SQL PLAN BASELINE 使用心得

BY SHOUG. 崔宏慧

How to Find SHOUG?

sql plan baseline 使用心得

测试内容：

1、 dba_sql_plan_baselines 表中和时间有关的四个字段 CREATED， LAST_MODIFIED，LAST_EXECUTED，LAST_VERIFIED 的变化规律
2、 候选 sql plan 变为 accepted sql plan baseline 的几种方法
3、 SQL 语句对应的 sql plan baseline 均失效的情况下 Optimizer 将新生成的执行计划演进为 sql plan baseline 的过程
4、 不同用户针对各自用户下的表，执行同一条 sql 语句， sql plan baseline 的共享机制

建立测试用表：

```
grant connect,resource,unlimited tablespace to scott identified by sdfg_1234;

create table scott.t1 tablespace ts_pub as select * from dba_objects;

create table scott.t2 tablespace ts_pub as select * from dba_objects where rownum<100;

exec dbms_stats.gather_table_stats(ownname=>'scott',tabname=>'t1',method_opt=>'for all columns size 1',cascade=>TRUE,no_invalidate=>FALSE);

exec dbms_stats.gather_table_stats(ownname=>'scott',tabname=>'t2',method_opt=>'for all columns size 1',cascade=>TRUE,no_invalidate=>FALSE);
```

1、 dba_sql_plan_baselines 表中和时间有关字段的变化规律，涉及到以下 4 个字段

CREATED
LAST_MODIFIED
LAST_EXECUTED
LAST_VERIFIED

```
###开启 session 级的 sql capture，自动生成首条 sql plan baseline
--session 1，设置 Session 级的 capture
SQL> select * from dba_sql_plan_baselines;

no rows selected

alter system optimizer_capture_sql_plan_baselines=TRUE;

select count(*) from scott.t1 where object_id in (select object_id from scott.t2);

--session 2，dba_sql_plan_baselines 中没有记录，因为上述 sql 只执行了一次
select sql_handle,sql_text,plan_name,creator,last_modified,last_executed,last_verified from dba_sql_plan_baselines;

--session 1，再次执行一遍 sql
select count(*) from scott.t1 where object_id in (select object_id from scott.t2);

--session 2，dba_sql_plan_baselines 产生了首条 sql plan baseline，首条初始状态就是 accepted
select
```

```
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_v
erified                                                                                    from
dba_sql_plan_baselines;
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ------------------------------------------------- ------------------------------ --- --- ----- ----------------------------- --------
---------------------
LAST_EXECUTED                  LAST_VERIFIED
------------------------------ -----------------------------
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   02-JUL-14 02.37.20.000000 PM   02-JUL-1
4 02.37.20.000000 PM
                     select object_id from scott.t2
02-JUL-14 02.37.20.000000 PM
```

###上述结果中的时间点字段值，last_verified 值为空，因为其是这条 sql 生成的首条 baseline 所以没有经过验证；因为是新建的 sql plan baseline 其余三个时间字段值都一样
CREATED：02-JUL-14 02.37.20.000000 PM
LAST_MODIFIED：02-JUL-14 02.37.20.000000 PM
LAST_EXECUTED：02-JUL-14 02.37.20.000000 PM
LAST_VERIFIED：NULL

###上述结果中的时间点字段值，last_verified 值为空，因为其是这条 sql 生成的首条 baseline 所以没有经过
--session 1，第三次执行 sql，执行前关闭 sql capture 参数

```
alter session set optimizer_capture_sql_plan_baselines=FALSE;

select count(*) from scott.t1 where object_id in (select object_id from scott.t2);
```

--session 2，观察时间字段状态，CREATED、LAST MODIFIED 两个字段值没有变化，这个可以理解，LAST_EXECUTED 值应该变化为最近一次的执行时间，但事实却没有变化，即使 alter system flush shared_pool 以后重新执行语句，也没有变化

```
select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_v
erified from dba_sql_plan_baselines;
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ------------------------------------------------- ------------------------------ --- --- ----- ----------------------------- --------
---------------------
LAST_EXECUTED                  LAST_VERIFIED
------------------------------ -----------------------------
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   02-JUL-14 02.37.20.000000 PM   02-JUL-1
4 02.37.20.000000 PM
                     select object_id from scott.t2
02-JUL-14 02.37.20.000000 PM
```

###通过 DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE 查看 sql_plan_baseline 对应的执行计划为 FTS

```
select                                                  *                                                  from
table(dbms_xplan.display_sql_plan_baseline(sql_handle=>'SQL_d11d993788ae4828',plan_name=>'
SQL_PLAN_d27ct6y4awk1822a9c5af'));
```

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

--------------------------------------------------------------------------------
SQL handle: SQL_d11d993788ae4828
SQL text: select count(*) from scott.t1 where object_id in (select object_id from
          scott.t2)
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Plan name: SQL_PLAN_d27ct6y4awk1822a9c5af         Plan id: 581551535
Enabled: YES     Fixed: NO     Accepted: YES     Origin: AUTO-CAPTURE
--------------------------------------------------------------------------------
```

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

Plan hash value: 1240933221

-------------------------------------------------------------------------------
| Id  | Operation                    | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |      |     1 |     9 |   462   (2)| 00:00:06 |
|   1 |  SORT AGGREGATE              |      |     1 |     9 |            |          |
|*  2 |   HASH JOIN RIGHT SEMI       |      |     3 |    27 |   462   (2)| 00:00:06 |
|   3 |    TABLE ACCESS FULL         | T2   |    99 |   297 |     5   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL         | T1   |   177K|  1042K|   455   (1)| 00:00:06 |

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

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

   2 - access("OBJECT_ID"="OBJECT_ID")

28 rows selected.
```

###t1 表的 object_id 字段上创建索引，再次执行 sql

```
create index scott.ind_objid_t1 on scott.t1(object_id) tablespace ts_pub;

exec dbms_stats.gather_table_stats(ownname=>'scott',tabname=>'t1',method_opt=>'for all columns size 1',cascade=>TRUE,no_invalidate=>FALSE);

exec dbms_stats.gather_table_stats(ownname=>'scott',tabname=>'t2',method_opt=>'for all columns size 1',cascade=>TRUE,no_invalidate=>FALSE);
```

###dba_sql_plan_baselines 里又生成了一条 plan_name= SQL_PLAN_d27ct6y4awk18b1b38b11(sql_handle 与前一条相同的 sql)，但没有被 accepted 的 baseline，这条记录的 CREATED、LAST_MODIFIED 字段表明了该条 baseline 的创建时间，LAST_EXECUTED、LAST_VERIFIED 均为空值

```
col sql_handle format a20
col creator format a5
col sql_text format a50
col created format a30
col last_modified format a30
col last_executed format a30
col last_verified format a30
set linesize 190
set pagesize 200
```

```
select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_exe
cuted,last_verified from dba_sql_plan_baselines;
```

```
SQL_HANDLE           SQL_TEXT                                        PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ----------------------------------------------- ------------------------------ --- --- ----- ----------------------------- --------
---------------------
LAST_EXECUTED                  LAST_VERIFIED
------------------------------ ------------------------------
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   02-JUL-14 02.37.20.000000 PM   02-JUL-1
4 02.37.20.000000 PM
                     select object_id from scott.t2
02-JUL-14 02.37.20.000000 PM

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES NO  SYS   02-JUL-14 03.22.41.000000 PM   02-JUL-1
4 03.22.41.000000 PM
                     select object_id from scott.t2
```

###执行 sql，虽然有索引，但因为 baseline 的存在，走的依然是 FTS

```
set autotrace traceonly;

SQL> select count(*) from scott.t1 where object_id in (select object_id from scott.t2);
```

Execution Plan

Plan hash value: 1240933221

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |      |     1 |     9 |   462   (2)| 00:00:06 |
|   1 |  SORT AGGREGATE              |      |     1 |     9 |            |          |
|*  2 |   HASH JOIN RIGHT SEMI       |      |     3 |    27 |   462   (2)| 00:00:06 |
|   3 |    TABLE ACCESS FULL         | T2   |    99 |   297 |     5   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL         | T1   |   177K|  1042K|   455   (1)| 00:00:06 |
-------------------------------------------------------------------------------
```

Predicate Information (identified by operation id):

```
   2 - access("OBJECT_ID"="OBJECT_ID")
```

Note

- SQL plan baseline "SQL_PLAN_d27ct6y4awk1822a9c5af" used for this statement

Statistics

```
          0  recursive calls
          0  db block gets
       2557  consistent gets
       2556  physical reads
          0  redo size
        526  bytes sent via SQL*Net to client
        519  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

###人工演进 sql plan baseline，根据 Buffer Get 优化前后的对比 2557/11=232.45，得出使用索引的 sql plan baseline 所获得的性能是 FTS 的 232 倍，oracle 情况下根据隐含参数 _plan_verify_improvement_margin(默认值为 150，表示 1.5 倍)的值决定性能达到原先多少倍时 accept 新的 sql plan baseline，此例中已经达到了 232 被，所以当让是 verified and accepted

```
set serveroutput on
set long 10000
declare
result_clob clob;
begin
result_clob:=DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(sql_handle=>'SQL_d11d9
93788ae4828',plan_name=>'SQL_PLAN_d27ct6y4awk18b1b38b11',verify=>'YES',comm
it=>'YES');
dbms_output.put_line(result_clob);
end;
/
```

```
--------------------------------------------------------------------------------,

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

Inputs:
-------
   SQL_HANDLE = SQL_d11d993788ae4828
   PLAN_NAME    =
SQL_PLAN_d27ct6y4awk18b1b38b11
   TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
   VERIFY       =
YES
   COMMIT       = YES

Plan:
SQL_PLAN_d27ct6y4awk18b1b38b11
-----------------------------------
   Plan was
verified: Time used .901 seconds.
   Plan passed performance criterion: 232.77
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):          59.641          .298          200.14


CPU Time(ms):                 34.444           0
   Buffer Gets:
2557              11          232.45
   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.

###查看 PLAN_NAME=SQL_PLAN_d27ct6y4awk18b1b38b11 对应 sql plan baseline，LAST_VERIFIED 和
LAST_MODIFIED 为同一个时间，LAST_VERIFIED 表示在这个时间完成了 Verify 动作，LAST_MODIFIED 表示在
Verify 通过后将此 baseline 从 not accepted 变为 accepted 的时间。

```
SQL_HANDLE           SQL_TEXT                                      PLAN_NAME                      ENA ACC CREAT CREATED                      LAST_MOD
IFIED
-------------------- --------------------------------------------- ------------------------------ --- --- ----- ---------------------------- --------
---------------------
LAST_EXECUTED                  LAST_VERIFIED
------------------------------ ----------------------------
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   02-JUL-14 02.37.20.000000 PM   02-JUL-1
4 02.37.20.000000 PM
                     select object_id from scott.t2
02-JUL-14 02.37.20.000000 PM

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES YES SYS   02-JUL-14 03.22.41.000000 PM   02-JUL-1
4 03.44.10.000000 PM
                     select object_id from scott.t2
                               02-JUL-14 03.44.10.000000 PM
```

CREATED: 02-JUL-14 03.22.41.000000 PM
LAST_MODIFIED: 02-JUL-14 03.44.10.000000 PM
LAST_VERIFIED:02-JUL-14 03.44.10.000000 PM

###执行该 SQL 后发现 last_executed 时间已经是最新的时间了
SQL> select count(*) from scott.t1 where object_id in (select object_id from scott.t2);

select sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_verified from dba_sql_plan_baselines;

```
SQL> select sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_verified from dba_sql_plan_baselines;

SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ------------------------------------------------- ------------------------------ --- --- ----- ------------------------------ --------
----------------------
LAST_EXECUTED                 LAST_VERIFIED
----------------------------- -----------------------------
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   02-JUL-14 02.37.20.000000 PM   02-JUL-1
4 02.37.20.000000 PM
                     select object_id from scott.t2
02-JUL-14 02.37.20.000000 PM

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES YES SYS   02-JUL-14 03.22.41.000000 PM   02-JUL-1
4 03.44.10.000000 PM
                     select object_id from scott.t2
02-JUL-14 04.25.33.000000 PM   02-JUL-14 03.44.10.000000 PM
```

LAST_EXECUTED:02-JUL-14 04.25.33.000000 PM

用 dbms_xplan.display_sql_plan_baseline 显示 Plan_name=SQL_PLAN_d27ct6y4awk18b1b38b11 的执行计划，这次采用的是 Nest Loop

SQL> select * from table(dbms_xplan.display_sql_plan_baseline(sql_handle=>'SQL_d11d993788ae4828',plan_name=>'SQL_PLAN_d27ct6y4awk18b1b38b11'));

PLAN_TABLE_OUTPUT

SQL handle: SQL_d11d993788ae4828
SQL text: select count(*) from scott.t1 where object_id in (select object_id fro
m

scott.t2)

Plan name: SQL_PLAN_d27ct6y4awk18b1b38b11 Plan id: 2981333777

PLAN_TABLE_OUTPUT

Enabled: YES Fixed: NO Accepted: YES Origin: AUTO-CAPTURE

Plan hash value: 2406492491

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time
|

PLAN_TABLE_OUTPUT

| 0 | SELECT STATEMENT | | 1 | 9 | 56 (2)| 00:00

```
:01 |

|     1 |   SORT AGGREGATE          |                  |       1 |       9 |                |
      |

|     2 |    NESTED LOOPS           |                  |      99 |     891 |      56     (2) | 00:00
:01 |

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

|     3 |      SORT UNIQUE          |                  |      99 |     297 |       5     (0) | 00:00
:01 |

|     4 |       TABLE ACCESS FULL | T2               |      99 |     297 |       5     (0) | 00:00
:01 |

|*    5 |      INDEX RANGE SCAN     | IND_OBJID_T1 |       1 |       6 |       1     (0) | 00:00
:01 |

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

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


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

   5 - access("OBJECT_ID"="OBJECT_ID")
```

阶段总结：

CREATED：sql plan 生成到 plan_history 的时间(可以是 accept 或者 not accept 状态)

LAST_MODIFIED：sql plan 上一次修改的时间，这个修改时间反映了 sql plan 演进过程中将 not accetped 的 sql plan 更新为 accepted 动作发生的时间，也能反映使用 alter_sql_plan_baseline 对于 sql plan 任何属性更改的时间

LAST_VERIFIED：sql plan 最后一次被验证的时间，同一个 plan 被验证一遍之后如果再重复进行验证，时间还是停留在首次验证的时间；第一条 sql plan 自动成为 sql plan baseline 时其 last_verified 时间为空，说明其没有经过 verify，即使后续对首条 sql plan 人工进行演进，其 last_verified 时间依然为空

LAST_EXECUTED：名义上为最后一次执行的时间，实际测下来定格在首次执行的时间，后续的执行并不会更新

2、 使 sql plan 变为 accepted sql plan baseline 的几种方法

(1) 调用 Dbms_spm.evolve_sql_plan_baseline 函数，需要人工调用(在 12c 版本里已经引入 sql plan evolve advisor 能实现自动演进 sql plan baseline)，这个是最常用的方法，只做如下说明：
其中 Verify=yes 表示经过 optimizer 验证
verify=no 表示不经过 optimizer 验证强制变为 accepted 状态

(2) 调用 Dbms_spm.LOAD_PLANS_FROM_CURSOR_CACHE 或者 LOAD_PLANS_FROM_SQLSET 函数，这里使用 LOAD_PLANS_FROM_CURSOR_CACHE 函数将 shared pool 中已经存在的执行计划 load 到 baseline，且状态变为 accepted；

```
###执行 sql，使其 cache 到 shared pool
variable v_objid number;
exec :v_objid:=1000;
select count(*) from scott.t1 where object_id<:v_objid;
 SQL> select sql_text,sql_id,child_number,plan_hash_value from v$sql where
        sql_text like 'select count(*) from scott.t1%';

SQL_TEXT
SQL_ID             CHILD_NUMBER PLAN_HASH_VALUE
-------------------------------------------------------------------------------- ------------- -------
----- --------------
select count(*) from scott.t1 where object_id<:v_objid
9hup7n51za19u                 0          4020739011

###显示执行计划
select * from
table(dbms_xplan.display_cursor(sql_id=>'9hup7n51za19u',cursor_child_no=>0,for
mat=>'ALL'));
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
SQL_ID  9hup7n51za19u, child number 0
-------------------------------------
select count(*) from scott.t1 where object_id<:v_objid

Plan hash value: 4020739011

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

| Id  | Operation              | Name             | Rows  | Bytes | Cost (%CPU)|
Time
  |

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

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

```
|     0 | SELECT STATEMENT    |                      |           |           |         5 (100)|
  |

|     1 |    SORT AGGREGATE     |                      |        1 |         6 |
|
  |

|*    2 |     INDEX RANGE SCAN| IND_OBJID_T1 |    8893 | 53358 |         5
(0)| 00:00:01

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
  |

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


Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

    1 - SEL$1
    2 - SEL$1 / T1@SEL$1

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

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

    2 - access("OBJECT_ID"<:V_OBJID)

Column Projection Information (identified by operation id):
-----------------------------------------------------------

    1 - (#keys=0) COUNT(*)[22]

###从 shared pool 中将上述 sql 的执行计划 load 到 sql plan baseline，load 进来之
后就变成了 Accepted，没有 verify 的过程
set serveroutput on
declare
result_int pls_integer;
begin
result_int:=DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE(sql_id=>'9hup7
n51za19u',plan_hash_value=>4020739011,fixed=>'NO',enabled=>'YES');
```

```
dbms_output.put_line(result_int);
end;
/
```

###在 dba_sql_plan_baselines 中找到了该条 sql plan baseline，已经被 accepted

```
select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_
executed,last_verified from dba_sql_plan_baselines where sql_text like '%v_objid';
```

```
SQL_HANDLE           SQL_TEXT                                           PLAN_NAME                      ENA ACC CREAT CREATED                        LAST_MOD
IFIED
-------------------- -------------------------------------------------- ------------------------------ --- --- ----- ------------------------------ --------
--------------------
LAST_EXECUTED                  LAST_VERIFIED
------------------------------ ------------------------------
SQL_f9a292a38d383e14 select count(*) from scott.t1 where object_id<:v_o SQL_PLAN_gm8nknf6mhghn28a6f5d9 YES YES SYS   03-JUL-14 10.49.51.000000 AM   03-JUL-1
4 10.49.51.000000 AM
                     bjid
```

###再次执行 sql 时已经能用到了这条 sql plan baseline 了

```
variable v_objid number;
exec :v_objid:=500;

select count(*) from scott.t1 where object_id<:v_objid;
set autotrace traceonly;
select count(*) from scott.t1 where object_id<:v_objid;
```

Execution Plan

```
----------------------------------------------------------
Plan hash value: 4020739011

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

| Id  | Operation             | Name            | Rows  | Bytes | Cost (%CPU)|
Time
  |

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

|   0 | SELECT STATEMENT      |                 |     1 |     6 |     5   (0)|
00:00:01
  |

|   1 |  SORT AGGREGATE       |                 |     1 |     6 |
|
  |

|*  2 |   INDEX RANGE SCAN| IND_OBJID_T1 |  8893 | 53358 |     5
(0)| 00:00:01
  |
```

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


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

   2 - access("OBJECT_ID"<TO_NUMBER(:V_OBJID))

Note
-----
   - SQL plan baseline "SQL_PLAN_gm8nknf6mhghn28a6f5d9" used for this
statement


Statistics
----------------------------------------------------------
         27  recursive calls
         16  db block gets
         15  consistent gets
         13  physical reads
       3136  redo size
        527  bytes sent via SQL*Net to client
        520  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

(3) 通过 dbms_sqltune 对 SQL 语句进行调优，并接受其调优建议
这里沿用本文第一部分对于 select count(*) from scott.t1 where object_id in (select object_id from scott.t2)语句生成的两条 sql plan baseline，作如下处理：删除走索引的那条 plan（只保留 FTS）->使用 sql tuning advisor 对语句进行调优->接受 advisor 使用索引访问的建议

```
###人工删除掉走索引的 sql plan
set numformat 99999999999999999999999999
col sql_handle format a20
col creator format a5
col sql_text format a50
col created             format a30
col last_modified    format a30
col last_executed    format a30
col last_verified    format a30
set linesize 180

select
```

```
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,la
st_verified from dba_sql_plan_baselines where sql_handle='SQL_d11d993788ae4828';
```

```
SQL_HANDLE           SQL_TEXT                                        PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MO
IFIED
-------------------- ----------------------------------------------- ------------------------------ --- --- ----- ----------------------------- -------
--------------------
LAST_EXECUTED                  LAST_VERIFIED
------------------------------ -----------------------------
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   02-JUL-14 02.37.20.000000 PM   02-JUL-
4 02.37.20.000000 PM
                     select object_id from scott.t2
02-JUL-14 02.37.20.000000 PM

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES YES SYS   02-JUL-14 03.22.41.000000 PM   03-JUL-
4 09.35.17.000000 AM
                     select object_id from scott.t2
02-JUL-14 04.25.33.000000 PM   02-JUL-14 03.44.10.000000 PM
```

```
--删除其中使用索引的那条
set serveroutput on
declare
        result_int pls_integer;
begin
        result_int:=dbms_spm.drop_sql_plan_baseline(sql_handle=>'SQL_d11d993788ae4828',pla
n_name=>'SQL_PLAN_d27ct6y4awk18b1b38b11');
        dbms_output.put_line(result_int);
end;
/
--删除成功只剩一条 FTS 的 plan
select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,las
t_verified from dba_sql_plan_baselines where sql_handle='SQL_d11d993788ae4828';
```

```
SQL_HANDLE           SQL_TEXT                                        PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ----------------------------------------------- ------------------------------ --- --- ----- ----------------------------- --------
---------------------
LAST_EXECUTED                  LAST_VERIFIED
------------------------------ -----------------------------
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   02-JUL-14 02.37.20.000000 PM   02-JUL-1
4 02.37.20.000000 PM
                     select object_id from scott.t2
02-JUL-14 02.37.20.000000 PM
```

```
###执行 dbms_sqltune，生成并接受优化建议
--生成 tuning 任务
declare
my_task_name varchar2(30);
my_sqltext clob;
begin
my_sqltext:='select count(*) from scott.t1 where object_id in (select object_id from
scott.t2)';
my_task_name:=dbms_sqltune.create_tuning_task(sql_text=>my_sqltext,user_name=>'SCOTT',
scope=>'COMPREHENSIVE',time_limit=>60,task_name=>'scott_sql_tune_1',description=>'tune
1');
end;
/

--执行 tuning 任务
begin
dbms_sqltune.execute_tuning_task(task_name=>'scott_sql_tune_1');
end;
/

###查看 sqltune 报告，截取了相关内容
```

```
set long 9000
set longchunksize 1000
set linesize 800
 select dbms_sqltune.report_tuning_task('scott_sql_tune_1') from dual;
```

1- Original With Adjusted Cost

```
---------------------------
Plan hash value: 1240933221


DBMS_SQLTUNE.REPORT_TUNING_TASK('SCOTT_SQL_TUNE_1')
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-----------------------------------------------
-------------------------------------------------------------------------
| Id  | Operation                    | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |      |     1 |     9 |   462   (2)| 00:00:06 |
|   1 |  SORT AGGREGATE              |      |     1 |     9 |            |          |
|
|*  2 |   HASH JOIN RIGHT SEMI       |      |     3 |    27 |   462   (2)| 00:00:06 |
|   3 |    TABLE ACCESS FULL         | T2   |    99 |   297 |     5   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL         | T1   |  177K |  1042K|   455   (1)| 00:00:06 |
-------------------------------------------------------------------------

Predicate Information (identified by operation id):

DBMS_SQLTUNE.REPORT_TUNING_TASK('SCOTT_SQL_TUNE_1')
```

2- Using SQL Profile

```
-------------------
Plan hash value: 2406492491

-------------------------------------------------------------------------------
| Id  | Operation                    | Name             | Rows  | Bytes | Cost (%CPU)| Time
|
-------------------------------------------------------------------------------

DBMS_SQLTUNE.REPORT_TUNING_TASK('SCOTT_SQL_TUNE_1')
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
----------------------------------------
|   0 | SELECT STATEMENT             |                  |     1 |     9 |    56   (2)|
00:00:01 |
|   1 |  SORT AGGREGATE              |                  |     1 |     9 |            |
```

```
|
|     2 |    NESTED LOOPS           |                |     99 |    891 |     56    (2)|
00:00:01 |
|     3 |     SORT UNIQUE           |                |     99 |    297 |      5    (0)|
00:00:01 |
|     4 |      TABLE ACCESS FULL    | T2             |     99 |    297 |      5    (0)|
00:00:01 |
|*    5 |     INDEX RANGE SCAN      | IND_OBJID_T1   |      1 |      6 |      1    (0)|
00:00:01 |
-------------------------------------------------------------------------------------

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

###接受 Advisor 推荐走索引的 Profile，同时可以看到 dba_sql_plan_baseline 里又增加了一条 accepted=yes 的 plan，这条正是我们刚才删除的，表明接受 dbms_sqltune 的调优结果也可以实现 sql plan baseline 的演进

```
execute
dbms_sqltune.accept_sql_profile(task_name=>'scott_sql_tune_1',task_owner=>'SCOTT',replac
e=>TRUE);

select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,la
st_verified from dba_sql_plan_baselines where sql_handle='SQL_d11d993788ae4828';
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ------------------------------------------------- ------------------------------ --- --- ----- ----------------------------- --------
--------------------
LAST_EXECUTED                  LAST_VERIFIED
------------------------------ -----------------------------
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   02-JUL-14 02.37.20.000000 PM   02-JUL-1
4 02.37.20.000000 PM
                     select object_id from scott.t2
02-JUL-14 02.37.20.000000 PM

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES YES SYS   03-JUL-14 02.26.01.000000 PM   03-JUL-1
4 02.26.01.000000 PM
                     select object_id from scott.t2
```

###验证已经新的 sql plan baseline 已经被使用

```
SQL> set autotrace traceonly explain
SQL>select count(*) from scott.t1 where object_id in (select object_id from scott.t2);

Execution Plan
----------------------------------------------------------
Plan hash value: 2406492491

-------------------------------------------------------------------------------------
| Id  | Operation                 | Name           | Rows  | Bytes | Cost (%CPU)| Time
|
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |                |     1 |     9 |    56   (2)| 00:00:01
|
|   1 |  SORT AGGREGATE           |                |     1 |     9 |            |
|
|   2 |   NESTED LOOPS            |                |    99 |   891 |    56   (2)| 00:00:01
|
|   3 |    SORT UNIQUE            |                |    99 |   297 |     5   (0)| 00:00:01
|
|   4 |     TABLE ACCESS FULL     | T2             |    99 |   297 |     5   (0)| 00:00:01
```

```
|
|*  5 |      INDEX RANGE SCAN    | IND_OBJID_T1 |       1 |       6 |       1    (0)|
00:00:01 |
---------------------------------------------------------------------------------

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

   5 - access("OBJECT_ID"="OBJECT_ID")

Note
-----
   - SQL profile "SYS_SQLPROF_0146fae6b2110000" used for this statement
   - SQL plan baseline "SQL_PLAN_d27ct6y4awk18b1b38b11" used for this statement
```

阶段总结：

方法(1)适用于已经存在于 sql plan history 里但还未被 accepted 的 sql plan，可以通过 optimizer 验证(verify=yes)后实现演进，或者不通过验证(verify=no)而直接演进为 sql plan baseline

方法(2)在不开启 session 级或 system 级自动捕捉 (optimizer_capture_sql_plan_baselines=FALSE)的情况下，人工将已经生成的执行计划装载为 sql plan baseline，即绕过 optimizer 的评估，直接演进为 accepted plan 的情况。这种方法需要人工确认该执行计划是一定是最优的，否则会导致后续按照该 baseline 执行的 SQL 产生性能问题

方法(3)语句出现性能问题后，求助 sql tuning advisor 得到并应用优化建议，生成 accepted 的 sql plan baseline，属于事后调优的范畴

3、SQL 语句对应的 sql plan baseline 均失效的情况下，sql plan 演进会跳过 verify 步骤，直接变为 accepted

```
###Drop 掉原有的 sql plan baseline
declare
result_int pls_integer;
begin
result_int:=dbms_spm.drop_sql_plan_baseline(sql_handle=>'SQL_d11d993788ae4828');
end;
/

###重新构建测试环境
create table scott.t1 tablespace ts_pub as select * from dba_objects;
create table scott.t2 tablespace ts_pub as select * from dba_objects where rownum<100;
create index scott.ind_objid_t1 on scott.t1(object_id) tablespace ts_pub;
exec dbms_stats.gather_table_stats(ownname=>'scott',tabname=>'t1',method_opt=>'for all columns
size 1',cascade=>TRUE,no_invalidate=>FALSE);
exec dbms_stats.gather_table_stats(ownname=>'scott',tabname=>'t2',method_opt=>'for all columns
size 1',cascade=>TRUE,no_invalidate=>FALSE);

alter session set optimizer_capture_sql_plan_baselines=TRUE;
```

select count(*) from scott.t1 where object_id in (select object_id from scott.t2); --执行至少两次

alter session set optimizer_capture_sql_plan_baselines=FALSE;

###drop 掉索引，再次执行 sql，观察到 dba_sql_plan_baselines 里，索引对应的 plan REPRODUCED 变成了 NO，受索引被 drop 的影响此条 plan baseline 失效了；同时新增了一条 FTS 的 plan，但状态为 not accepted

drop index scott.ind_objid_t1;

select count(*) from scott.t1 where object_id in (select object_id from scott.t2);

select sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_verified,REPRODUCED from dba_sql_plan_baselines where sql_handle='SQL_d11d993788ae4828';

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ------------------------------------------------- ------------------------------ --- --- ----- ------------------------------ --------
--------------------
LAST_EXECUTED                  LAST_VERIFIED                  REP
------------------------------ ------------------------------ ---
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES NO  SYS   03-JUL-14 04.08.10.000000 PM   03-JUL-1
4 04.08.10.000000 PM
                     select object_id from scott.t2
                                                                   YES

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES YES SYS   03-JUL-14 02.26.01.000000 PM   03-JUL-1
4 04.08.10.000000 PM
                     select object_id from scott.t2
                                                                   NO
```

###现在把 FTS 的 plan 演进为 Accepted sql plan baseline，从 EVOLVE_SQL_PLAN_BASELINE 函数的输出可以看出，虽然指定了 verify=YES，但因走索引的 plan 已经失效，oracle 并没有进行 verify 就直接 accept 此 plan 了。

```
set serveroutput on
set long 10000
declare
result_clob clob;
begin
result_clob:=DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(sql_handle=>'SQL_d11d993788ae4828
',plan_name=>'SQL_PLAN_d27ct6y4awk1822a9c5af',verify=>'YES',commit=>'YES');
dbms_output.put_line(result_clob);
end;
/
```

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

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

Inputs:
-------
   SQL_HANDLE = SQL_d11d993788ae4828
   PLAN_NAME  =
SQL_PLAN_d27ct6y4awk1822a9c5af
```

```
   TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
   VERIFY        =
YES
   COMMIT        = YES

Plan:
SQL_PLAN_d27ct6y4awk1822a9c5af
-----------------------------------
   Plan was
not verified.
   Using cost-based plan as could not reproduce any
   accepted and
enabled baseline plan.
   Plan was changed to an accepted
plan.

-------------------------------------------------------------------------
------
                                          Report
Summary
-------------------------------------------------------------------------
-------
Number of plans verified: 0
Number of plans accepted: 1
```

###演进的结果验证，FTS 对应的 sql plan baseline 已经变成 Accepted=yes 了

```
select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_v
erified,REPRODUCED from dba_sql_plan_baselines where sql_handle='SQL_d11d993788ae4828'
```

```
SQL_HANDLE           SQL_TEXT                                        PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ----------------------------------------------- ------------------------------ --- --- ----- ----------------------------- --------
---------------------
LAST_EXECUTED                LAST_VERIFIED                REP
---------------------------- ---------------------------- ---
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   03-JUL-14 04.08.10.000000 PM   03-JUL-1
4 04.44.36.000000 PM
                     select object_id from scott.t2
                                                                                                    YES

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES YES SYS   03-JUL-14 02.26.01.000000 PM   03-JUL-1
4 04.08.10.000000 PM
                     select object_id from scott.t2
                                                                                                    NO
```

###对于走索引的这条 sql plan baseline，若要使其重新生效，即 reproduced 从 NO 变为 YES，必须重新建立索引并且执行一次 sql 才行

```
select count(*) from scott.t1 where object_id in (select object_id from scott.t2);

create index scott.ind_objid_t1 on scott.t1(object_id) tablespace ts_pub;

exec dbms_stats.gather_table_stats(ownname=>'scott',tabname=>'t1',method_opt=>'for all columns
size 1',cascade=>TRUE,no_invalidate=>FALSE);

exec dbms_stats.gather_table_stats(ownname=>'scott',tabname=>'t2',method_opt=>'for all columns
size 1',cascade=>TRUE,no_invalidate=>FALSE);
```

###仅通过 Verify 并不能使其重新生效，提示已经是 accepted sql plan baseline

```
set serveroutput on
set long 10000
declare
```

```
result_clob clob;
begin
result_clob:=DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(sql_handle=>'SQL_d11d993788ae4828'
,plan_name=>'SQL_PLAN_d27ct6y4awk18b1b38b11',verify=>'YES',commit=>'YES');
dbms_output.put_line(result_clob);
end;
/
```

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

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

Inputs:
-------
   SQL_HANDLE = SQL_d11d993788ae4828
   PLAN_NAME  =
SQL_PLAN_d27ct6y4awk18b1b38b11
   TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
   VERIFY     =
YES
   COMMIT     = YES

Plan:
SQL_PLAN_d27ct6y4awk18b1b38b11
------------------------------------
   It is
already an accepted
plan.

--------------------------------------------------------------------------
------
                                    Report
Summary
--------------------------------------------------------------------------
-------
There were no SQL plan baselines that required processing.
```

```
select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last
_executed,last_verified,REPRODUCED from dba_sql_plan_baselines where
sql_handle='SQL_d11d993788ae4828'
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MOD
IFIED
-------------------- ------------------------------------------------- ------------------------------ --- --- ----- ----------------------------- --------
--------------------
LAST_EXECUTED                LAST_VERIFIED                 REP
---------- ---------- ---------- ---------- ---
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   03-JUL-14 04.08.10.000000 PM   03-JUL-1
4 04.58.15.000000 PM
                     select object_id from scott.t2)
                                                                       YES

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES YES SYS   03-JUL-14 02.26.01.000000 PM   03-JUL-1
4 04.08.10.000000 PM
                     select object_id from scott.t2)
                                                                       NO
```

###只有重新执行 sql，reproduced 才会变为 YES，此外还可以观察到这两条有效的 sql plan baseline 的 last_verified 字段均为空，表明这两条 sql plan 入驻的时候都没有经过 verify，也间接说明了入驻的当时没有有效的 sql plan baseline 存在，是被直接"保送"进了 sql plan baseline

```
select count(*) from scott.t1 where object_id in (select object_id from scott.t2);

select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_verified,REPRODUCED from dba_sql_plan_baselines where sql_handle='SQL_d11d993788ae4828';
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREAT CREATED                       LAST_MODIFIED
-------------------- ------------------------------------------------- ------------------------------ --- --- ----- ----------------------------- --------------------
LAST_EXECUTED                  LAST_VERIFIED                  REP
------------------------------ ------------------------------ ---
SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk1822a9c5af YES YES SYS   03-JUL-14 04.08.10.000000 PM   03-JUL-14 04.58.15.000000 PM
                     select object_id from scott.t2)
                                                                                                      YES

SQL_d11d993788ae4828 select count(*) from scott.t1 where object_id in ( SQL_PLAN_d27ct6y4awk18b1b38b11 YES YES SYS   03-JUL-14 02.26.01.000000 PM   03-JUL-14 05.01.45.000000 PM
                     select object_id from scott.t2)
03-JUL-14 05.01.45.000000 PM                                       YES
```

4、不同用户针对各自用户下的表，执行同一条 sql 语句， sql plan baseline 的共享机制

测试场景描述：两个用户 scott1、scott2 下各有一张名为 t1 的表，scott1.t1(object_id)上建立名为 ind_objid_t 的 non-unique 索引，且在 scott1 用户下执行 select * from t1 where object_id<100000 生成首条 sql plan baseline；之后分别在以下几种场景下使用 Scott2 用户执行同样的语句：select * from t1 where object_id<100000，观察是否能用到 scott1 用户生成的首条 sql plan baseline，这几种场景包括：

(1) Scott2.t1(object_id)字段没有索引
(2) Scott2.t1(object_id)字段创建 non-unique 索引，索引名称和 Scott1 保持一致
(3) Scott2.t1(object_id)字段创建 non-unique 索引， 索引名称和 Scott1 保持一致，人工增大 clustering_factor，使 Optimizer 偏向选择 FTS
(4) Scott2.t1(object_id)字段创建 non-unique 索引，索引名称有别于 Scott1
(5) Scott2.t1(object_id)字段创建 unique 索引，索引名称和 Scott1 保持一致
(6) 重建 Scott2.t1 表，同时更改 scott2.t1 表结构，除了 object_id 字段外，其余字段均和 Scott2.t1 中的字段不相同

数据环境准备：

###生成 scott1 用户下的表

```
grant connect,resource,unlimited tablespace to scott1 identified by scott1_1234;
grant plustrace to scott1;
create table scott1.t1 tablespace ts_pub as select * from dba_objects;
create index scott1.ind_objid_t on scott1.t1(object_id) tablespace ts_pub;
exec
dbms_stats.gather_table_stats(ownname=>'scott1',tabname=>'t1',method_opt=>'for all columns size 1',cascade=>TRUE,no_invalidate=>FALSE);
```

###生成 scott2 用户下的表

```
grant connect,resource,unlimited tablespace to scott2 identified by scott2_5678;
grant plustrace to scott2;
create table scott2.t1 tablespace ts_pub as select * from dba_objects;
```

```
exec dbms_stats.gather_table_stats(ownname=>'scott2',tabname=>'t1',method_opt=>'for all columns size 1',cascade=>TRUE,no_invalidate=>FALSE);

##清理现有环境中的 sql plan baseline，保持 dba_sql_plan_baseline 为空
set serveroutput on
declare
result_int pls_integer;
cursor t_cur is select distinct sql_handle from dba_sql_plan_baselines;
begin
for v_cur in t_cur loop
result_int:=dbms_spm.drop_sql_plan_baseline(sql_handle=>v_cur.sql_handle);
dbms_output.put_line(result_int);
end loop;
end;
/

alter system flush shared_pool;

##scott1 用户生成首条 sql plan baseline，
sqlplus scott1/scott1_1234
alter session set optimizer_capture_sql_plan_baselines=true;
select * from t1 where object_id<100000;   --执行至少两遍
alter session set optimizer_capture_sql_plan_baselines=false;

select sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_verified,REPRODUCED from dba_sql_plan_baselines;
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREAT CREATED
-------------------- ------------------------------------------------- ------------------------------ --- --- ----- ------------------------------
LAST_MODIFIED                  LAST_EXECUTED                 LAST_VERIFIED                  REP
------------------------------ ----------------------------- ------------------------------ ---
SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000           SQL_PLAN_93szh6uub7b2453067583 YES YES SCOTT 04-JUL-14 06.29.12.000000 PM
                                                                                                      1
04-JUL-14 06.29.12.000000 PM   04-JUL-14 06.29.12.000000 PM                                  YES
```

```
select * from table(dbms_xplan.display_sql_plan_baseline(sql_handle=>'SQL_91e3f036b4b3ac44',plan_name=>'SQL_PLAN_93szh6uub7b2453067583'));   --对应的执行计划是 index range scan
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|     0 | SELECT STATEMENT                |                 |    3560 |      337K|
213     (0)|
      00:00:03 |

|     1 |    TABLE ACCESS BY INDEX ROWID| T1              |    3560 |      337K|
213     (0)|
      00:00:03 |

|*    2 |     INDEX RANGE SCAN            | IND_OBJID_T |    3560 |          |
10     (0)|
      00:00:01 |
```

场景(1): Scott2.t1(object_id)字段没有索引，Scott2 用户执行 select * from t1 where object_id<100000;

```
select * from t1 where object_id<100000;
```

###t1.object_id 字段没有索引，无法用上 Scott1 用户下的 baseline，但会把 Scott1 用户创建的 plan 变成 reproduced=NO 同时在 sql plan history 里生成了一条 FTS 的 plan，Creator 为 scott2，状态为 not accepted

```
select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_exe
cuted,last_verified,REPRODUCED from dba_sql_plan_baselines;
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREATOR CREATED
-------------------- ------------------------------------------------- ------------------------------ --- --- ------- ------------------------------
LAST_MODIFIED                  LAST_EXECUTED                 LAST_VERIFIED                  REP
------------------------------ ----------------------------- ------------------------------ ---
SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000           SQL_PLAN_93szh6uub7b2453067583 YES YES SCOTT1  04-JUL-14 06.29.12.000000 PM
04-JUL-14 06.47.37.000000 PM   04-JUL-14 06.29.12.000000 PM                                  NO

SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000           SQL_PLAN_93szh6uub7b24dbd90e8e YES NO  SCOTT2  04-JUL-14 06.47.37.000000 PM
04-JUL-14 06.47.37.000000 PM                                                                  YES
```

```
select * from
table(dbms_xplan.display_sql_plan_baseline(sql_handle=>'SQL_91e3f036b4b3ac44',pla
n_name=>'SQL_PLAN_93szh6uub7b24dbd90e8e')); --plan_name=
```

SQL_PLAN_93szh6uub7b24dbd90e8e 执行计划如下

```
--------------------------------------------------------------------------------
SQL handle: SQL_91e3f036b4b3ac44
SQL text: select * from t1 where object_id<100000
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Plan name: SQL_PLAN_93szh6uub7b24dbd90e8e         Plan id: 3688435342
Enabled: YES     Fixed: NO     Accepted: NO     Origin: AUTO-CAPTURE
--------------------------------------------------------------------------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 838529891

-----------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  3560 |   337K|   456   (1)| 00:00:06 |
|*  1 |  TABLE ACCESS FULL | T1   |  3560 |   337K|   456   (1)| 00:00:06 |
-----------------------------------------------------------------------
```

阶段结论：scott2 用户的 t1 表上没有索引，优化器为 sql 生成的执行计划无法与 scott1 用户创建的 sql plan baseline 匹配，所以只能采用 FTS 的访问路径添加到 sql plan history，同时将 scott1 用户 plan_name= SQL_PLAN_93szh6uub7b2453067583 置为 reproduced=NO。可见优化器在匹配 sql plan baseline 时依据的是 sql_handle，和这个 plan 的 creator 无关。

场景(2): Scott2.t1(object_id)字段创建 non-unique 索引，索引名称和 Scott1 保持一致
##接着场景(1)，在 scott2.t1(object_id)创建和 scott1 同名的索引
create index scott2.ind_objid_t on scott2.t1(object_id) tablespace ts_pub;
exec
dbms_stats.gather_table_stats(ownname=>'scott2',tabname=>'t1',method_opt=>'for all columns size 1',cascade=>TRUE,no_invalidate=>FALSE);

##scott2 执行 sql，看到 plan_name=SQL_PLAN_93szh6uub7b2453067583 重新变为 REPRODUCED=YES 了，而且通过 sql 语句的执行计划可以看到 plan_name= SQL_PLAN_93szh6uub7b2453067583 重新被使用上了
set autotrace traceonly
select * from t1 where object_id<100000;
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)|
Time |

| 0 | SELECT STATEMENT | | 3560 | 337K| 213 (0)|
00:00:03 |

| 1 | TABLE ACCESS BY INDEX ROWID| T1 | 3560 | 337K| 213 (0)|
00:00:03 |

|* 2 | INDEX RANGE SCAN | IND_OBJID_T | 3560 | | 10 (0)|
00:00:01 |

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

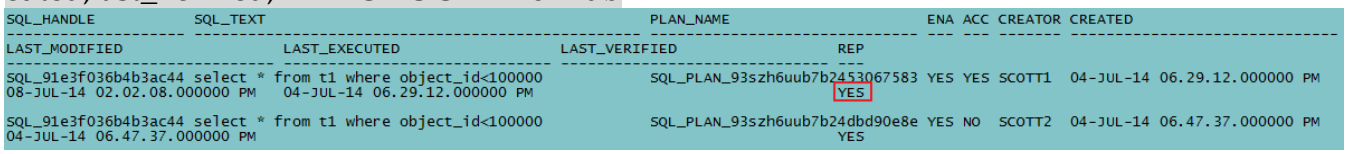
2 - access("OBJECT_ID"<100000)

Note

- SQL plan baseline "SQL_PLAN_93szh6uub7b2453067583" used for this statement

select

sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_executed,last_verified,REPRODUCED from db



```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREATOR CREATED
-------------------- ------------------------------------------------- ------------------------------ --- --- ------- ------------------------------
LAST_MODIFIED                  LAST_EXECUTED                 LAST_VERIFIED                 REP
------------------------------ ----------------------------- ----------------------------- ---
SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000           SQL_PLAN_93szh6uub7b2453067583 YES YES SCOTT1  04-JUL-14 06.29.12.000000 PM
08-JUL-14 02.02.08.000000 PM   04-JUL-14 06.29.12.000000 PM                                  YES

SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000           SQL_PLAN_93szh6uub7b24dbd90e8e YES NO  SCOTT2  04-JUL-14 06.47.37.000000 PM
04-JUL-14 06.47.37.000000 PM                                                                 YES
```

阶段结论：scott2. t1 表与 scott2.t1 完全相同，这个相同包括表结构、索引名称、统计信息等都和 scott1.t1 保持一致，所以生成的执行计划能完全匹配 scott1 走索引的 plan_name，REPRODUCED 重新置为 YES

场景(3): Scott2.t1(object_id)字段创建 non-unique 索引， 索引名称和 Scott1 保持一致，人工增大 clustering_factor，使 Optimizer 偏向选择 FTS

##创建 Scott2.t1(object_id)索引

。。。步骤同上，此处省略

##先把 optimizer_use_sql_plan_baselines 设成 false，观察一下未启用 sql plan baseline 的情况下，改大 scott2.t1
表索引的 clustering_factor 值，对执行计划的影响

---修改前走的是 index range scan

```
alter session set optimizer_use_sql_plan_baselines=FALSE;

select table_name,index_name,clustering_factor from user_indexes where
       table_name='T1';
TABLE_NAME                     INDEX_NAME
       CLUSTERING_FACTOR
------------------------------ ------------------------------ -----------------
T1                             IND_OBJID_T
       10126

set autotrace traceonly
select * from t1 where object_id<100000;

| Id    | Operation                    | Name            | Rows    | Bytes | Cost
       (%CPU)|
 Time         |

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

|     0 | SELECT STATEMENT             |                 |    3560 |      337K|      213
       (0)|
 00:00:03 |

|     1 |    TABLE ACCESS BY INDEX ROWID| T1              |    3560 |      337K|
       213      (0)|
 00:00:03 |
```

```
|*  2 |     INDEX RANGE SCAN                | IND_OBJID_T |    3560 |           |       10
       (0)|
 00:00:01 |
```

---修改后走的是 fts

```
exec
       dbms_stats.set_index_stats(ownname=>'SCOTT2',indname=>'IND_OBJID_T',clstf
       ct=>2000000);

select table_name,index_name,clustering_factor from user_indexes where
       table_name='T1';

TABLE_NAME                     INDEX_NAME
       CLUSTERING_FACTOR
------------------------------ ------------------------------ -----------------
T1                             IND_OBJID_T
       2000000

set autotrace traceonly
select * from t1 where object_id<100000;
-----------------------------------------------------------------------------
| Id  | Operation                | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |      |  3560 |   337K|   456   (1)| 00:00:06 |
|*  1 |  TABLE ACCESS FULL| T1   |  3560 |   337K|   456   (1)| 00:00:06 |
-----------------------------------------------------------------------------
```

##optimizer_use_sql_plan_baselines 置为 true，观察在启用 sql plan baseline 的情况下，在 IND_OBJID_T 索引统

计信息改变之后，oracle 是否还会继续去启用

plan_name=SQL_PLAN_93szh6uub7b2453067583 这条走索引的 plan

--为使结果更为明朗，这里先删除掉 scott2 用户在场景(1)里创建出的走 FTS 的 plan

```
set serveroutput on
declare
result_int pls_integer;
begin
result_int:=dbms_spm.drop_sql_plan_baseline(sql_handle=>'SQL_91e3f036b4b3ac44',pl
an_name=>'SQL_PLAN_93szh6uub7b24dbd90e8e');
dbms_output.put_line(result_int);
end;
/
```

--只剩一条走索引的 plan= SQL_PLAN_93szh6uub7b2453067583

```
Select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_exec
       uted,last_verified,REPRO
DUCED from dba_sql_plan_baselines;
```

```
SQL_HANDLE           SQL_TEXT                                        PLAN_NAME                      ENA ACC CREATOR CREATED
-------------------- ----------------------------------------------- ------------------------------ --- --- ------- ------------------------------
LAST_MODIFIED                  LAST_EXECUTED                 LAST_VERIFIED                  REP
------------------------------ ----------------------------- ------------------------------ ---
SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000        SQL_PLAN_93szh6uub7b2453067583 YES YES SCOTT1  04-JUL-14 06.29.12.000000 PM
08-JUL-14 02.02.08.000000 PM   04-JUL-14 06.29.12.000000 PM                                  YES
```

--scott2 用户执行 sql，plan= SQL_PLAN_93szh6uub7b2453067583 会被启用

```
alter session set optimizer_use_sql_plan_baselines=TRUE;
set autotrace traceonly
select * from t1 where object_id<100000;
```

```
| Id  | Operation                   | Name        | Rows  | Bytes | Cost
      (%CPU)|
 Time      |

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

|   0 | SELECT STATEMENT            |             |  3560 |   337K| 40066
      (1)|
 00:08:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| T1          |  3560 |   337K| 40066
      (1)|
 00:08:01 |

|*  2 |   INDEX RANGE SCAN          | IND_OBJID_T |  3560 |       |    10
      (0)|
 00:00:01 |

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

Predicate Information (identified by operation id):

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

   2 - access("OBJECT_ID"<100000)

Note
-----
   - SQL plan baseline "SQL_PLAN_93szh6uub7b2453067583" used for this statement
```

---但同时也会生成一个 FTS 的 plan，clustering_factor 值远大于 table 所占用的 blocks 的

情况下，、优化器认为
FTS 才是合适的选择

```
select
sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_exec
uted,last_verified,REPRO
DUCED from dba_sql_plan_baselines
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREATOR CREATED
-------------------- ------------------------------------------------- ------------------------------ --- --- ------- ---------------------------------
LAST_MODIFIED                 LAST_EXECUTED                 LAST_VERIFIED                 REP
----------------------------- ----------------------------- ----------------------------- ---
SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000           SQL_PLAN_93szh6uub7b2453067583 YES YES SCOTT1  04-JUL-14 06.29.12.000000 PM
08-JUL-14 02.02.08.000000 PM   04-JUL-14 06.29.12.000000 PM                                 YES

SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000           SQL_PLAN_93szh6uub7b24dbd90e8e YES NO  SCOTT2  08-JUL-14 02.49.36.000000 PM
08-JUL-14 02.49.36.000000 PM                                                                YES
```

阶段结论：只要 sql plan baseline 的 reproduced!=NO，就一定会被优化器选中，哪怕这条 baseline 对应的执行计划效率再差。与此同时优化器执行 sql 时还是要去收集所执行对象的统计信息，并且把它计算出的执行计划添加到 sql plan history 作为演进时的候选对象。

场景(4): Scott2.t1(object_id)字段创建 non-unique 索引，索引名称有别于 Scott1

```
##修改 Scott2.ind_objid_t 索引名称
alter index scott2.IND_OBJID_T rename to IND_OBJID_T2;

exec
dbms_stats.gather_table_stats(ownname=>'scott2',tabname=>'t1',method_opt=>'for all
columns size
1',cascade=>TRUE,no_invalidate=>FALSE);
```

##重新执行 sql，得到了不同的执行计划(这里的不同主要是指索引名称的改变，访问的路径还是 index range scan)，结果是在 dba_sql_plan_baseline 里新增了 1 条 plan_name=SQL_PLAN_93szh6uub7b2483309cfd，与此同时还发现 scott1 用户下的 plan_name= SQL_PLAN_93szh6uub7b2453067583 reproduced 属性变为 NO，原因是索引名称变了匹配不上了，即 IND_OBJID_T !=IND_OBJID_T2

```
set autotrace traceonly
select * from t1 where object_id<100000;
```

```
| Id  | Operation                    | Name             | Rows  | Bytes | Cost
(%CPU)
| Time     |

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

|   0 | SELECT STATEMENT             |                  |  3560 |   337K|
213   (0)
| 00:00:03 |
```

```
|   1 |   TABLE ACCESS BY INDEX ROWID| T1           |  3560 |   337K|
213   (0)
| 00:00:03 |

|*  2 |    INDEX RANGE SCAN          | IND_OBJID_T2 |  3560 |       |
10   (0)
| 00:00:01 |
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREATOR CREATED
-------------------- ------------------------------------------------- ------------------------------ --- --- ------- ------------------------------
LAST_MODIFIED                LAST_EXECUTED                LAST_VERIFIED                REP
---------------------------- ---------------------------- ---------------------------- ---
SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000          SQL_PLAN_93szh6uub7b2453067583 YES YES SCOTT1  04-JUL-14 06.29.12.000000 PM
08-JUL-14 03.15.52.000000 PM   04-JUL-14 06.29.12.000000 PM                                NO

SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000          SQL_PLAN_93szh6uub7b2483309cfd YES NO  SCOTT2  08-JUL-14 03.15.52.000000 PM
08-JUL-14 03.15.52.000000 PM                                                               YES

SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000          SQL_PLAN_93szh6uub7b24dbd90e8e YES NO  SCOTT2  08-JUL-14 02.49.36.000000 PM
08-JUL-14 02.49.36.000000 PM                                                               YES
```

阶段结论：虽然我们平时关注的主要是执行计划中的 access-path 部分，但其实索引名称也是执行计划的重要组成部分也是决定 sql plan baseline 能否被重用的一个重要因素

场景(5): Scott2.t1(object_id)字段创建 unique 索引，索引名称和 Scott1 保持一致

```
##scott2 重建索引，名称和 scott1 名称等同，但索引类型变为 unique
--先 Drop 掉 creator=scott2 的两条 sql plan
set serveroutput on
declare
result_int1 pls_integer;
result_int2 pls_integer;
begin
result_int1:=dbms_spm.drop_sql_plan_baseline(sql_handle=>'SQL_91e3f036b4b3ac44',p
   lan_name=>'SQL_PLAN_93szh6uub7b24dbd90e8e');
result_int2:=dbms_spm.drop_sql_plan_baseline(sql_handle=>'SQL_91e3f036b4b3ac44',p
   lan_name=>'SQL_PLAN_93szh6uub7b2483309cfd');
dbms_output.put_line(result_int1);
dbms_output.put_line(result_int2);
end;
/

select
   sql_handle,sql_text,plan_name,enabled,accepted,creator,created,last_modified,last_ex
   ecuted,last_verified,REPRODUCED from dba_sql_plan_baselines;
```

```
SQL_HANDLE           SQL_TEXT                                          PLAN_NAME                      ENA ACC CREATOR CREATED
-------------------- ------------------------------------------------- ------------------------------ --- --- ------- ------------------------------
LAST_MODIFIED                LAST_EXECUTED                LAST_VERIFIED                REP
---------------------------- ---------------------------- ---------------------------- ---
SQL_91e3f036b4b3ac44 select * from t1 where object_id<100000          SQL_PLAN_93szh6uub7b2453067583 YES YES SCOTT1  04-JUL-14 06.29.12.000000 PM
08-JUL-14 03.15.52.000000 PM   04-JUL-14 06.29.12.000000 PM                                NO
```

```
--重建 scott2.t1 上的索引
drop index scott2.ind_objid_t2;
create unique index scott2.ind_objid_t on scott2.t1(object_id) tablespace ts_pub;
exec
   dbms_stats.gather_table_stats(ownname=>'scott2',tabname=>'t1',method_opt=>'for
   all columns size 1',cascade=>TRUE,no_invalidate=>FALSE);
```

##scott2 执行 sql 观察到 scott1 用户的 plan_name=
SQL_PLAN_93szh6uub7b2453067583 还是能够被利用

```
set autotrace traceonly
select * from t1 where object_id<100000;
| Id  | Operation                           | Name            | Rows  | Bytes | Cost
  (%CPU)|
 Time     |

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

|   0 | SELECT STATEMENT                    |                 |  3560 |   337K|   212
  (0)|
 00:00:03 |

|   1 |  TABLE ACCESS BY INDEX ROWID| T1              |  3560 |   337K|
  212   (0)|
 00:00:03 |

|* 2 |   INDEX RANGE SCAN                  | IND_OBJID_T |  3560 |       |     9
  (0)|
 00:00:01 |

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


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

   2 - access("OBJECT_ID"<100000)

Note
-----
   - SQL plan baseline "SQL_PLAN_93szh6uub7b2453067583" used for this statement
```

阶段结论：虽然这次索引变成了 unique 的，但执行计划中并没有使用 index unique scan，用的依然是 index range scan，这就和 plan=SQL_PLAN_93szh6uub7b2453067583 所指向的 access-path 保持一致，说明只要在 access-path，索引名称相同的情况下，oracle 不会对索引是否为 unique 有强制的要求

场景(6): 重建 Scott2.t1 表，同时更改 scott2.t1 表结构，除了 object_id 字段外，其余字段均和 Scott2.t1 中的字段不相同

##重构 Scott2.t1 表

```
drop table scott2.t1;
```

```
create table scott2.t1 (col1 varchar2(2),object_id number,col3 varchar2(100)) tablespace
ts_pub;

declare
begin
for i in 1..170000 loop
insert into scott2.t1 values('AA',i,'scott2.t1');
end loop;
commit;
end;
/

create index scott2.ind_objid_t on scott2.t1(object_id) tablespace ts_pub;
exec
dbms_stats.gather_table_stats(ownname=>'scott2',tabname=>'t1',method_opt=>'for all
columns size
1',cascade=>TRUE,no_invalidate=>FALSE);

##scott 用户执行 sql，sql plan baseline 能够被重用
set autotrace traceonly
select * from t1 where object_id<100000;

| Id  | Operation                    | Name        | Rows  | Bytes | Cost
(%CPU)|
   Time     |

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

|   0 | SELECT STATEMENT             |             |   100K|  1757K|   545
(1)|
   00:00:07 |

|   1 |  TABLE ACCESS BY INDEX ROWID| T1          |   100K|  1757K|
545   (1)|
   00:00:07 |

|*  2 |   INDEX RANGE SCAN           | IND_OBJID_T |   100K|       |
225   (1)|
   00:00:03 |

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


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

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

   2 - access("OBJECT_ID"<100000)

Note
-----
   - SQL plan baseline "SQL_PLAN_93szh6uub7b2453067583" used for this statement
```

阶段结论：只要执行计划能完全匹配上，就能利用到已生成的 sql plan baseline，对于表结构，
表内容等项目 oracle 不作检查，可见 sql plan baseline 对环境的适应能力是很强的，除了对象不可用之外(例如索引被删除)，都能将预先生成的执行计划提供给优化器执行。

作者个人简介

崔宏慧

任职于中国移动集团上海有限公司，主要从事系统维护工作，先后担任过系统管理员，存储管理员，对 AIX 操作系统、EMC 存储相关领域知识有较为深刻的理解，07 年开始接触 Oracle，目前担任帐务系统数据库 DBA，期间参与了上海移动异地容灾系统建设并多次主导核心数据库容灾演练、数据库升版、数据库存储迁移、BOSS 系统重构等项目。