```
SPO c:\temp\coe_xfr_sql_profile.log;
SET DEF ON TERM OFF ECHO ON FEED OFF VER OFF HEA ON LIN 2000 PAGES 100 LONG 8000000 LONGC 800000 TRIMS ON TI OFF TIMI
OFF SERVEROUT ON SIZE 1000000 NUMF "" SQLP SQL>;
REM
REM $Header: 215187.1 coe_xfr_sql_profile.sql 11.4.1.4 2010/07/12 csierra $
REM
REM Copyright (c) 2000-2010, Oracle Corporation. All rights reserved.
REM
REM AUTHOR
REM   carlos.sierra@oracle.com
REM
REM SCRIPT
REM   coe_xfr_sql_profile.sql
REM
REM DESCRIPTION
REM   This script generates another that contains the commands to
REM   create a manual custom SQL Profile out of a known plan from
REM   memory or AWR. The manual custom profile can be implemented
REM   into the same SOURCE system where the plan was retrieved,
REM   or into another similar TARGET system that has same schema
REM   objects referenced by the SQL that generated the known plan.
REM
REM PRE-REQUISITES
REM   1. Oracle Tuning Pack license.
REM
REM PARAMETERS
REM   1. SQL_ID (required)
REM   2. Plan Hash Value for which a manual custom SQL Profile is
REM      needed (required). A list of known plans is presented.
REM
REM EXECUTION
REM   1. Connect into SQL*Plus as SYSDBA or user with access to
REM      data dictionary.
REM   2. Execute script coe_xfr_sql_profile.sql passing SQL_ID and
REM      plan hash value (parameters can be passed inline or until
REM      requested).
REM
REM EXAMPLE
REM   # sqlplus system
REM   SQL> START coe_xfr_sql_profile.sql [SQL_ID] [PLAN_HASH_VALUE];
REM   SQL> START coe_xfr_sql_profile.sql gnjy0mn4y9pbm 2055843663;
REM   SQL> START coe_xfr_sql_profile.sql gnjy0mn4y9pbm;
REM   SQL> START coe_xfr_sql_profile.sql;
REM
REM NOTES
REM   1. For possible errors see coe_xfr_sql_profile.log
REM   2. If SQLT is installed in SOURCE, you can use instead:
REM      sqlt/utl/sqltprofile.sql
REM   3. Be aware that using DBMS_SQLTUNE requires a license for
REM      Oracle Tuning Pack.
REM
SET TERM ON ECHO OFF;
PRO
PRO Parameter 1:
PRO SQL_ID (required)
PRO
DEF sql_id = '&1';
PRO
WITH
p AS (
SELECT plan_hash_value
  FROM gv$sql_plan
 WHERE sql_id = TRIM('&&sql_id.')
   AND other_xml IS NOT NULL
 UNION
SELECT plan_hash_value
  FROM dba_hist_sql_plan
 WHERE sql_id = TRIM('&&sql_id.')
   AND other_xml IS NOT NULL ),
m AS (
SELECT plan_hash_value,
```

```
       SUM(elapsed_time)/SUM(executions) avg_et_secs
  FROM gv$sql
 WHERE sql_id = TRIM('&&sql_id.')
   AND executions > 0
 GROUP BY
       plan_hash_value ),
a AS (
SELECT plan_hash_value,
       SUM(elapsed_time_total)/SUM(executions_total) avg_et_secs
  FROM dba_hist_sqlstat
 WHERE sql_id = TRIM('&&sql_id.')
   AND executions_total > 0
 GROUP BY
       plan_hash_value )
SELECT p.plan_hash_value,
       ROUND(NVL(m.avg_et_secs, a.avg_et_secs)/1e6, 3) avg_et_secs
  FROM p, m, a
 WHERE p.plan_hash_value = m.plan_hash_value(+)
   AND p.plan_hash_value = a.plan_hash_value(+)
 ORDER BY
       avg_et_secs NULLS LAST;
PRO
PRO Parameter 2:
PRO PLAN_HASH_VALUE (required)
PRO
DEF plan_hash_value = '&2';
PRO
PRO Values passed:
PRO ~~~~~~~~~~~~~
PRO SQL_ID         : "&&sql_id."
PRO PLAN_HASH_VALUE: "&&plan_hash_value."
PRO
SET TERM OFF ECHO ON;
WHENEVER SQLERROR EXIT SQL.SQLCODE;

VAR sql_text CLOB;
VAR other_xml CLOB;
EXEC :sql_text := NULL;
EXEC :other_xml := NULL;

-- get sql_text from memory
DECLARE
  l_sql_text VARCHAR2(32767);
BEGIN -- 10g see bug 5017909
  FOR i IN (SELECT DISTINCT piece, sql_text
              FROM gv$sqltext_with_newlines
             WHERE sql_id = TRIM('&&sql_id.')
             ORDER BY 1, 2)
  LOOP
    IF :sql_text IS NULL THEN
      DBMS_LOB.CREATETEMPORARY(:sql_text, TRUE);
      DBMS_LOB.OPEN(:sql_text, DBMS_LOB.LOB_READWRITE);
    END IF;
    l_sql_text := REPLACE(i.sql_text, CHR(00), ' ');
    DBMS_LOB.WRITEAPPEND(:sql_text, LENGTH(l_sql_text), l_sql_text);
  END LOOP;
  IF :sql_text IS NOT NULL THEN
    DBMS_LOB.CLOSE(:sql_text);
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('getting sql_text from memory: '||SQLERRM);
    :sql_text := NULL;
END;
/

-- get sql_text from awr
BEGIN
  IF :sql_text IS NULL OR NVL(DBMS_LOB.GETLENGTH(:sql_text), 0) = 0 THEN
    SELECT REPLACE(sql_text, CHR(00), ' ')
      INTO :sql_text
```

```
      FROM dba_hist_sqltext
     WHERE sql_id = TRIM('&&sql_id.')
       AND sql_text IS NOT NULL
       AND ROWNUM = 1;
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('getting sql_text from awr: '||SQLERRM);
    :sql_text := NULL;
END;
/

SELECT :sql_text FROM DUAL;

-- get other_xml from memory
BEGIN
  FOR i IN (SELECT other_xml
              FROM gv$sql_plan
             WHERE sql_id = TRIM('&&sql_id.')
               AND plan_hash_value = TO_NUMBER(TRIM('&&plan_hash_value.'))
               AND other_xml IS NOT NULL
             ORDER BY
                   child_number, id)
  LOOP
    :other_xml := i.other_xml;
    EXIT; -- 1st
  END LOOP;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('getting other_xml from memory: '||SQLERRM);
    :other_xml := NULL;
END;
/

-- get other_xml from awr
BEGIN
  IF :other_xml IS NULL OR NVL(DBMS_LOB.GETLENGTH(:other_xml), 0) = 0 THEN
    FOR i IN (SELECT other_xml
                FROM dba_hist_sql_plan
               WHERE sql_id = TRIM('&&sql_id.')
                 AND plan_hash_value = TO_NUMBER(TRIM('&&plan_hash_value.'))
                 AND other_xml IS NOT NULL
               ORDER BY
                     id)
    LOOP
      :other_xml := i.other_xml;
      EXIT; -- 1st
    END LOOP;
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('getting other_xml from awr: '||SQLERRM);
    :other_xml := NULL;
END;
/

SELECT :other_xml FROM DUAL;

-- generates script that creates sql profile in target system:
SET ECHO OFF;
PRO coe_xfr_sql_profile_&&sql_id._&&plan_hash_value..sql.
SET FEED OFF LIN 666 TRIMS ON TI OFF TIMI OFF SERVEROUT ON SIZE 1000000 FOR WOR;
SPO OFF;
SPO c:\temp\coe_xfr_sql_profile_&&sql_id._&&plan_hash_value..sql;
DECLARE
  l_pos NUMBER;
  l_hint VARCHAR2(32767);
BEGIN
  DBMS_OUTPUT.PUT_LINE('SPO c:\temp\coe_xfr_sql_profile_&&sql_id._&&plan_hash_value..log;');
  DBMS_OUTPUT.PUT_LINE('SET ECHO ON TERM ON LIN 2000 TRIMS ON NUMF 99999999999999999999;');
  DBMS_OUTPUT.PUT_LINE('REM');
```

```
  DBMS_OUTPUT.PUT_LINE('REM $Header: 215187.1 coe_xfr_sql_profile_&&sql_id._&&plan_hash_value..sql 11.4.1.4
'||TO_CHAR(SYSDATE, 'YYYY/MM/DD')||' csierra $');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('REM Copyright (c) 2000-2010, Oracle Corporation. All rights reserved.');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('REM AUTHOR');
  DBMS_OUTPUT.PUT_LINE('REM   carlos.sierra@oracle.com');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('REM SCRIPT');
  DBMS_OUTPUT.PUT_LINE('REM   coe_xfr_sql_profile_&&sql_id._&&plan_hash_value..sql');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('REM DESCRIPTION');
  DBMS_OUTPUT.PUT_LINE('REM   This script is generated by coe_xfr_sql_profile.sql');
  DBMS_OUTPUT.PUT_LINE('REM   It contains the SQL*Plus commands to create a custom');
  DBMS_OUTPUT.PUT_LINE('REM   SQL Profile for SQL_ID &&sql_id. based on plan hash');
  DBMS_OUTPUT.PUT_LINE('REM   value &&plan_hash_value..');
  DBMS_OUTPUT.PUT_LINE('REM   The custom SQL Profile to be created by this script');
  DBMS_OUTPUT.PUT_LINE('REM   will affect plans for SQL commands with signature');
  DBMS_OUTPUT.PUT_LINE('REM   matching the one for SQL Text below.');
  DBMS_OUTPUT.PUT_LINE('REM   Review SQL Text and adjust accordingly.');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('REM PARAMETERS');
  DBMS_OUTPUT.PUT_LINE('REM   None.');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('REM EXAMPLE');
  DBMS_OUTPUT.PUT_LINE('REM   SQL> START coe_xfr_sql_profile_&&sql_id._&&plan_hash_value..sql;');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('REM NOTES');
  DBMS_OUTPUT.PUT_LINE('REM   1. Should be run as SYSTEM or SYSDBA.');
  DBMS_OUTPUT.PUT_LINE('REM   2. User must have CREATE ANY SQL PROFILE privilege.');
  DBMS_OUTPUT.PUT_LINE('REM   3. SOURCE and TARGET systems can be the same or similar.');
  DBMS_OUTPUT.PUT_LINE('REM   4. To drop this custom SQL Profile after it has been created:');
  DBMS_OUTPUT.PUT_LINE('REM      EXEC DBMS_SQLTUNE.DROP_SQL_PROFILE(''coe_&&sql_id._&&plan_hash_value.'');');
  DBMS_OUTPUT.PUT_LINE('REM   5. Be aware that using DBMS_SQLTUNE requires a license');
  DBMS_OUTPUT.PUT_LINE('REM      for the Oracle Tuning Pack.');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('WHENEVER SQLERROR EXIT SQL.SQLCODE;');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('VAR signature NUMBER;');
  DBMS_OUTPUT.PUT_LINE('REM');
  DBMS_OUTPUT.PUT_LINE('DECLARE');
  DBMS_OUTPUT.PUT_LINE('sql_txt CLOB;');
  DBMS_OUTPUT.PUT_LINE('h       SYS.SQLPROF_ATTR;');
  DBMS_OUTPUT.PUT_LINE('BEGIN');
  DBMS_OUTPUT.PUT_LINE('sql_txt := q''[');
  WHILE NVL(LENGTH(:sql_text), 0) > 0
  LOOP
    l_pos := INSTR(:sql_text, CHR(10));
    IF l_pos > 0 THEN
      DBMS_OUTPUT.PUT_LINE(SUBSTR(:sql_text, 1, l_pos - 1));
      :sql_text := SUBSTR(:sql_text, l_pos + 1);
    ELSE
      DBMS_OUTPUT.PUT_LINE(:sql_text);
      :sql_text := NULL;
    END IF;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE(']'';');
  DBMS_OUTPUT.PUT_LINE('h := SYS.SQLPROF_ATTR(');
  DBMS_OUTPUT.PUT_LINE('q''[BEGIN_OUTLINE_DATA]'',');
  FOR i IN (SELECT /*+ opt_param('parallel_execution_enabled', 'false') */
                   SUBSTR(EXTRACTVALUE(VALUE(d), '/hint'), 1, 4000) hint
             FROM TABLE(XMLSEQUENCE(EXTRACT(XMLTYPE(:other_xml), '/*/outline_data/hint'))) d)
  LOOP
    l_hint := i.hint;
    WHILE NVL(LENGTH(l_hint), 0) > 0
    LOOP
      IF LENGTH(l_hint) <= 500 THEN
        DBMS_OUTPUT.PUT_LINE('q''['||l_hint||']'',');
        l_hint := NULL;
      ELSE
        l_pos := INSTR(SUBSTR(l_hint, 1, 500), ' ', -1);
```

```
        DBMS_OUTPUT.PUT_LINE('q''['||SUBSTR(l_hint, 1, l_pos)||']'',');
        l_hint := '   '||SUBSTR(l_hint, l_pos);
      END IF;
    END LOOP;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('q''[END_OUTLINE_DATA]'');');
  DBMS_OUTPUT.PUT_LINE(':signature := DBMS_SQLTUNE.SQLTEXT_TO_SIGNATURE(sql_txt);');
  DBMS_OUTPUT.PUT_LINE('DBMS_SQLTUNE.IMPORT_SQL_PROFILE (');
  DBMS_OUTPUT.PUT_LINE('sql_text    => sql_txt,');
  DBMS_OUTPUT.PUT_LINE('profile     => h,');
  DBMS_OUTPUT.PUT_LINE('name        => ''coe_&&sql_id._&&plan_hash_value.'',');
  DBMS_OUTPUT.PUT_LINE('description => ''coe &&sql_id. &&plan_hash_value. ''||:signature||'''',');
  DBMS_OUTPUT.PUT_LINE('category    => ''DEFAULT'',');
  DBMS_OUTPUT.PUT_LINE('validate    => TRUE,');
  DBMS_OUTPUT.PUT_LINE('replace     => TRUE,');
  DBMS_OUTPUT.PUT_LINE('force_match => FALSE /* TRUE:FORCE (match even when different literals in SQL). FALSE:EXACT
(similar to CURSOR_SHARING) */ );');
  DBMS_OUTPUT.PUT_LINE('END;');
  DBMS_OUTPUT.PUT_LINE('/');
  DBMS_OUTPUT.PUT_LINE('WHENEVER SQLERROR CONTINUE');
  DBMS_OUTPUT.PUT_LINE('SET ECHO OFF;');
  DBMS_OUTPUT.PUT_LINE('PRINT signature');
  DBMS_OUTPUT.PUT_LINE('PRO');
  DBMS_OUTPUT.PUT_LINE('PRO ... manual custom SQL Profile has been created');
  DBMS_OUTPUT.PUT_LINE('PRO');
  DBMS_OUTPUT.PUT_LINE('SET TERM ON ECHO OFF LIN 80 TRIMS OFF NUMF "";');
  DBMS_OUTPUT.PUT_LINE('SPO OFF;');
  DBMS_OUTPUT.PUT_LINE('PRO');
  DBMS_OUTPUT.PUT_LINE('PRO COE_XFR_SQL_PROFILE_&&sql_id._&&plan_hash_value. completed');
END;
/
SPO OFF;
SET DEF ON TERM ON ECHO OFF FEED 6 VER ON HEA ON LIN 80 PAGES 14 LONG 80 LONGC 80 TRIMS OFF TI OFF TIMI OFF SERVEROUT
OFF NUMF "" SQLP SQL>;
PRO
PRO Execute coe_xfr_sql_profile_&&sql_id._&&plan_hash_value..sql
PRO on TARGET system in order to create a custom SQL Profile
PRO with plan &&plan_hash_value linked to adjusted sql_text.
PRO
UNDEFINE 1 2 sql_id plan_hash_value
PRO
PRO COE_XFR_SQL_PROFILE completed.
```