VISCOSITY
NORTH AMERICA

On the Seventh Day of 18c/19c, Viscosity Gave to me...

SQL Plan Management

December 16, 2020

Today, we are going to talk about **SQL Plan Management or SPM,** which is a preventative mechanism to help stabilize SQL performance in Oracle databases. SPM is part of the base product as of version 11.1, but has added many features in Oracle 12c / 18c / 19c. Utilizing SPM allows for management of execution plans ensuring that the database uses only known or verified plans; and can help prevent performance issues caused by SQL plan changes. The main steps of using SPM include:

- Plan capture – storing relevant information about plans for a set of SQL statements
- Plan selection – the optimizer identifies plan changes based on stored plans history, and uses accepted SQL plan baselines to maintain SQL performance
- Plan evolution - accepting new plans in existing baselines, either manually or automatically, normally after verifying that the new plan performs well

We don't have room here to cover every detail of SPM, so let's do a simple example of capturing a baseline from the cursor cache, and then seeing that the baseline is being used. *Note: plans or baselines can also be captured automatically.*

By default, Oracle will use SQL Plan baselines if they exist and are accepted. Default behavior for this has not changed since the 11gR1 version of Oracle. This can be adjusted by setting the parameter `OPTIMIZER_USE_SQL_PLAN_BASELINES` to `FALSE` which will disable the usage. Also, it should be noted that this parameter is independent of capturing baselines.

Here we have a simple query joining two tables with the same set of data.

```
SQL_ID  adjwa8r407rzc, child number 0
-------------------------------------
select /*+ gather_plan_statistics */ sum(t1.c), sum(t2.c)
  from   t1, t2
 where  t1.a = t2.a and    t1.d = :idnum
```

The data in the table has 25,001 duplicate rows, and 24,999 unique rows. Not to get into too much detail, but this leads to the possibility of two plans. One for looking up the duplicate rows and one for the unique rows.

```
SQL_ID        PLAN_HASH_VALUE CHILD_NUMBER
------------- --------------- ------------
adjwa8r407rzc      3534348942            0
adjwa8r407rzc       906334482            1
```

Technology Overview

Database: 18c / 19c

Company: Oracle Corporation

Topic: SQL Plan Management

Viscosity can help with any of your Database Upgrade needs

Viscosity has performed numerous zero-downtime database migrations and upgrades over the years and has a proven track record with business critical and mission critical databases.

Viscosity's Database Migration & Upgrade Services can plan, upgrade, validate and migrate all database content - quickly and effectively with our automated approach and proven methodology.

Learn more about how you can maintain and maximize your investments at viscosityna.com or email us at hello@viscosityna.com.

www.viscosityna.com

Let's create a SQL Plan Baseline on the first plan, so that the optimizer only considers that plan as good. We will use the LOAD_PLANS_FROM_CURSOR_CACHE functions of the DBMS_SPM package.

```
VARIABLE v_plan_cnt NUMBER
EXECUTE :v_plan_cnt := DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE(
      sql_id => 'adjwa8r407rzc',
      plan_hash_value => 3534348942);
```

We can see that one plan was stored as a baseline:

```
SQL> select :v_plan_cnt from dual;

:V_PLAN_CNT
-----------
          1
```

Let's also check that a baseline was created:

```
SELECT PLAN_NAME, SQL_HANDLE, ENABLED,
       FIXED, OPTIMIZER_COST,
       EXECUTIONS, ELAPSED_TIME,
       CPU_TIME, BUFFER_GETS,
       DISK_READS
  FROM DBA_SQL_PLAN_BASELINES;

PLAN_NAME                      SQL_HANDLE           ENA FIX COST EXE EL_TIME CPU_TIME BUFGET DR
------------------------------ -------------------- --- --- ---- --- ------- -------- ------ --
SQL_PLAN_ct5006ahu8hpbceb2ac1e SQL_cc940032a1a442ab YES NO   553   2 1137796   615734  13070  1
```

Note: the baseline is enabled by default, but not fixed. This happens when we manually load a baseline. As the baseline is not fixed, Oracle will attempt to capture new plans as baselines for this statement. To prevent this, we will FIX the baseline so that no other plans are considered.

Fixing a plan is done by using the ALTER_SQL_PLAN_BASELINE function of the DBMS_SPM package.

```
SET SERVEROUTPUT ON
DECLARE
  v_dplans number;
BEGIN
  v_dplans :=
    DBMS_SPM.ALTER_SQL_PLAN_BASELINE(
      SQL_HANDLE => 'SQL_28d363dc39b4d314',
      PLAN_NAME =>  'SQL_PLAN_2jnv3vhwv9nsnc6a45b88',
      ATTRIBUTE_NAME => 'fixed',
      ATTRIBUTE_VALUE => 'YES');

  DBMS_OUTPUT.PUT_LINE( 'fixed ' || v_dplans || ' plans');
```

```
END;
/

SQL> fixed 1 plans
```

The function returns the number of plans that were modified. If you do not provide a plan name, then all plans for the given SQL_HANDLE will be affected.

Note: to remove the fixed status, set the ATTRIBUTE_VALUE *to* NO.

Now, let's see if new baselines are being used. The best way to do this is to query the DBA_SQL_PLAN_BASELINES view. This view has columns that show the origin of the SQL Baseline, the status, when it was last executed, and when it was verified. An example query:

```
SELECT b.plan_name, s.sql_id, s.executions,
       TO_CHAR(s.last_active_time, 'YYYYMMDD HH24:MI:SS') last_active_time,
       b.sql_handle, b.creator, b.origin,
       TO_CHAR(b.created, 'YYYYMMDD HH24:MI:SS') created,
       TO_CHAR(b.last_executed, 'YYYYMMDD HH24:MI:SS') baseline_executed,
       TO_CHAR(b.last_verified, 'YYYYMMDD HH24:MI:SS') baseline_verified,
       b.enabled, b.accepted, b.fixed
  FROM dba_sql_plan_baselines b, v$sql s
  WHERE s.sql_plan_baseline (+) = b.plan_name
    AND s.exact_matching_signature (+) = b.signature
      -- s.force_matching_signature (+) = b.signature
  ORDER BY 5 DESC, 6 DESC;
```

Note: depending on your database setting you will need to join on V$SQL *either using* FORCE *or* EXACT *signature. Also, the outer join on the* V$SQL *table as the statement may not be in your cursor cache.*

Abridged output from this query shows the last time the baseline was used for execution:

```
SQL_ID        EXECUTIONS LAST_ACTIVE_TIME  SQL_HANDLE
------------- ---------- ----------------- ----------------------
adjwa8r407rzc          2 20201211 15:36:57 SQL_cc940032a1a442ab
```

You can also see if the plan is being used by checking the NOTES section of an explain plan:

```
SQL_ID  adjwa8r407rzc, child number 2
-------------------------------------
select /*+ gather_plan_statistics */ sum(t1.c), sum(t2.c)
  from   t1, t2
  where  t1.a = t2.a and    t1.d = :idnum

Plan hash value: 3534348942
```

```
-------------------------------------------------------------------------------
--------
| Id  | Operation                     | Name | Starts | E-Rows |E-Bytes| Cost
(%CPU)|
-------------------------------------------------------------------------------
--------
|   0 | SELECT STATEMENT              |      |      1
|       |       |   741K(100)|
|   1 |  SORT AGGREGATE               |      |      1 |      1 |    22
|           |
|   2 |   NESTED LOOPS                |      |      1
|    246K|  5305K|   741K  (1)|
|   3 |    NESTED LOOPS               |      |      1
|    246K|  5305K|   741K  (1)|
|*  4 |     TABLE ACCESS FULL         | T1   |      1
|    246K|  3134K|   550   (1)|
|*  5 |     INDEX RANGE SCAN          | T2I  |      1 |      1
|       |     2   (0)|
|   6 |    TABLE ACCESS BY INDEX ROWID| T2   |      1 |      1 |     9
|     3   (0)|
-------------------------------------------------------------------------------
--------

Note
-----
   - SQL plan baseline SQL_PLAN_ct5006ahu8hpbceb2ac1e used for this
statement
```

Summary

SQL plan management can improve or preserve performance during database upgrades, system, and data changes. A database upgrade that installs a new optimizer version, usually results in plan changes for a small percentage of SQL statements. Using SPM, you can stabilize plan regression issues.

Hopefully this example gets you thinking about how you can use SPM in your environment. Look for upcoming presentation with Viscosity, where we will dive deeper into the many features of SPM both manual and automated.

One of the sayings we have at Viscosity is our customer's, "have four aces in their pocket". Over the next 5 days, the talented staff at Viscosity along with our Oracle ACEs will address more Oracle Database 18c and 19c new features. Continue to join us next year, as we continue our Oracle Database 19c hands-on-lab workshops.

Happy Holidays!