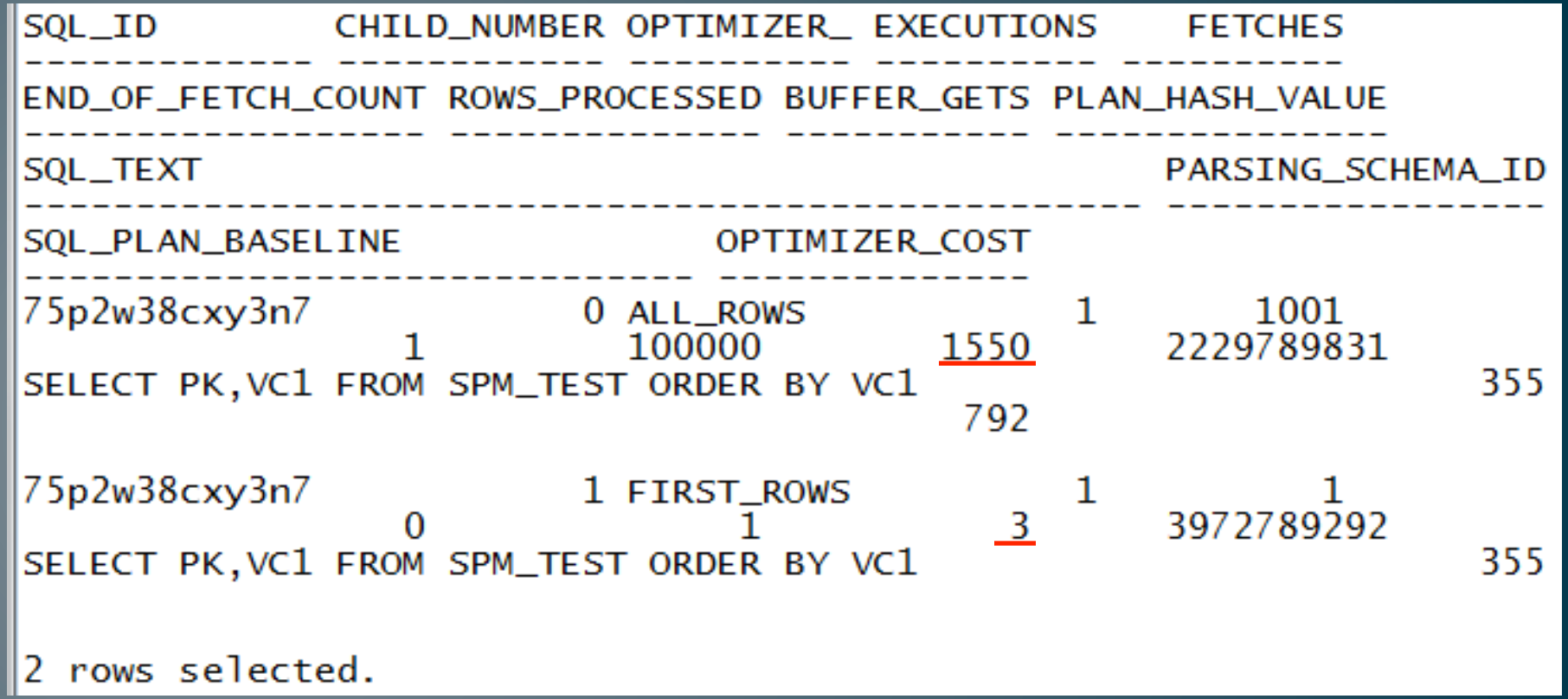
```
SQL> alter system flush shared_pool;

System altered.

SQL> delete from sys.sqllog$;

0 rows deleted.

SQL> show parameter basel
More...

NAME                                 TYPE  VALUE
------------------------------------ ----  ------------------
optimizer_capture_sql_plan_baselines bool  FALSE
                                     ean
optimizer_use_sql_plan_baselines     bool  TRUE
                                     ean
SQL>
```

```
Rem
Rem Load two child-cursors into the shared_pool for our query.
Rem
alter session set optimizer_mode=all_rows;

begin
   --
   for r in (select pk,vc1 from spm_test order by vc1)
   loop
      --
      null;
      --
   end loop;
   --
end;
/
```

Cursor fetches all rows

```
alter session set optimizer_mode=first_rows_1;

declare
cursor c1 is select pk,vc1 from spm_test order by vc1;
r1 c1%rowtype;
begin
   --
   open c1;
   fetch c1 into r1;
   close c1;
   --
end;
/
```

Cursor fetches first row only



```
SQL_ID        CHILD_NUMBER OPTIMIZER_ EXECUTIONS    FETCHES
------------- ------------ ---------- ---------- ----------
END_OF_FETCH_COUNT ROWS_PROCESSED BUFFER_GETS PLAN_HASH_VALUE
------------------ -------------- ----------- ---------------
SQL_TEXT                                                 PARSING_SCHEMA_ID
-------------------------------------------------------- -----------------
SQL_PLAN_BASELINE              OPTIMIZER_COST
------------------------------ --------------
75p2w38cxy3n7             0 ALL_ROWS              1       1001
                1          100000         1550      2229789831
SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1                          355
                                        792

75p2w38cxy3n7             1 FIRST_ROWS            1          1
                0             1              3      3972789292
SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1                          355


2 rows selected.
```

Query V$SQL

```
SQL> variable num_loaded number;
SQL> exec :num_loaded := DBMS_SPM.Load_Plans_From_Cursor_Cache -
> (sql_id=>'75p2w38cxy3n7',plan_hash_value=>2229789831,fixed=>'NO',enabled=>'YES');

PL/SQL procedure successfully completed.

SQL> print num_loaded
More...

NUM_LOADED
----------
         1

SQL> exec :num_loaded := DBMS_SPM.Load_Plans_From_Cursor_Cache( -
> sql_id=>'75p2w38cxy3n7',plan_hash_value=>3972789292,fixed=>'NO',enabled=>'YES');

PL/SQL procedure successfully completed.

SQL> print num_loaded
More...

NUM_LOADED
----------
         1

SQL>
```

Load both plans into the SPM repository

```
SQL> l
  1  select sql_handle, sql_text, plan_name, enabled, accepted,parsing_s
  2         ,optimizer_cost,fetches,end_of_fetch_count,rows_processed
  3  from dba_sql_plan_baselines
  4* order by sql_handle,plan_name
SQL> /
More...

SQL_HANDLE           SQL_TEXT
-------------------- --------------------------------------------------
PLAN_NAME                      ENA ACC PARS OPTIMIZER_COST    FETCHES
------------------------------ --- --- ---- -------------- ----------
END_OF_FETCH_COUNT ROWS_PROCESSED
------------------ --------------
SQL_481455a48a6bbf6d SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1
SQL_PLAN_4h52pnk56rgvd0eaebc72 YES YES SPM1                3           1
                   0                1

SQL_481455a48a6bbf6d SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1
SQL_PLAN_4h52pnk56rgvdd6d8fb93 YES YES SPM1              792        1001
                   1           100000


2 rows selected.

SQL>
```

```
SQL> alter session set optimizer_mode=all_rows;

Session altered.

SQL> explain plan for select pk,vc1 from spm_test order by vc1;

Explained.

SQL> @vp
More...

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2229789831

-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |   100K|   878K|       |   792   (1)|
|   1 |  SORT ORDER BY     |          |   100K|   878K|  1976K|   792   (1)|
|   2 |   TABLE ACCESS FULL| SPM_TEST |   100K|   878K|       |   410   (1)|
-------------------------------------------------------------------------------

Note
-----
   - SQL plan baseline "SQL_PLAN_4h52pnk56rgvdd6d8fb93" used for this statement

13 rows selected.
```

```
SQL> alter session set optimizer_mode=first_rows_1;

Session altered.

SQL> explain plan for select pk,vc1 from spm_test order by vc1;

Explained.

SQL> @vp
More...

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3972789292

-------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     1 |     9 |     3   (0)|
|   1 |  TABLE ACCESS BY INDEX ROWID| SPM_TEST     |   100K|   878K|     3   (0)|
|   2 |   INDEX FULL SCAN            | SPM_TEST_VC1 |     1 |       |     2   (0)|
-------------------------------------------------------------------------------

Note
-----
   - SQL plan baseline "SQL_PLAN_4h52pnk56rgvd0eaebc72" used for this statement

13 rows selected.
```

```
Rem
Rem Now introduce a new index.
Rem
create unique index spm_test_i3 on spm_test(vc1,pk);

begin
  dbms_stats.gather_table_stats(
    ownname => USER ,
    tabname => 'SPM_TEST' ,
    estimate_percent => 100 ,
    cascade => true);
end;
/
```

```
SQL> alter system flush shared_pool;

System altered.

SQL> delete from sys.sqllog$;

0 rows deleted.
```

```
SQL> alter session set optimizer_mode=all_rows;

Session altered.

SQL> begin
  1     --
  2     for r in (select pk,vc1 from spm_test order by vc1)
  3     loop
  4       --
  5       null;
  6       --
  7     end loop;
  8     --
  9   end;
 10 /

PL/SQL procedure successfully completed.

SQL> /

PL/SQL procedure successfully completed.
```

2nd time required for it to show up in V $SQL

```
SQL_HANDLE           SQL_TEXT
-------------------- ------------------------------------------------------------
PLAN_NAME                      ENA ACC PARS OPTIMIZER_COST    FETCHES
------------------------------ --- --- ---- -------------- ----------
END_OF_FETCH_COUNT ROWS_PROCESSED
------------------ --------------
SQL_481455a48a6bbf6d SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1
SQL_PLAN_4h52pnk56rgvd0eaebc72 YES YES SPM1                3           1
                   0                 1

SQL_481455a48a6bbf6d SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1
SQL_PLAN_4h52pnk56rgvdbce7d0d6 YES NO  SPM1              264           0
                   0                 0

SQL_481455a48a6bbf6d SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1
SQL_PLAN_4h52pnk56rgvdd6d8fb93 YES YES SPM1              792        1001
                   1            100000


3 rows selected.
```

Baselines after this execution

```
SQL_ID        CHILD_NUMBER OPTIMIZER_ EXECUTIONS    FETCHES
------------- ------------ --------- ---------- ----------
END_OF_FETCH_COUNT ROWS_PROCESSED BUFFER_GETS PLAN_HASH_VALUE
------------------ -------------- ----------- ---------------
SQL_TEXT                                                 PARSING_SCHEMA_ID
-------------------------------------------------------- -----------------
SQL_PLAN_BASELINE                OPTIMIZER_COST
-------------------------------- --------------
75p2w38cxy3n7             2 FIRST_ROWS           1       1001
                1          100000       101206       3972789292
SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1                          355
SQL_PLAN_4h52pnk56rgvd0eaebc72                3


1 row selected.
```

Query V$SQL

Q2: Baseline selection

SQL Plan Management ensures that runtime performance never degrades due to the change of an execution plan.

Broken…

- We introduce a new index … ROWS baselined SQL starts performing wo…

- Hear-say:
  - We re-cost all available accepted plans using optimizer-env stored against them
  - And then choose one with lowest cost
    - Should be: using the current optimizer-env

Q2: Baseline selection

- Final remark on this experiment:
  - Instability will of course be fixed, once new plan is evolved…

Popular questions

1. What if an accepted baseline plan is “no longer valid”?
2. How is baseline plan chosen in case multiple plans exist?
3. How is baseline plan evolution performed?

Q3: How is evolution performed?

- Let's rewind our previous experiment and now use auto-capture
  - Run query in all_rows mode ➔ first accepted baseline created
  - Run query in first_rows_1 mode ➔ second, to be evolved baseline created
  - Manually evolve it
- Script demo02.txt
  - See spm1.prf

```
SQL> drop index spm_test_i3;

Index dropped.

SQL> declare
  1  pl_num number;
  2  begin
  3    --
  4    for r in (select sql_handle, plan_name
  5                from dba_sql_plan_baselines)
  6    loop
  7      --
  8      pl_num := dbms_spm.drop_sql_plan_baseline(r.sql_handle,r.plan_name);
  9      --
 10    end loop;
 11    --
 12  end;
 13  /

PL/SQL procedure successfully completed.

SQL> alter system flush shared_pool;

System altered.

SQL> delete from sys.sqllog$;

0 rows deleted.

SQL>
```

```
Rem
Rem Load two child-cursors into the shared_pool for our query.
Rem
alter session set optimizer_mode=all_rows;

begin
  --
  for r in (select pk,vc1 from spm_test order by vc1)
  loop
    --
    null;
    --
  end loop;
  --
  for r in (select pk,vc1 from spm_test order by vc1)
  loop
    --
    null;
    --
  end loop;
  --
end;
/
```

```
alter session set optimizer_mode=first_rows_1;

declare
cursor c1 is select pk,vc1 from spm_test order by vc1;
r1 c1%rowtype;
begin
   --
   open c1;
   fetch c1 into r1;
   close c1;
   --
   open c1;
   fetch c1 into r1;
   close c1;
   --
end;
/
```

```
SQL_HANDLE            SQL_TEXT
--------------------- ------------------------------------------------------------
PLAN_NAME                       ENA ACC PARS OPTIMIZER_COST    FETCHES
------------------------------- --- --- ---- -------------- ----------
END_OF_FETCH_COUNT ROWS_PROCESSED
------------------ --------------
SQL_481455a48a6bbf6d SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1
SQL_PLAN_4h52pnk56rgvd0eaebc72 YES NO  SPM1                3          0
                   0                 0

SQL_481455a48a6bbf6d SELECT PK,VC1 FROM SPM_TEST ORDER BY VC1
SQL_PLAN_4h52pnk56rgvdd6d8fb93 YES YES SPM1              792          0
                   0                 0


2 rows selected.
```

Two baselines, first one got accepted, second one to be evolved

```
Rem
Rem Evolve not accepted plans for our SQL.
Rem
set long 10000

variable c1 clob;

begin
  --
  :c1 := DBMS_SPM.evolve_sql_plan_baseline
           (sql_handle => 'SQL_481455a48a6bbf6d'
               ,verify => 'YES'
               ,commit => 'YES');
  --
end;
/

print c1
```

C1 contains report of evolve proces result

```
                    Evolve SQL Plan Baseline Report
--------------------------------------------------------------------------------
Inputs:
-------
  SQL_HANDLE = SQL_481455a48a6bbf6d
  PLAN_NAME  =
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = YES
  COMMIT     = YES

Plan: SQL_PLAN_4h52pnk56rgvd0eaebc72
------------------------------------
  Plan was verified: Time used 2.28 seconds.
  Plan failed performance criterion: 2.33 times worse than baseline plan.

                          Baseline Plan      Test Plan    Stats Ratio
                          -------------      ---------    -----------
  Execution Status:             COMPLETE       COMPLETE
  Rows Processed:                 100000         100000
  Elapsed Time(ms):              119.622        288.383           .41
  CPU Time(ms):                  112.411         261.96           .43
  Buffer Gets:                      1480         100209           .01
  Physical Read Requests:              0              0
  Physical Write Requests:             0              0
  Physical Read Bytes:                 0              0
  Physical Write Bytes:                0              0
  Executions:                          1              1

Number of plans verified: 1
Number of plans accepted: 0
```

```
********************************************************************************

SQL ID: d56x858gwqp8v Plan Hash: 3604544403

SELECT PK,VC1
FROM
 SPM_TEST ORDER BY VC1


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute     10      0.00       0.00          0          0          0           0
Fetch       10      0.95       1.00          0      14800          0     1000000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       21      0.95       1.00          0      14800          0     1000000

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 355     (recursive depth: 1)
********************************************************************************
```

Evolve proces ran the accepated baseline 10 times

```
********************************************************************************

SQL ID: d56x858gwqp8v Plan Hash: 246332530

SELECT PK,VC1
FROM
 SPM_TEST ORDER BY VC1


call     count       cpu    elapsed                    ery    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00                    0          0           0
Execute      6      0.00                               0          0           0
Fetch        6      1.09                               0     601254           0      600000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       13      1.09                               0     601254           0      600000

Misses in library cache during parse: 1
Optimizer mode: FIRST_ROWS
Parsing user id: 355     (recursive depth: 1)
********************************************************************************
```

"This is known issue in fw from other group"

And the to-be-accepted was run 6 times
5th run was still better, 6th run nomore

But it fetched till %NOTFOUND here too....

Q3: How is evolution performed?

- Forced to manually accept the 2nd plan (verify => no)
- Now if we were to:
  - Introduce the new index plan and try to evolve that too
    - Note: this plan is applicable for both optimizer-modes
  - New plan becomes accepted for the wrong reason
    - Continue in script demo02.txt
    - Spm2.prf
- Not really fair, as we're already 'passed' something broken

Q3: How is evolution performed?

- Other thoughts:
  - What if data is distributed differently at time of evolve?
  - How about baselines for DML (insert/update/delete/merge/mti)
    - Presumably rolled back after 10 evolve executions
    - Better not have triggers with non-transactional side-effects
  - ***No: only query part of DML is executed on evolve***

Other nice-to-knows

- SPM puts break on other optimizer features such as cardinality feedback and adaptive cursor sharing
  - New plans generated by these features will have to wait till the next evolution

- Baseline for SQL parsed under schema S1, can be selected for same SQL parsed under schema S2
  - http://intermediatesql.com/oracle/oracle-11g-sql-plan-management-the-dark-side-of-spm-part-4/

Recap: SPM version 1.0

- Hard parse SQL ➔ produces CBO-plan
- SQL is hashed
- CBO-plan is hashed
- Two hashes are used to search for accepted baseline
  - If found, use it
  - If not found, choose the one with the least cost (?)
    - This can be one parsed under different schema
    - Or one using different optimizer environment
    - Or (possibly, not tested) one having mismatch in some other area
      - See: *_mismatch columns in v$sql_shared_cursor

Possible mismatches?

- OPTIMIZER_MISMATCH
- OUTLINE_MISMATCH
- STATS_ROW_MISMATCH
- LITERAL_MISMATCH
- FORCE_HARD_PARSE
- EXPLAIN_PLAN_CURSOR
- BUFFERED_DML_MISMATCH
- PDML_ENV_MISMATCH
- INST_DRTLD_MISMATCH
- SLAVE_QC_MISMATCH
- TYPECHECK_MISMATCH
- AUTH_CHECK_MISMATCH
- BIND_MISMATCH
- DESCRIBE_MISMATCH
- LANGUAGE_MISMATCH
- TRANSLATION_MISMATCH
- BIND_EQUIV_FAILURE
- INSUFF_PRIVS
- INSUFF_PRIVS_REM
- REMOTE_TRANS_MISMATCH
- LOGMINER_SESSION_MISMATCH
- INCOMP_LTRL_MISMATCH
- OVERLAP_TIME_MISMATCH
- EDITION_MISMATCH
- MV_QUERY_GEN_MISMATCH
- USER_BIND_PEEK_MISMATCH
- TYPCHK_DEP_MISMATCH
- NO_TRIGGER_MISMATCH

Wrapping up

- Provided high-level tutorial
  - Still many more details to it: read docs, blogs, google for it
- Answer some of obvious questions not found in docs
  - Hope to not have scared you away from SPM
  - Issues presented seem to be easily fixable by Oracle
  - And/or may not be applicable in your environment
- Make you think about this feature

One last slide on 12c enhancements…

Wrapping up: 12c enhancements

- New evolve auto task: **`sys_auto_spm_evolve_task`**
  - Info in **`dba_advisor_tasks`**, and via **`dbms_spm.report_auto_evolve_task`**
  - Requires Tuning Pack
- SPM evolve now works with advisory task infrastructure
  - EM integration, persistent store of evolution reports
- Next to plan-hash, plan rows now also stored in repository
  - Easier diagnosability in case plan could not be reproduced

Q & A

toan.kappelacari@oracle.com

Thank you for your attention

https://www.surveymonkey.com/s/hotsym2013

Only takes 6 clicks

Toon.koppelaars@oracle.com

```
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)
 REPRODUCED                         VARCHAR2(3)
 AUTOPURGE                          VARCHAR2(3)
 OPTIMIZER_COST                     NUMBER
 MODULE                             VARCHAR2(64)
 ACTION                             VARCHAR2(64)
 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
```