# Oracle 11g 针对SQL性能的新特性（三）- SQL Plan Management

By: [Guest Author](https://blogs.oracle.com/database4cn/guest-contributor)

## SQL Plan Management (SPM)

**历史**

SQL的执行效率，取决于它的执行计划是否高效。 优化器的算法是一个平衡，需要收集尽量少的信息，用尽量快的速度试图去得到一个最优的执行计划，这也决定了它不是万能的。 所以Oracle提供了一些辅助手段来“修复”优化器可能产生的错误，并不断改进这些方法。   
  
 Oracle 8： hint  
 Oracle 8i&9: stored outline  
 Oracle 10: sql profile  
 Oracle 11: sql plan manangement

**简介**

在Oracle 11g之前，执行计划一直是作为“运行时”生成的对象存在。虽然oracle提供了一些方法去指导它的生成，但Oracle一直没有试图去保存完整的执行计划。 从11g开始，执行计划就可以作为一类资源被保存下来，允许特定SQL语句只能选择“已知”的执行计划。

同其他方法相比，SPM更加的灵活。如我们所熟知的，一条带有绑定变量的SQL语句，最好的执行计划会根据绑定变量的值而不同，11g以前的方法都无法解决这个问题。在11g中，与adaptive cursor sharing配合，SPM允许你同时接受多个执行计划。执行时，根据不同的变量值，SPM会花费很少的运算从中选择一条最合适的。

**概念**

SQL Plan Management SPM：oracle 11g 中提供的新特性，用来更好地控制执行计划。   
 Plan History： 优化器生成的所有执行计划的总称  
 SQL Plan Baseline： Plan History里那些被标记为“ACCEPTED”的执行计划的总称  
 Plan Evolution： 把一条执行计划从Plan History里标记为“ACCEPTED”的过程  
 SQL Management Base SMB： 字典表里保存的执行计划的总称，包括Plan History，SQL Plan Baseline和SQL profile。

**SPM的特点**

o 与profile和outline相比，更加灵活的控制手段  
   + 可以有很多的计划被保存下来，只有"ENABLED"并且"ACCEPTED"的执行计划才可以被选择。   
   + 允许有多个"ACCEPTED"的执行计划，根据实际情况进行选择。   
   + 可以用手工或者自动的方式，把执行计划演化(evolve)为"ACCEPTED"。 还可以控制只让性能更好的计划被接受。  
   + 允许设置"FIXED"的计划。这样其他的计划将不会被选择。

o SPM使计划真正的稳定。 outline的缺点是太过死板，当数据量大幅度变化时无法做出相应的改变。 SQL proifle的缺点是，当数据量变化时，STA(SQL Tuning Advisor)会不可预知地去更改执行计划。 而SPM则会提供几个完整的plan供选择。

**SPM的控制方式**

SPM通过几个标记来实现对执行计划的控制：

o Enabled (控制活动)  
   + YES (活动的，但不一定会被使用)  
   + NO (可以理解为被标记删除)  
 o Accepted (控制使用)  
   + YES (只有 “Enabled” 并且 “Accepted” 的计划才会被选择使用)  
   + NO (如果是“Enabled” 那么只有被evolve成“Accepted”才有可能被执行)  
 o Fixed (控制优先级)  
   + YES (如果是“Enabled”并且“Accepted”，会优先选择这个计划，这个计划会被视为不需要改变的)  
   + NO (普通的计划，无需优先)  
  
 另有一个被动的标记：  
 o Reproduced (有效性)  
   + YES (优化器可以使用这个计划)  
   + NO (计划无效，比如索引被删除)

**SPM如何捕捉执行计划**

o 自动捕捉  
   1. 首先把OPTIMIZER\_CAPTURE\_SQL\_PLAN\_BASELINES设置成TRUE  
   2. 从这个时刻开始，所有执行两次以上的SQL语句会被观测，执行计划会进入Plan History。有个别例外的，参见note 788853.1  
   3. 生成的第一个执行计划被标记为ENABLED并且是ACCEPTED，后续的执行计划会被标记为ENABLED但不是ACCEPTED。  
   4. 这时把OPTIMIZER\_CAPTURE\_SQL\_PLAN\_BASELINES设置会FALSE，新的语句将不会创建Baseline。  
   5. 需要注意的是，即使关闭了自动捕捉，针对存在baseline的SQL，由于ACS的作用，仍旧会有新的PLAN生成，新的Plan仍会进入Plan History，标记为ENABLED但不是ACCEPTED。参见“执行计划的选择”。

o 批量导入 (这些导入的baseline都会被自动标记为ACCEPTED)   
    Oralce提供四种方式把计划导入到sql plan baseline中。   
    + 从 SQL Tuning Set STS 导入  
       DBMS\_SPM.LOAD\_PLANS\_FROM\_SQLSET  
    + 从Stored Outlines 中导入   
       DBMS\_SPM.MIGRATE\_STORED\_OUTLINE  
    + 从内存中存在的计划中导入  
       DBMS\_SPM.LOAD\_PLANS\_FROM\_CURSOR\_CACHE;  
    + 通过staging table从另外一个系统中移植  
       DBMS\_SPM.CREATE\_STGTAB\_BASELINE  
       DBMS\_SPM.PACK\_STGTAB\_BASELINE  
       DBMS\_SPM.UNPACK\_STGTAB\_BASELINE

**执行计划的选择过程**

在OPTIMIZER\_USE\_SQL\_PLAN\_BASELINES被设置成默认值TRUE，SQl Plan Baseline就会起作用。

1. 首先，无论是否存在baseline，oracle都会正常进行硬解析或者软解析，为SQL生成一个执行计划。 由于ACS和bind peeking的作用，存在baseline的SQL有可能在这时生成一个不同于baseline的执行计划。  
2. 如果baseline不存在，就按生成的计划执行。如果baseline存在，那么要查看history里是否有这个计划，如果没有，就将这个计划插入，并标记为ENABLED,NON-ACCEPTED.   
 3. 在baseline中查看是否有FIXED的计划存在，如果存在，执行FIXED的计划，如果存在多个FIXED的计划，根据统计信息重新计算cost，选择cost小的那个。  
 4. 如果FIXED的计划不存在，就选择ACCEPTED的计划执行。 如果存在多个ACCEPTED的计划，根据统计信息重新计算cost，选择cost小的那个。

*\* 注意这里每次重新计算cost的代价不大，因为执行计划是已知的，优化器不必遍历所有的可能，只需根据算法计算出已知计划的cost便可*

**执行计划的演化(evolution)**

执行计划的演化指Plan History里的执行计划从NON-ACCEPTED，变成ACCEPTED的过程。 如果上所述，由于ACS和Bind Peeking的作用，存在baseline的SQL有可能生成新的执行计划，被保存到Plan History中。 Oracle提供了API，通过自动或手工的方式，将一个计划标记为ACCEPTED，这个计划就会被后续的执行所选择。

使用DBMS\_SPM.EVOLVE\_SQL\_PLAN\_BASELINE这个API来控制执行计划的演化。语法：  
 DBMS\_SPM.EVOLVE\_SQL\_PLAN\_BASELINE (  
   sql\_handle IN VARCHAR2 := NULL, --> NULL 表示针对所有SQL  
  plan\_name  IN VARCHAR2 := NULL,  
   time\_limit IN INTEGER  := DBMS\_SPM.AUTO\_LIMIT,  
   verify     IN VARCHAR2 := 'YES',  
   commit     IN VARCHAR2 := 'YES' )  
 RETURN CLOB;

这里由两个标记控制：  
 o Verify   
   + YES (只有性能更好的计划才会被演化)  
   + NO (演化所有的计划)  
 o Commit  
   + YES (直接演化)  
   + NO (只生成报告)

这里可以通过不同的排列组合，达到不同的效果：  
 o 自动接收所有性能更好的执行计划 (Verify->YES, Commit->YES)  
 o 自动接收所有新的执行计划 (Verify->NO, Commit->YES)  
 o 比较性能，生成报告，人工确认是否演化 (Verify->NO, Commit->NO)

*\* 对于性能的验证的方式，oracle会去实际执行来比较buffer gets*

**修改已有的Baseline**

通过DBMS\_SPM.ALTER\_SQL\_PLAN\_BASELINE来完成。

DBMS\_SPM.ALTER\_SQL\_PLAN\_BASELINE (  
   sql\_handle      IN VARCHAR2 := NULL,  
   plan\_name       IN VARCHAR2 := NULL,  
   attribute\_name  IN VARCHAR2,  
   attribute\_value IN VARCHAR2 )  
 RETURN PLS\_INTEGER;

比如，把某个baseline 标记为FIXED，更多属性请参见[官方文档](http://docs.oracle.com/cd/E11882_01/appdev.112/e25788/d_spm.htm" \l "ARPLS68156)。

SET SERVEROUT ON;  
 DECLARE  
   x NUMBER;  
 BEGIN  
   x := DBMS\_SPM.ALTER\_SQL\_PLAN\_BASELINE (  
     sql\_handle      => '&&sql\_handle',  
     plan\_name       => '&&plan\_name',  
     attribute\_name  => 'FIXED',  
     attribute\_value => 'YES' );  
 END;  
 /

**常见应用**

o 我们常见的一个场景是，一条SQL在使用hint时会生成一个好的计划，我们需要以此在原SQL上创建一个baseline。 具体方法请参加note 787692.1

**注意**

o 当您使用多种方式控制执行计划时：  
   + Stored Outline存在时，它具有最高的优先级。  
   + 已经实施的SQL profile会被自动加入到SQL plan baseline中  
   + STA(SQL Tuning Advisor) 会自动接收新的profile，意味着它会生成新的baseline  
 o 如果可能话，尽量移植到SPM，混合多种方式会变得复杂

**相关参数**

optimizer\_capture\_sql\_plan\_baselines  
 optimizer\_use\_sql\_plan\_baselines  
 create\_stored\_outline  
 use\_stored\_outlines

**参考文档**

[White paper: SQL Plan Management in Oracle Database 11g](http://www.oracle.com/technetwork/database/focus-areas/bi-datawarehousing/twp-sql-plan-management-11gr2-133099.pdf)

How to Use SQL Plan Management (SPM) - Example Usage (Doc ID 456518.1)  
 Plan Stability Features (Including SPM) Start Point (Doc ID 1359841.1)  
 HOW TO LOAD SQL PLANS INTO SPM FROM AWR (Doc ID 789888.1)  
 Sql Plan Baseline Not always created (Doc ID 788853.1)

Transporting SQL PLAN Baselines from one database to another. (Doc ID 880485.1)