KENT STATE UNIVERSITY

ORACLE/BANNER TUNING

5/19/2016
Danielle Tricker
dtricker@kent.edu

Agenda

- Who am I – Nine years at Kent as an Oracle/Banner DBA.

  Recently 12c Certified, previous certs in 9i and 7.

  Just took the spring online 2016 COMT 46315, SQL with Oracle at Kent.

1. When/Why do we need to tune?
2. Explain Plan Overview – For DBA's/Developers
3. When/Why upgrade to 12 – Anyone using 12 yet?
4. 12c Optimizer Overview
5. Proper Table Joins using conventional/ANSI SQL
6. Banner Tips
7. My Ellucian Live 2016 Topics of Interest

1. When/Why do we need to tune

- 1. Number one answer is usually a complaint. Why is this slow?
- 2. Threshold violations from monitoring tools.
- 3. Upgrades change code and functionality sometimes causing new performance issues.
- 4. Poorly written SQL – missing joins
- 5. Software bugs, Ellucian or Oracle.
- 6. Hardware changes including storage (both need to optimized to avoid wasting money)
- 7. OS upgrades/OS changes.
- 8. The Oracle optimizer can make poor decisions for a number of different reasons.
- 9. Improperly set Oracle parameters.
- 10. Improper or missing Oracle statistics.
- 11. Extremely large structures
- 12. Missing or unusable indexes
- 13. Locking or deadlocking issues
- 14. To many triggers!
- 15. Complex queries doing many unions, aggregations, views accessing views, functions, cases….

2. EXPLAIN PLAN OVERVIEWS

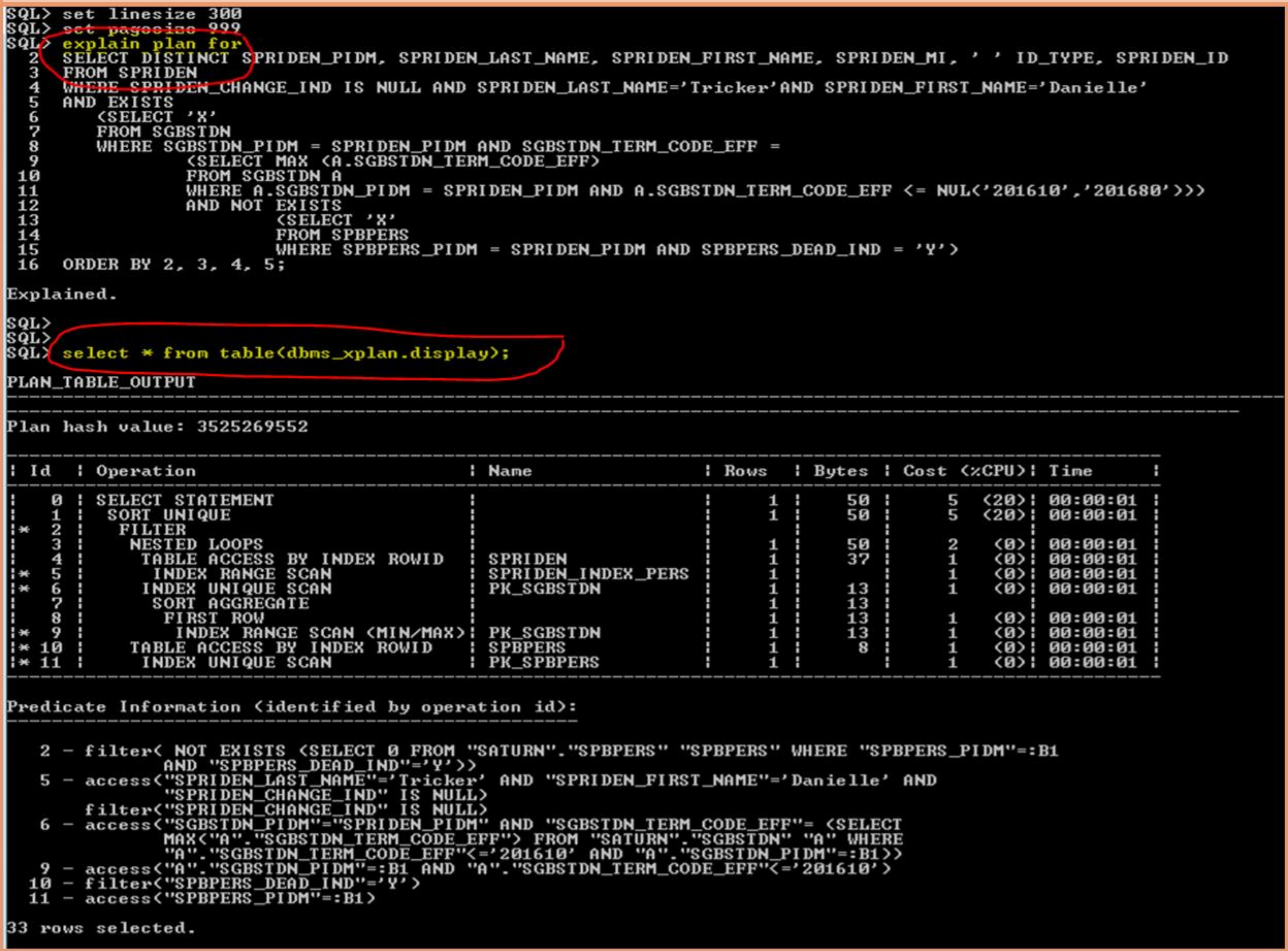
Explain **Autotrace** **Sqltrace**

- Explain plans are like old fashioned Mapquest directions – they show the expected path to a destination, but the path could change at a later date depending on traffic or construction. It is just a prediction.
- Explain plans can be generated without providing bind variable data
- To Use: "explain plan for" before select statement or use tool – such as Toad
- Then do a select * from table(dbms_xplan.display) – 12c can show adaptive queries

- Autotrace executes the query, so it explains the actual path it took to retrieve the data. There are no guesses on what the plan will be.
- Bind variable data must be provided
- To Use: "set autotrace on" before running query

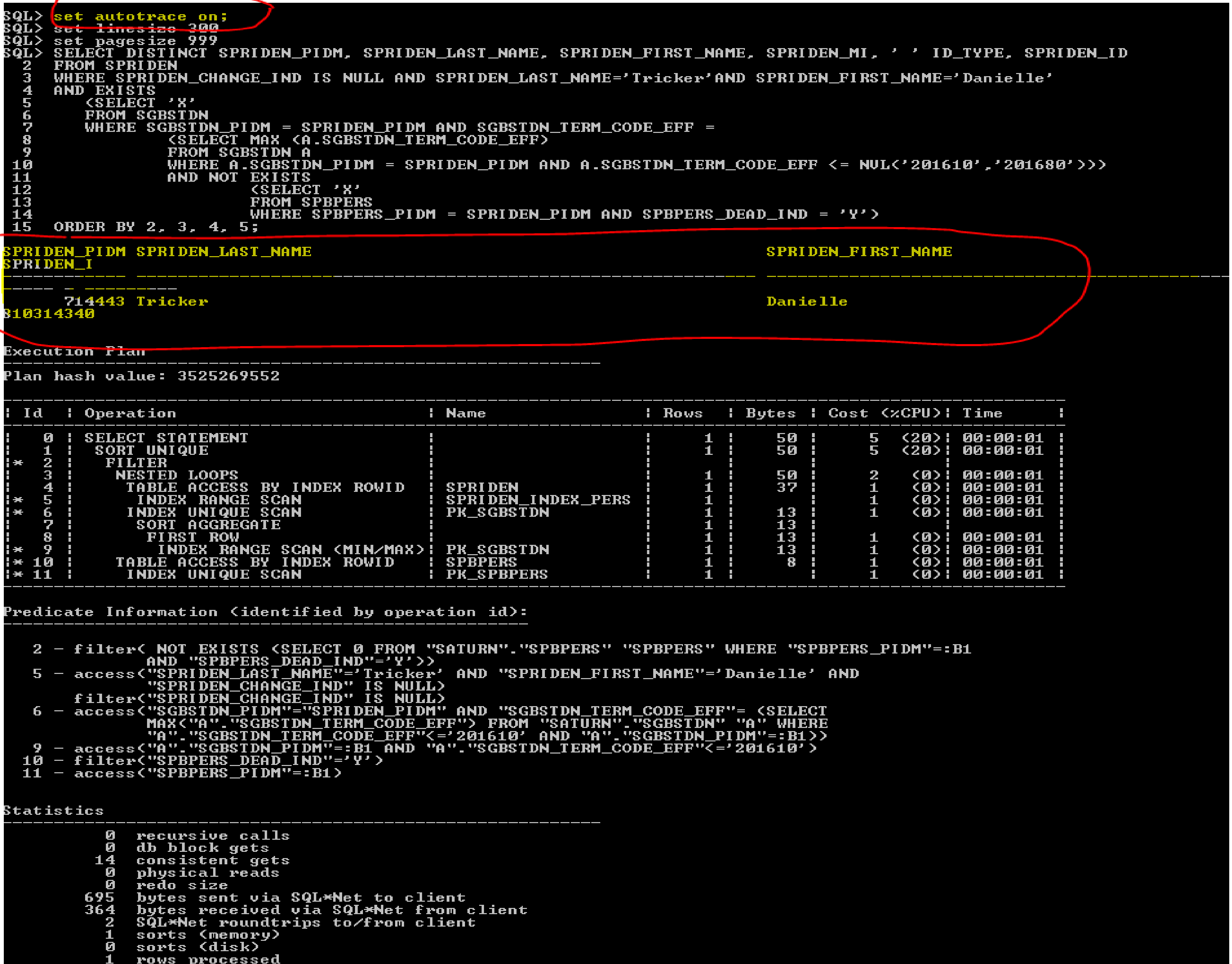
- Sqltrace is used to generate tkprof trace reports
- To Use: "alter session set sql_trace = true" - Make sure you don't accidentally type alter system instead of alter session.

EXPLAIN PLAN – This will change with 12c adaptive execution plans



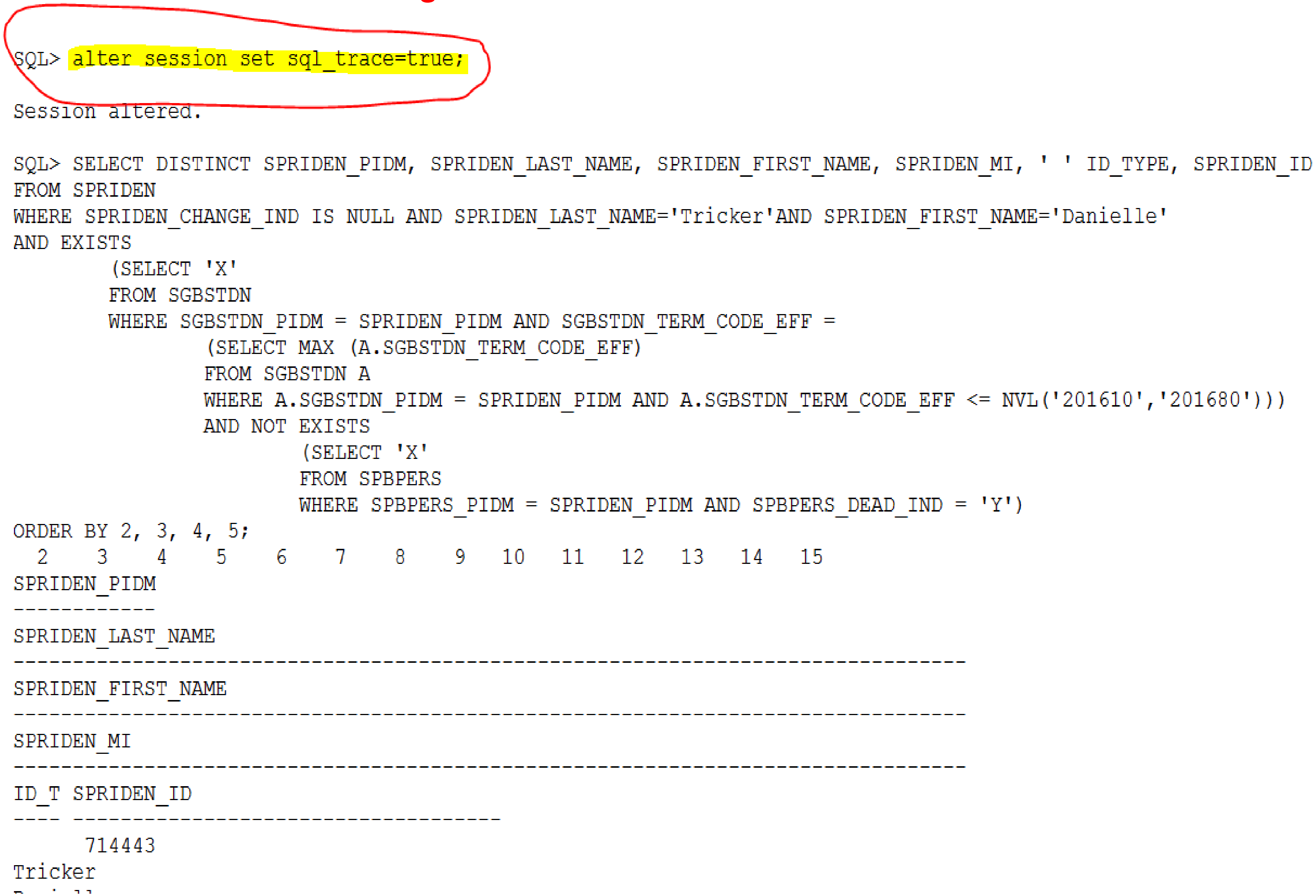
```
SQL> set linesize 300
SQL> set pagesize 999
SQL> explain plan for
  2  SELECT DISTINCT SPRIDEN_PIDM, SPRIDEN_LAST_NAME, SPRIDEN_FIRST_NAME, SPRIDEN_MI, ' ' ID_TYPE, SPRIDEN_ID
  3  FROM SPRIDEN
  4  WHERE SPRIDEN_CHANGE_IND IS NULL AND SPRIDEN_LAST_NAME='Tricker'AND SPRIDEN_FIRST_NAME='Danielle'
  5  AND EXISTS
  6      (SELECT 'X'
  7      FROM SGBSTDN
  8      WHERE SGBSTDN_PIDM = SPRIDEN_PIDM AND SGBSTDN_TERM_CODE_EFF =
  9              (SELECT MAX (A.SGBSTDN_TERM_CODE_EFF)
 10              FROM SGBSTDN A
 11              WHERE A.SGBSTDN_PIDM = SPRIDEN_PIDM AND A.SGBSTDN_TERM_CODE_EFF <= NVL('201610','201680')))
 12              AND NOT EXISTS
 13                      (SELECT 'X'
 14                      FROM SPBPERS
 15                      WHERE SPBPERS_PIDM = SPRIDEN_PIDM AND SPBPERS_DEAD_IND = 'Y')
 16  ORDER BY 2, 3, 4, 5;

Explained.

SQL>
SQL>
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
Plan hash value: 3525269552

--------------------------------------------------------------------------------------------------------
| Id  | Operation                        | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |                     |     1 |    50 |     5  (20)| 00:00:01 |
|   1 |  SORT UNIQUE                     |                     |     1 |    50 |     5  (20)| 00:00:01 |
|*  2 |   FILTER                         |                     |       |       |            |          |
|   3 |    NESTED LOOPS                  |                     |     1 |    50 |     2   (0)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID  | SPRIDEN             |     1 |    37 |     1   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN            | SPRIDEN_INDEX_PERS  |     1 |       |     1   (0)| 00:00:01 |
|*  6 |     INDEX UNIQUE SCAN            | PK_SGBSTDN          |     1 |    13 |     1   (0)| 00:00:01 |
|   7 |      SORT AGGREGATE              |                     |     1 |    13 |            |          |
|   8 |       FIRST ROW                  |                     |     1 |    13 |     1   (0)| 00:00:01 |
|*  9 |        INDEX RANGE SCAN (MIN/MAX)| PK_SGBSTDN          |     1 |    13 |     1   (0)| 00:00:01 |
|* 10 |    TABLE ACCESS BY INDEX ROWID   | SPBPERS             |     1 |     8 |     1   (0)| 00:00:01 |
|* 11 |     INDEX UNIQUE SCAN            | PK_SPBPERS          |     1 |       |     1   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------------------

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

   2 - filter( NOT EXISTS (SELECT 0 FROM "SATURN"."SPBPERS" "SPBPERS" WHERE "SPBPERS_PIDM"=:B1
              AND "SPBPERS_DEAD_IND"='Y'))
   5 - access("SPRIDEN_LAST_NAME"='Tricker' AND "SPRIDEN_FIRST_NAME"='Danielle' AND
              "SPRIDEN_CHANGE_IND" IS NULL)
       filter("SPRIDEN_CHANGE_IND" IS NULL)
   6 - access("SGBSTDN_PIDM"="SPRIDEN_PIDM" AND "SGBSTDN_TERM_CODE_EFF"= (SELECT
              MAX("A"."SGBSTDN_TERM_CODE_EFF") FROM "SATURN"."SGBSTDN" "A" WHERE
              "A"."SGBSTDN_TERM_CODE_EFF"<='201610' AND "A"."SGBSTDN_PIDM"=:B1))
   9 - access("A"."SGBSTDN_PIDM"=:B1 AND "A"."SGBSTDN_TERM_CODE_EFF"<='201610')
  10 - filter("SPBPERS_DEAD_IND"='Y')
  11 - access("SPBPERS_PIDM"=:B1)

33 rows selected.
```

AUTOTRACE



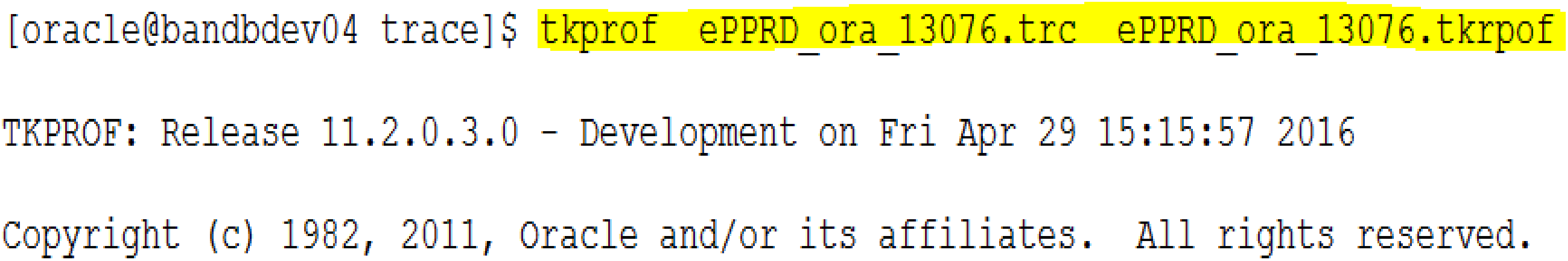
```
SQL> set autotrace on;
SQL> set linesize 300
SQL> set pagesize 999
SQL> SELECT DISTINCT SPRIDEN_PIDM, SPRIDEN_LAST_NAME, SPRIDEN_FIRST_NAME, SPRIDEN_MI, ' ' ID_TYPE, SPRIDEN_ID
  2  FROM SPRIDEN
  3  WHERE SPRIDEN_CHANGE_IND IS NULL AND SPRIDEN_LAST_NAME='Tricker'AND SPRIDEN_FIRST_NAME='Danielle'
  4  AND EXISTS
  5     (SELECT 'X'
  6     FROM SGBSTDN
  7     WHERE SGBSTDN_PIDM = SPRIDEN_PIDM AND SGBSTDN_TERM_CODE_EFF =
  8             (SELECT MAX (A.SGBSTDN_TERM_CODE_EFF)
  9             FROM SGBSTDN A
 10             WHERE A.SGBSTDN_PIDM = SPRIDEN_PIDM AND A.SGBSTDN_TERM_CODE_EFF <= NVL('201610','201680')))
 11             AND NOT EXISTS
 12                     (SELECT 'X'
 13                     FROM SPBPERS
 14                     WHERE SPBPERS_PIDM = SPRIDEN_PIDM AND SPBPERS_DEAD_IND = 'Y')
 15  ORDER BY 2, 3, 4, 5;

SPRIDEN_PIDM SPRIDEN_LAST_NAME                                                            SPRIDEN_FIRST_NAME
SPRIDEN_I
------------ ---------------------------------------------------------------------------- ------------------------------------------------------------------------------------------
----- - ---------
      714443 Tricker                                                                      Danielle
810314340


Execution Plan
----------------------------------------------------------
Plan hash value: 3525269552

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                      |     1 |    50 |     5  (20)| 00:00:01 |
|   1 |  SORT UNIQUE                       |                      |     1 |    50 |     5  (20)| 00:00:01 |
|*  2 |   FILTER                           |                      |       |       |            |          |
|   3 |    NESTED LOOPS                    |                      |     1 |    50 |     2   (0)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID    | SPRIDEN              |     1 |    37 |     1   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN              | SPRIDEN_INDEX_PERS   |     1 |       |     1   (0)| 00:00:01 |
|*  6 |     INDEX UNIQUE SCAN              | PK_SGBSTDN           |     1 |    13 |     1   (0)| 00:00:01 |
|   7 |      SORT AGGREGATE                |                      |     1 |    13 |            |          |
|   8 |       FIRST ROW                    |                      |     1 |    13 |     1   (0)| 00:00:01 |
|*  9 |        INDEX RANGE SCAN (MIN/MAX)  | PK_SGBSTDN           |     1 |    13 |     1   (0)| 00:00:01 |
|* 10 |    TABLE ACCESS BY INDEX ROWID     | SPBPERS              |     1 |     8 |     1   (0)| 00:00:01 |
|* 11 |     INDEX UNIQUE SCAN              | PK_SPBPERS           |     1 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

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

   2 - filter( NOT EXISTS (SELECT 0 FROM "SATURN"."SPBPERS" "SPBPERS" WHERE "SPBPERS_PIDM"=:B1
              AND "SPBPERS_DEAD_IND"='Y'))
   5 - access("SPRIDEN_LAST_NAME"='Tricker' AND "SPRIDEN_FIRST_NAME"='Danielle' AND
              "SPRIDEN_CHANGE_IND" IS NULL)
       filter("SPRIDEN_CHANGE_IND" IS NULL)
   6 - access("SGBSTDN_PIDM"="SPRIDEN_PIDM" AND "SGBSTDN_TERM_CODE_EFF"= (SELECT
              MAX("A"."SGBSTDN_TERM_CODE_EFF") FROM "SATURN"."SGBSTDN" "A" WHERE
              "A"."SGBSTDN_TERM_CODE_EFF"<='201610' AND "A"."SGBSTDN_PIDM"=:B1))
   9 - access("A"."SGBSTDN_PIDM"=:B1 AND "A"."SGBSTDN_TERM_CODE_EFF"<='201610')
  10 - filter("SPBPERS_DEAD_IND"='Y')
  11 - access("SPBPERS_PIDM"=:B1)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         14  consistent gets
          0  physical reads
          0  redo size
        695  bytes sent via SQL*Net to client
        364  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

Turn on SQLTRACE to generate a trace file that TKPROF can use



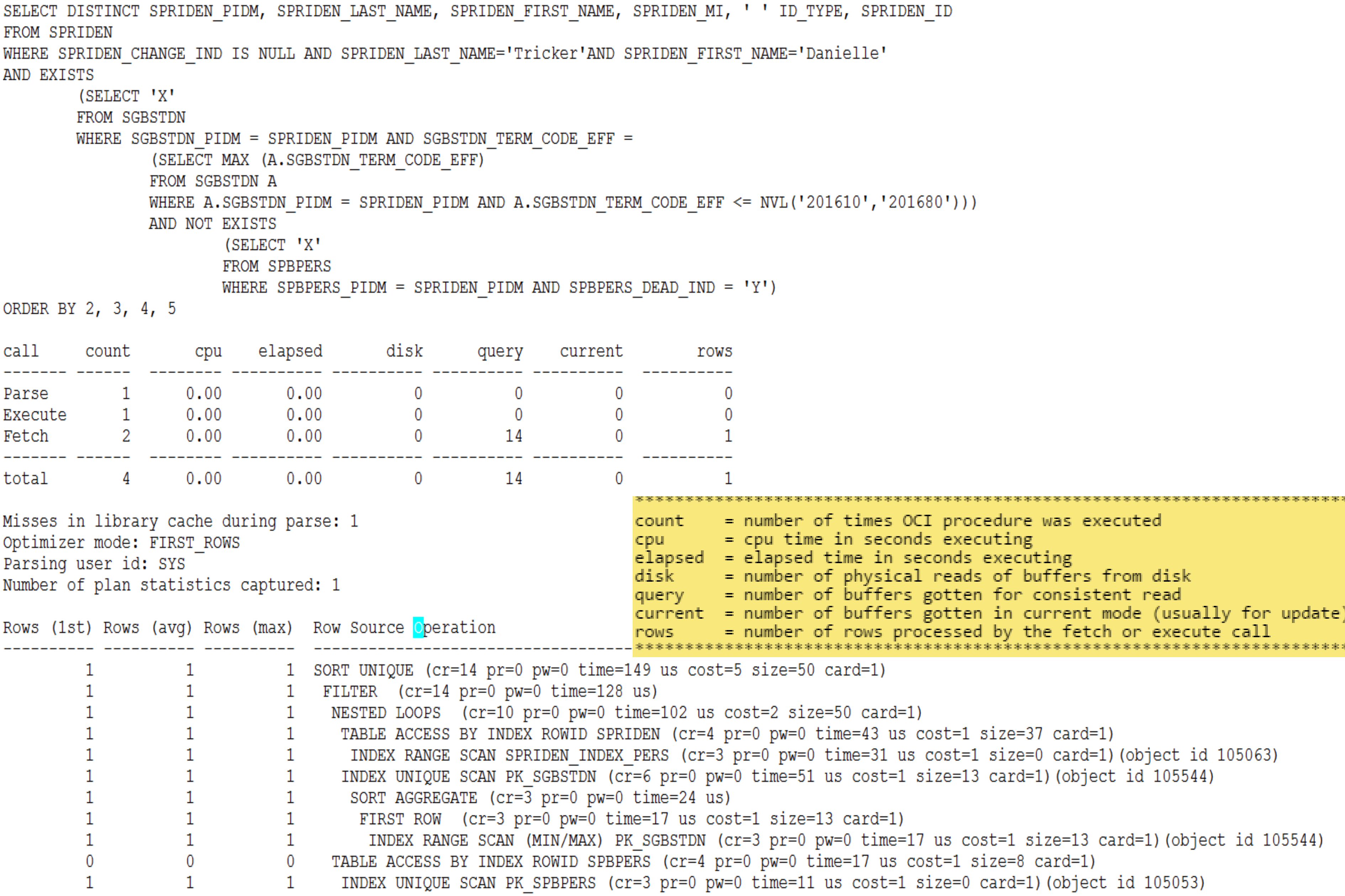
```
SQL> alter session set sql_trace=true;

Session altered.

SQL> SELECT DISTINCT SPRIDEN_PIDM, SPRIDEN_LAST_NAME, SPRIDEN_FIRST_NAME, SPRIDEN_MI, ' ' ID_TYPE, SPRIDEN_ID
FROM SPRIDEN
WHERE SPRIDEN_CHANGE_IND IS NULL AND SPRIDEN_LAST_NAME='Tricker'AND SPRIDEN_FIRST_NAME='Danielle'
AND EXISTS
        (SELECT 'X'
        FROM SGBSTDN
        WHERE SGBSTDN_PIDM = SPRIDEN_PIDM AND SGBSTDN_TERM_CODE_EFF =
                (SELECT MAX (A.SGBSTDN_TERM_CODE_EFF)
                FROM SGBSTDN A
                WHERE A.SGBSTDN_PIDM = SPRIDEN_PIDM AND A.SGBSTDN_TERM_CODE_EFF <= NVL('201610','201680')))
                AND NOT EXISTS
                        (SELECT 'X'
                        FROM SPBPERS
                        WHERE SPBPERS_PIDM = SPRIDEN_PIDM AND SPBPERS_DEAD_IND = 'Y')
ORDER BY 2, 3, 4, 5;
  2    3    4    5    6    7    8    9   10   11   12   13   14   15
SPRIDEN_PIDM
------------
SPRIDEN_LAST_NAME
--------------------------------------------------------------------------------
SPRIDEN_FIRST_NAME
--------------------------------------------------------------------------------
SPRIDEN_MI
--------------------------------------------------------------------------------
ID_T SPRIDEN_ID
---- ------------------------------------
      714443
Tricker
```

Generate TKPROF Report from the background trace directory



```
[oracle@bandbdev04 trace]$ tkprof  ePPRD_ora_13076.trc  ePPRD_ora_13076.tkrpof

TKPROF: Release 11.2.0.3.0 - Development on Fri Apr 29 15:15:57 2016

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.
```

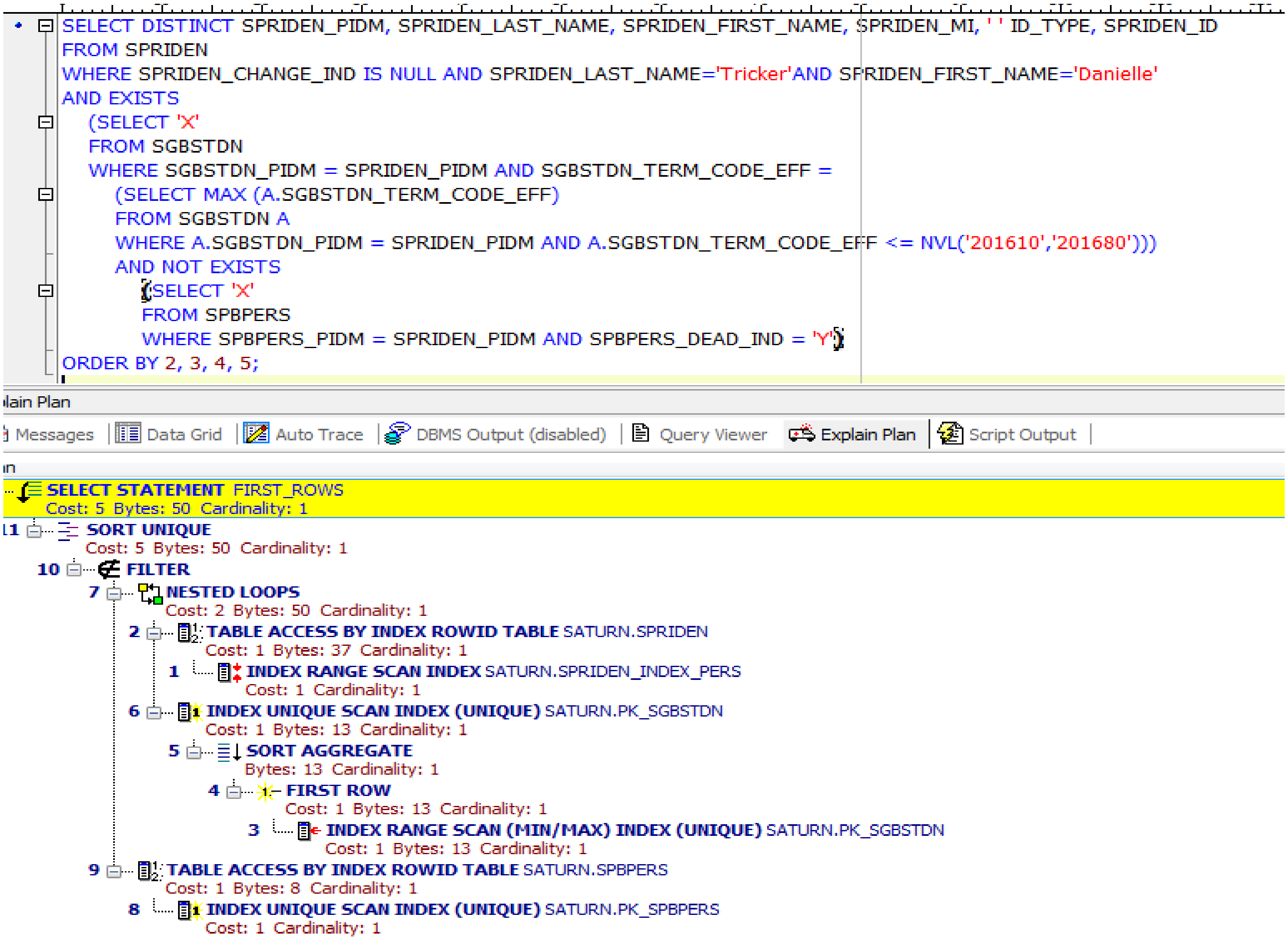
Full TKPROF



```
SELECT DISTINCT SPRIDEN_PIDM, SPRIDEN_LAST_NAME, SPRIDEN_FIRST_NAME, SPRIDEN_MI, ' ' ID_TYPE, SPRIDEN_ID
FROM SPRIDEN
WHERE SPRIDEN_CHANGE_IND IS NULL AND SPRIDEN_LAST_NAME='Tricker'AND SPRIDEN_FIRST_NAME='Danielle'
AND EXISTS
        (SELECT 'X'
        FROM SGBSTDN
        WHERE SGBSTDN_PIDM = SPRIDEN_PIDM AND SGBSTDN_TERM_CODE_EFF =
                (SELECT MAX (A.SGBSTDN_TERM_CODE_EFF)
                FROM SGBSTDN A
                WHERE A.SGBSTDN_PIDM = SPRIDEN_PIDM AND A.SGBSTDN_TERM_CODE_EFF <= NVL('201610','201680')))
                AND NOT EXISTS
                        (SELECT 'X'
                        FROM SPBPERS
                        WHERE SPBPERS_PIDM = SPRIDEN_PIDM AND SPBPERS_DEAD_IND = 'Y')
ORDER BY 2, 3, 4, 5

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          0         14          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.00          0         14          0           1

Misses in library cache during parse: 1
Optimizer mode: FIRST_ROWS
Parsing user id: SYS
Number of plan statistics captured: 1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT UNIQUE (cr=14 pr=0 pw=0 time=149 us cost=5 size=50 card=1)
         1          1          1   FILTER  (cr=14 pr=0 pw=0 time=128 us)
         1          1          1    NESTED LOOPS  (cr=10 pr=0 pw=0 time=102 us cost=2 size=50 card=1)
         1          1          1     TABLE ACCESS BY INDEX ROWID SPRIDEN (cr=4 pr=0 pw=0 time=43 us cost=1 size=37 card=1)
         1          1          1      INDEX RANGE SCAN SPRIDEN_INDEX_PERS (cr=3 pr=0 pw=0 time=31 us cost=1 size=0 card=1)(object id 105063)
         1          1          1     INDEX UNIQUE SCAN PK_SGBSTDN (cr=6 pr=0 pw=0 time=51 us cost=1 size=13 card=1)(object id 105544)
         1          1          1      SORT AGGREGATE (cr=3 pr=0 pw=0 time=24 us)
         1          1          1       FIRST ROW  (cr=3 pr=0 pw=0 time=17 us cost=1 size=13 card=1)
         1          1          1        INDEX RANGE SCAN (MIN/MAX) PK_SGBSTDN (cr=3 pr=0 pw=0 time=17 us cost=1 size=13 card=1)(object id 105544)
         0          0          0    TABLE ACCESS BY INDEX ROWID SPBPERS (cr=4 pr=0 pw=0 time=17 us cost=1 size=8 card=1)
         1          1          1     INDEX UNIQUE SCAN PK_SPBPERS (cr=3 pr=0 pw=0 time=11 us cost=1 size=0 card=1)(object id 105053)
```

```
***********************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
***********************************************************************
```

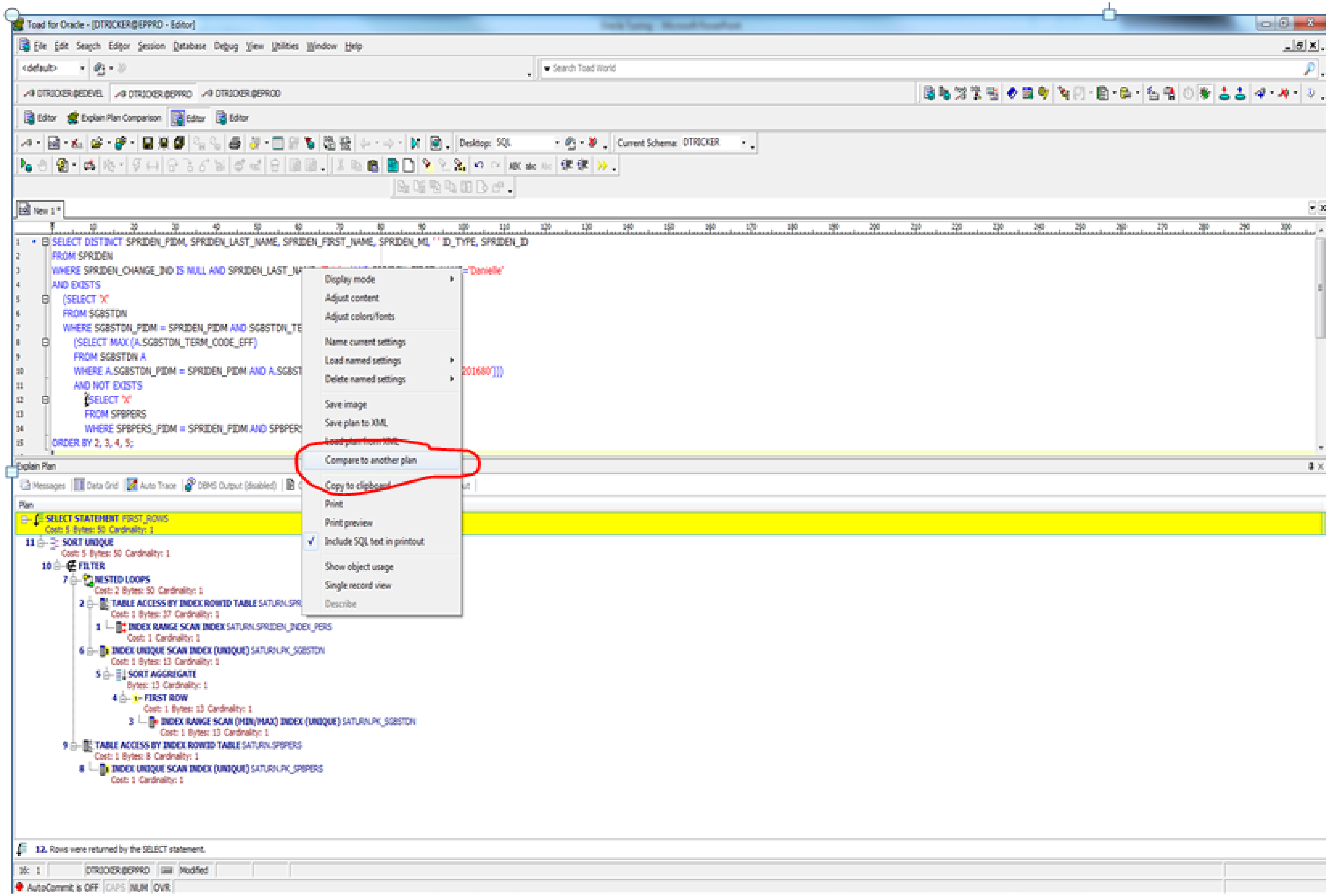
Toad Explain Plan for TOAD users:



```sql
SELECT DISTINCT SPRIDEN_PIDM, SPRIDEN_LAST_NAME, SPRIDEN_FIRST_NAME, SPRIDEN_MI, ' ' ID_TYPE, SPRIDEN_ID
FROM SPRIDEN
WHERE SPRIDEN_CHANGE_IND IS NULL AND SPRIDEN_LAST_NAME='Tricker'AND SPRIDEN_FIRST_NAME='Danielle'
AND EXISTS
   (SELECT 'X'
   FROM SGBSTDN
   WHERE SGBSTDN_PIDM = SPRIDEN_PIDM AND SGBSTDN_TERM_CODE_EFF =
      (SELECT MAX (A.SGBSTDN_TERM_CODE_EFF)
      FROM SGBSTDN A
      WHERE A.SGBSTDN_PIDM = SPRIDEN_PIDM AND A.SGBSTDN_TERM_CODE_EFF <= NVL('201610','201680'))
      AND NOT EXISTS
         (SELECT 'X'
         FROM SPBPERS
         WHERE SPBPERS_PIDM = SPRIDEN_PIDM AND SPBPERS_DEAD_IND = 'Y'))
ORDER BY 2, 3, 4, 5;
```

```
SELECT STATEMENT  FIRST_ROWS
Cost: 5  Bytes: 50  Cardinality: 1
11  SORT UNIQUE
      Cost: 5  Bytes: 50  Cardinality: 1
   10  FILTER
      7  NESTED LOOPS
            Cost: 2  Bytes: 50  Cardinality: 1
         2  TABLE ACCESS BY INDEX ROWID TABLE SATURN.SPRIDEN
               Cost: 1  Bytes: 37  Cardinality: 1
            1  INDEX RANGE SCAN INDEX SATURN.SPRIDEN_INDEX_PERS
                  Cost: 1  Cardinality: 1
         6  INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SGBSTDN
               Cost: 1  Bytes: 13  Cardinality: 1
            5  SORT AGGREGATE
                  Bytes: 13  Cardinality: 1
               4  FIRST ROW
                     Cost: 1  Bytes: 13  Cardinality: 1
                  3  INDEX RANGE SCAN (MIN/MAX) INDEX (UNIQUE) SATURN.PK_SGBSTDN
                        Cost: 1  Bytes: 13  Cardinality: 1
      9  TABLE ACCESS BY INDEX ROWID TABLE SATURN.SPBPERS
            Cost: 1  Bytes: 8  Cardinality: 1
         8  INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SPBPERS
               Cost: 1  Cardinality: 1
```

Right click and select Compose to another plan



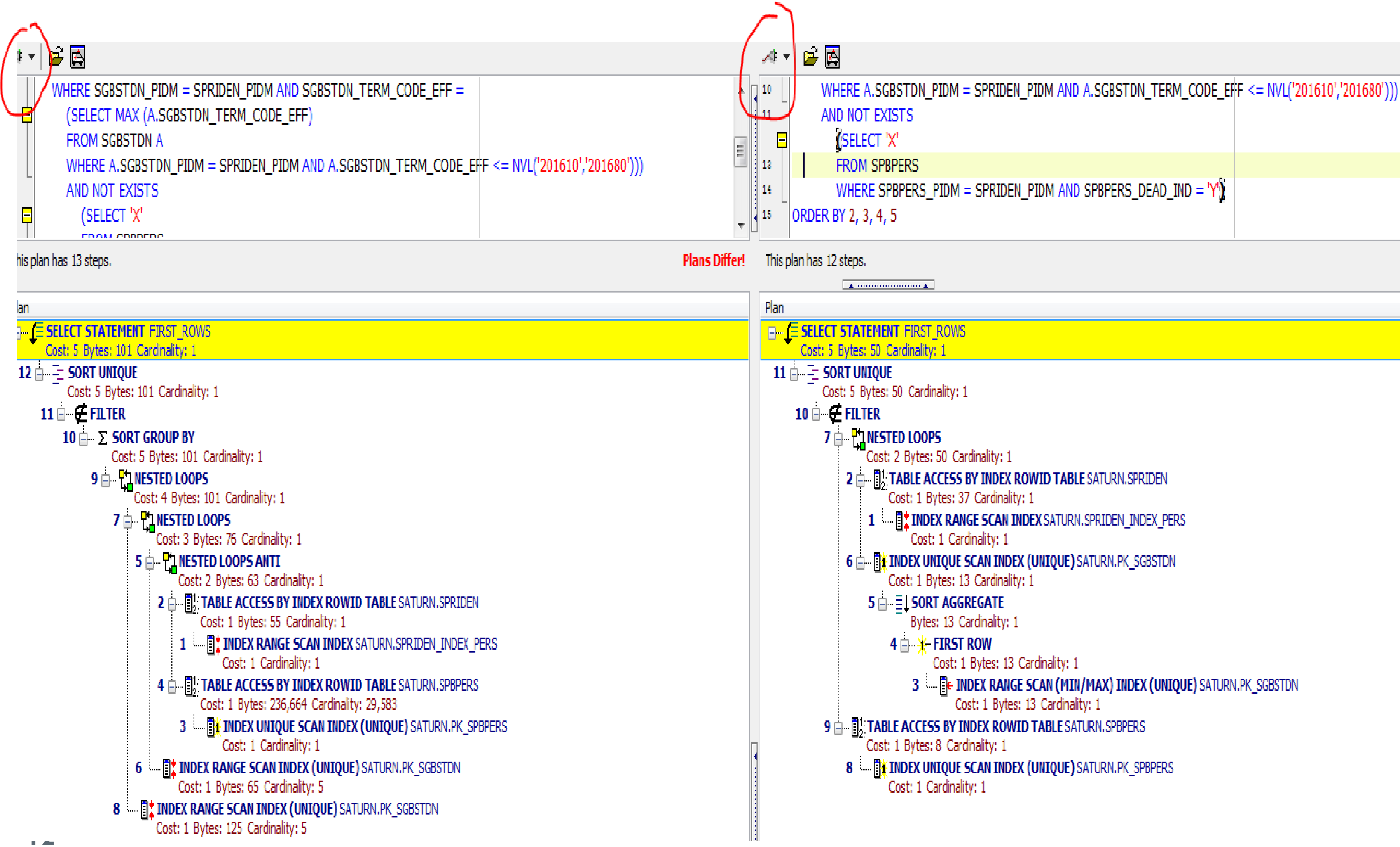
```
SELECT DISTINCT SPRIDEN_PIDM, SPRIDEN_LAST_NAME, SPRIDEN_FIRST_NAME, SPRIDEN_MI, ' ' ID_TYPE, SPRIDEN_ID
FROM SPRIDEN
WHERE SPRIDEN_CHANGE_IND IS NULL AND SPRIDEN_LAST_NA                           ='Danielle'
AND EXISTS
   (SELECT 'X'
   FROM SGBSTDN
   WHERE SGBSTDN_PIDM = SPRIDEN_PIDM AND SGBSTDN_TE
      (SELECT MAX (A.SGBSTDN_TERM_CODE_EFF)
      FROM SGBSTDN A
      WHERE A.SGBSTDN_PIDM = SPRIDEN_PIDM AND A.SGBST                          201680')))
      AND NOT EXISTS
         (SELECT 'X'
         FROM SPBPERS
         WHERE SPBPERS_PIDM = SPRIDEN_PIDM AND SPBPERS
ORDER BY 2, 3, 4, 5;
```

```
SELECT STATEMENT FIRST_ROWS
Cost: 5 Bytes: 50 Cardinality: 1
11 SORT UNIQUE
   Cost: 5 Bytes: 50 Cardinality: 1
   10 FILTER
      7 NESTED LOOPS
        Cost: 2 Bytes: 50 Cardinality: 1
         2 TABLE ACCESS BY INDEX ROWID TABLE SATURN.SPR
           Cost: 1 Bytes: 37 Cardinality: 1
            1 INDEX RANGE SCAN INDEX SATURN.SPRIDEN_INDEX_PERS
              Cost: 1 Cardinality: 1
         6 INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SGBSTDN
           Cost: 1 Bytes: 13 Cardinality: 1
            5 SORT AGGREGATE
              Bytes: 13 Cardinality: 1
               4 FIRST ROW
                 Cost: 1 Bytes: 13 Cardinality: 1
                  3 INDEX RANGE SCAN (MIN/MAX) INDEX (UNIQUE) SATURN.PK_SGBSTDN
                    Cost: 1 Bytes: 13 Cardinality: 1
      9 TABLE ACCESS BY INDEX ROWID TABLE SATURN.SPBPERS
        Cost: 1 Bytes: 8 Cardinality: 1
         8 INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SPBPERS
           Cost: 1 Cardinality: 1
```

12 Rows were returned by the SELECT statement.

16: 1

Use the drop down buttons in red to select different databases to compare



```
WHERE SGBSTDN_PIDM = SPRIDEN_PIDM AND SGBSTDN_TERM_CODE_EFF =
  (SELECT MAX (A.SGBSTDN_TERM_CODE_EFF)
  FROM SGBSTDN A
  WHERE A.SGBSTDN_PIDM = SPRIDEN_PIDM AND A.SGBSTDN_TERM_CODE_EFF <= NVL('201610','201680')))
  AND NOT EXISTS
    (SELECT 'X'
    FROM SPBPERS
```

his plan has 13 steps.

Plans Differ!

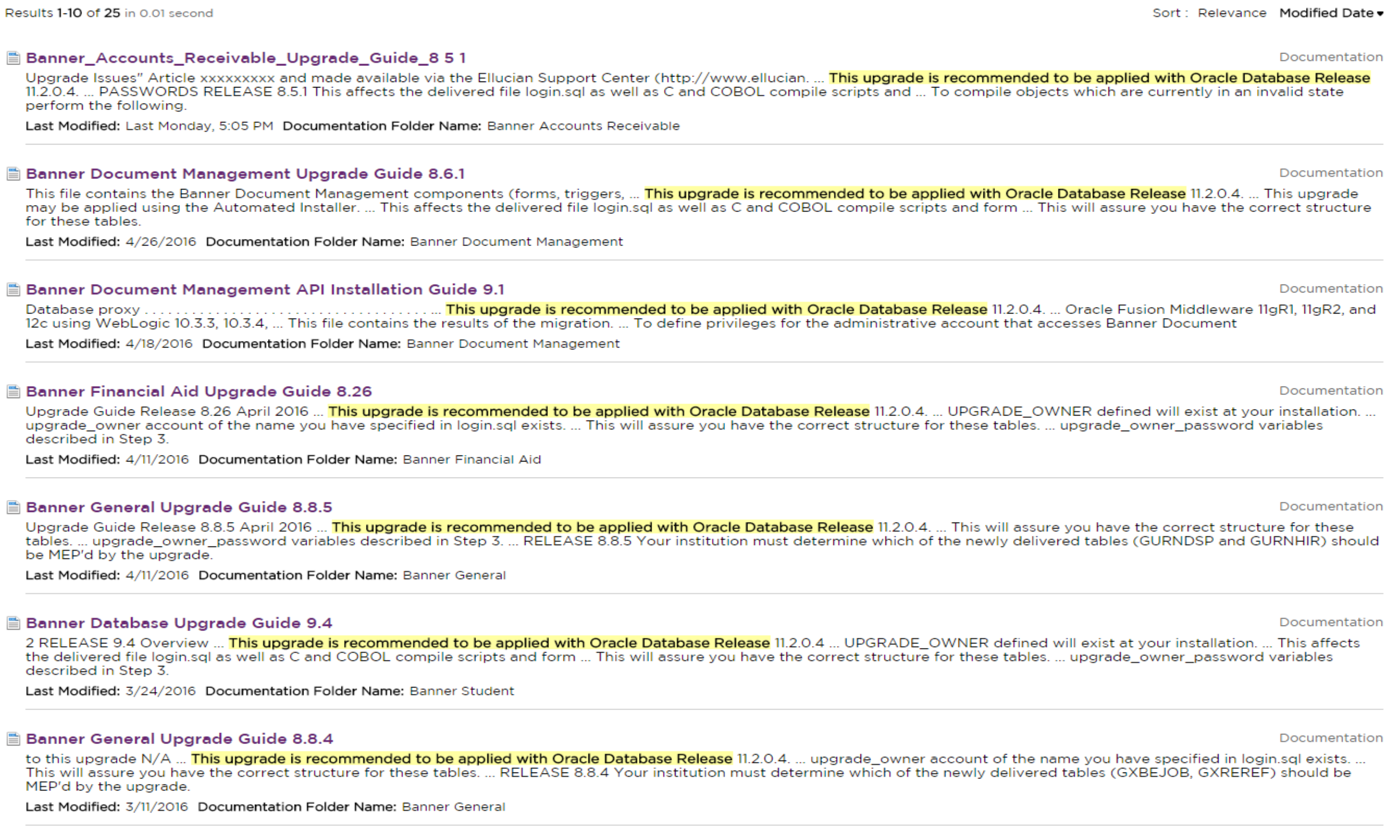
```
lan
SELECT STATEMENT FIRST_ROWS
Cost: 5 Bytes: 101 Cardinality: 1
12 SORT UNIQUE
    Cost: 5 Bytes: 101 Cardinality: 1
  11 FILTER
    10 SORT GROUP BY
        Cost: 5 Bytes: 101 Cardinality: 1
      9 NESTED LOOPS
          Cost: 4 Bytes: 101 Cardinality: 1
        7 NESTED LOOPS
            Cost: 3 Bytes: 76 Cardinality: 1
          5 NESTED LOOPS ANTI
              Cost: 2 Bytes: 63 Cardinality: 1
            2 TABLE ACCESS BY INDEX ROWID TABLE SATURN.SPRIDEN
                Cost: 1 Bytes: 55 Cardinality: 1
              1 INDEX RANGE SCAN INDEX SATURN.SPRIDEN_INDEX_PERS
                  Cost: 1 Cardinality: 1
            4 TABLE ACCESS BY INDEX ROWID TABLE SATURN.SPBPERS
                Cost: 1 Bytes: 236,664 Cardinality: 29,583
              3 INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SPBPERS
                  Cost: 1 Cardinality: 1
          6 INDEX RANGE SCAN INDEX (UNIQUE) SATURN.PK_SGBSTDN
              Cost: 1 Bytes: 65 Cardinality: 5
        8 INDEX RANGE SCAN INDEX (UNIQUE) SATURN.PK_SGBSTDN
            Cost: 1 Bytes: 125 Cardinality: 5
```

```
WHERE A.SGBSTDN_PIDM = SPRIDEN_PIDM AND A.SGBSTDN_TERM_CODE_EFF <= NVL('201610','201680')))
AND NOT EXISTS
  (SELECT 'X'
  FROM SPBPERS
  WHERE SPBPERS_PIDM = SPRIDEN_PIDM AND SPBPERS_DEAD_IND = 'Y')
ORDER BY 2, 3, 4, 5
```

This plan has 12 steps.

```
Plan
SELECT STATEMENT FIRST_ROWS
Cost: 5 Bytes: 50 Cardinality: 1
11 SORT UNIQUE
    Cost: 5 Bytes: 50 Cardinality: 1
  10 FILTER
    7 NESTED LOOPS
        Cost: 2 Bytes: 50 Cardinality: 1
      2 TABLE ACCESS BY INDEX ROWID TABLE SATURN.SPRIDEN
          Cost: 1 Bytes: 37 Cardinality: 1
        1 INDEX RANGE SCAN INDEX SATURN.SPRIDEN_INDEX_PERS
            Cost: 1 Cardinality: 1
      6 INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SGBSTDN
          Cost: 1 Bytes: 13 Cardinality: 1
        5 SORT AGGREGATE
            Bytes: 13 Cardinality: 1
          4 FIRST ROW
              Cost: 1 Bytes: 13 Cardinality: 1
            3 INDEX RANGE SCAN (MIN/MAX) INDEX (UNIQUE) SATURN.PK_SGBSTDN
                Cost: 1 Bytes: 13 Cardinality: 1
    9 TABLE ACCESS BY INDEX ROWID TABLE SATURN.SPBPERS
        Cost: 1 Bytes: 8 Cardinality: 1
      8 INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SPBPERS
          Cost: 1 Cardinality: 1
```

3. When/Why Upgrade to 12c - Search on: "This upgrade is recommended to be applied with Oracle Database Release"



Results 1-10 of 25 in 0.01 second

Sort : Relevance **Modified Date ▾**

Banner_Accounts_Receivable_Upgrade_Guide_8 5 1 — Documentation

Upgrade Issues" Article xxxxxxxxx and made available via the Ellucian Support Center (http://www.ellucian. ... This upgrade is recommended to be applied with Oracle Database Release 11.2.0.4. ... PASSWORDS RELEASE 8.5.1 This affects the delivered file login.sql as well as C and COBOL compile scripts and ... To compile objects which are currently in an invalid state perform the following.

Last Modified: Last Monday, 5:05 PM **Documentation Folder Name:** Banner Accounts Receivable

Banner Document Management Upgrade Guide 8.6.1 — Documentation

This file contains the Banner Document Management components (forms, triggers, ... This upgrade is recommended to be applied with Oracle Database Release 11.2.0.4. ... This upgrade may be applied using the Automated Installer. ... This affects the delivered file login.sql as well as C and COBOL compile scripts and form ... This will assure you have the correct structure for these tables.

Last Modified: 4/26/2016 **Documentation Folder Name:** Banner Document Management

Banner Document Management API Installation Guide 9.1 — Documentation

Database proxy . . . . This upgrade is recommended to be applied with Oracle Database Release 11.2.0.4. ... Oracle Fusion Middleware 11gR1, 11gR2, and 12c using WebLogic 10.3.3, 10.3.4, ... This file contains the results of the migration. ... To define privileges for the administrative account that accesses Banner Document

Last Modified: 4/18/2016 **Documentation Folder Name:** Banner Document Management

Banner Financial Aid Upgrade Guide 8.26 — Documentation

Upgrade Guide Release 8.26 April 2016 ... This upgrade is recommended to be applied with Oracle Database Release 11.2.0.4. ... UPGRADE_OWNER defined will exist at your installation. ... upgrade_owner account of the name you have specified in login.sql exists. ... This will assure you have the correct structure for these tables. ... upgrade_owner_password variables described in Step 3.

Last Modified: 4/11/2016 **Documentation Folder Name:** Banner Financial Aid

Banner General Upgrade Guide 8.8.5 — Documentation

Upgrade Guide Release 8.8.5 April 2016 ... This upgrade is recommended to be applied with Oracle Database Release 11.2.0.4. ... This will assure you have the correct structure for these tables. ... upgrade_owner_password variables described in Step 3. ... RELEASE 8.8.5 Your institution must determine which of the newly delivered tables (GURNDSP and GURNHIR) should be MEP'd by the upgrade.

Last Modified: 4/11/2016 **Documentation Folder Name:** Banner General

Banner Database Upgrade Guide 9.4 — Documentation

2 RELEASE 9.4 Overview ... This upgrade is recommended to be applied with Oracle Database Release 11.2.0.4 ... UPGRADE_OWNER defined will exist at your installation. ... This affects the delivered file login.sql as well as C and COBOL compile scripts and form ... This will assure you have the correct structure for these tables. ... upgrade_owner_password variables described in Step 3.

Last Modified: 3/24/2016 **Documentation Folder Name:** Banner Student

Banner General Upgrade Guide 8.8.4 — Documentation

to this upgrade N/A ... This upgrade is recommended to be applied with Oracle Database Release 11.2.0.4. ... upgrade_owner account of the name you have specified in login.sql exists. ... This will assure you have the correct structure for these tables. ... RELEASE 8.8.4 Your institution must determine which of the newly delivered tables (GXBEJOB, GXREREF) should be MEP'd by the upgrade.

Last Modified: 3/11/2016 **Documentation Folder Name:** Banner General

*****From Ellucian: Technology Planning and Readiness Top 5 - All of the Banner XE administrative modules are scheduled for final delivery in December 2016. A transition to the Banner XE modules fully removes the dependency on any version of Oracle Forms and Reports.**

4. 12c Optimizer Overview

It is new and complex, and there are many questions such as boracle post below.

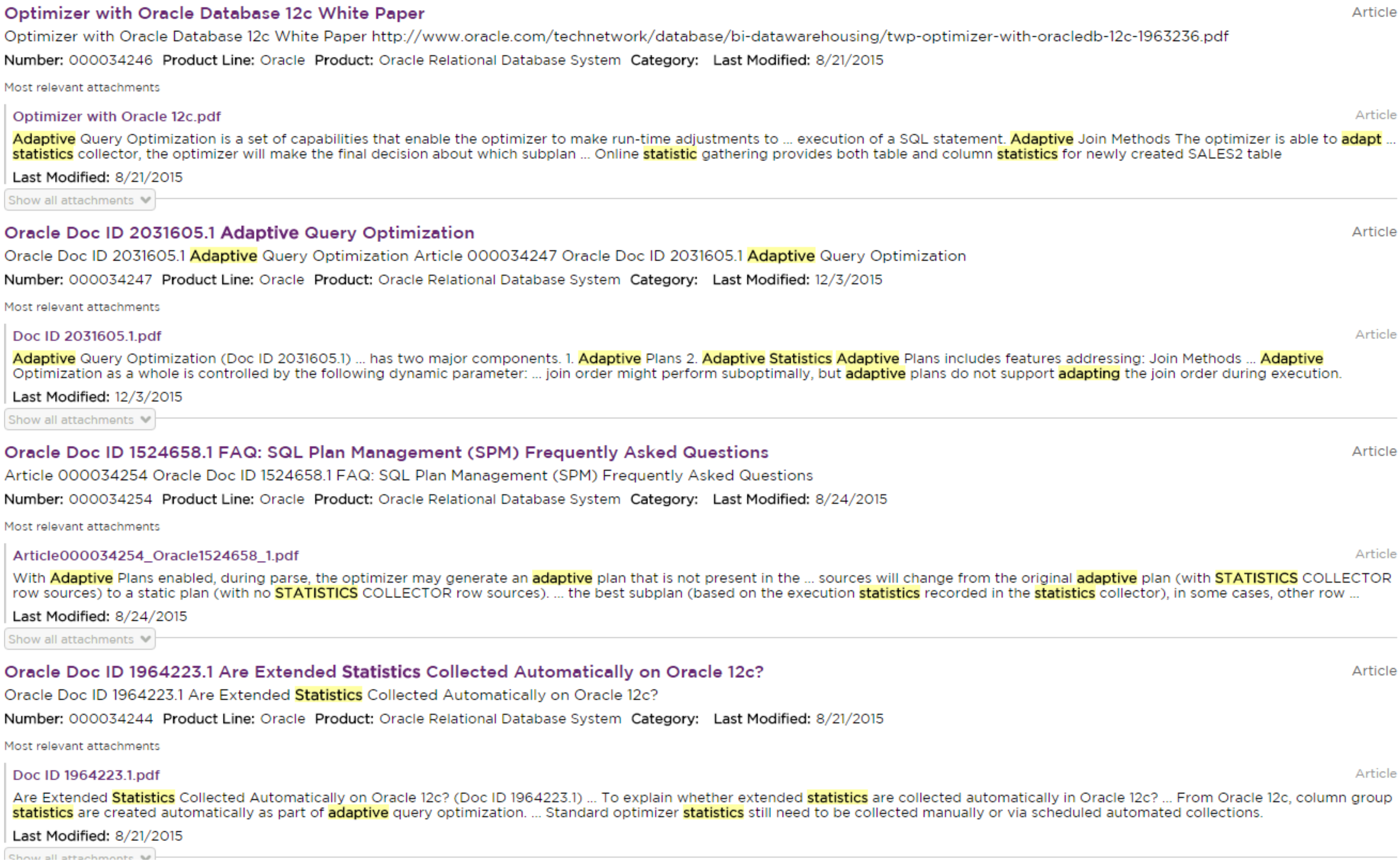
I'm interested in hearing from sites that have tried using adaptive query optimization with Banner and/or ODS 12c databases. Is it working well for you, did you try it and then turn it back off, or did you do something else? If you have been successful using it, did you have to do anything special to make it work well?

I ask because one member of our DBA team has heard all sorts of wonderful things about the 12c adaptive query optimizer. I, however, have heard the opposite, that it causes nothing but grief. But I don't
want to discount it as an option if making it work is just a matter of proper configuration.

On a slightly different topic, I ran into this Iggy Fernandez article about what you should do if you enable adaptive query optimization, then change your mind and disable it. You may need to do some additional cleanup beyond setting "optimizer_adaptive_features=false".

http://www.toadworld.com/platforms/oracle/w/wiki/11586.completely-disabling-adaptive-query-optimization-in-oracle-database-12c
Stephany Freeman
University of Oregon

Ellucian articles on Adaptive Optimization:



Optimizer with Oracle Database 12c White Paper — Article

Optimizer with Oracle Database 12c White Paper http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-optimizer-with-oracledb-12c-1963236.pdf

Number: 000034246 **Product Line:** Oracle **Product:** Oracle Relational Database System **Category:** **Last Modified:** 8/21/2015

Most relevant attachments

Optimizer with Oracle 12c.pdf — Article

Adaptive Query Optimization is a set of capabilities that enable the optimizer to make run-time adjustments to ... execution of a SQL statement. Adaptive Join Methods The optimizer is able to adapt ... statistics collector, the optimizer will make the final decision about which subplan ... Online statistic gathering provides both table and column statistics for newly created SALES2 table

Last Modified: 8/21/2015

Oracle Doc ID 2031605.1 Adaptive Query Optimization — Article

Oracle Doc ID 2031605.1 Adaptive Query Optimization Article 000034247 Oracle Doc ID 2031605.1 Adaptive Query Optimization

Number: 000034247 **Product Line:** Oracle **Product:** Oracle Relational Database System **Category:** **Last Modified:** 12/3/2015

Most relevant attachments

Doc ID 2031605.1.pdf — Article

Adaptive Query Optimization (Doc ID 2031605.1) ... has two major components. 1. Adaptive Plans 2. Adaptive Statistics Adaptive Plans includes features addressing: Join Methods ... Adaptive Optimization as a whole is controlled by the following dynamic parameter: ... join order might perform suboptimally, but adaptive plans do not support adapting the join order during execution.

Last Modified: 12/3/2015

Oracle Doc ID 1524658.1 FAQ: SQL Plan Management (SPM) Frequently Asked Questions — Article

Article 000034254 Oracle Doc ID 1524658.1 FAQ: SQL Plan Management (SPM) Frequently Asked Questions

Number: 000034254 **Product Line:** Oracle **Product:** Oracle Relational Database System **Category:** **Last Modified:** 8/24/2015

Most relevant attachments

Article000034254_Oracle1524658_1.pdf — Article

With Adaptive Plans enabled, during parse, the optimizer may generate an adaptive plan that is not present in the ... sources will change from the original adaptive plan (with STATISTICS COLLECTOR row sources) to a static plan (with no STATISTICS COLLECTOR row sources). ... the best subplan (based on the execution statistics recorded in the statistics collector), in some cases, other row ...

Last Modified: 8/24/2015

Oracle Doc ID 1964223.1 Are Extended Statistics Collected Automatically on Oracle 12c? — Article

Oracle Doc ID 1964223.1 Are Extended Statistics Collected Automatically on Oracle 12c?

Number: 000034244 **Product Line:** Oracle **Product:** Oracle Relational Database System **Category:** **Last Modified:** 8/21/2015

Most relevant attachments

Doc ID 1964223.1.pdf — Article

Are Extended Statistics Collected Automatically on Oracle 12c? (Doc ID 1964223.1) ... To explain whether extended statistics are collected automatically in Oracle 12c? ... From Oracle 12c, column group statistics are created automatically as part of adaptive query optimization. ... Standard optimizer statistics still need to be collected manually or via scheduled automated collections.

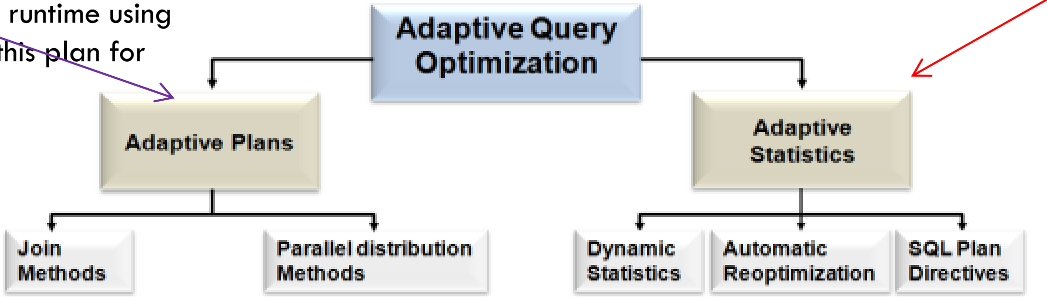
Last Modified: 8/21/2015

12c Optimizer Overview

12c Adaptive Query Optimization/Adaptive Execution Plans

- The optimizer makes **runtime** adjustments to explain plans to find better execution plans. This helps when statistics are not sufficient or complex predicates are used. **In Oracle releases prior to 12, once an execution plan was determined there was no possible deviation from it at runtime.**
- There are two parts: 1. Adaptive Plans
  2. Adaptive Statistics

1. **Adaptive Plans** make up for misestimates in cardinality. The optimizer adapts plans at runtime using actual statistics and then uses this plan for all subsequent executions. These plans are used for **Join Methods** and **Parallel Dist methods.**



Adaptive Query Optimization

Adaptive Plans

Adaptive Statistics

Join Methods

Parallel distribution Methods

Dynamic Statistics

Automatic Reoptimization

SQL Plan Directives

2. **Adaptive Statistics** are used when tables statistics behind a complex query are not good enough to generate a good plan. Adaptive statistics comprise **Dynamic Statistics, Automatic Re-optimization, and SQL Plan Directives**

Picture from: http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-optimizer-with-oracledb-12c-1963236.pdf

1. Adaptive Plans -

- Adaptive plans wait right until the execution time of a query to make the final plan by comparing cardinality estimates to the actual row counts.

- Adaptive plans will make adjustments on the fly to avoid poor performance on the first execution. The two type of plan adjustments that are made on the fly are **1. Join Methods**
  2. Parallel Distribution Methods.

- Adaptive Plans are enabled right out of the box in 12c. You can however, turn it off, by making run the reporting-only mode. It will collect information, but will not change execution plans. To do this, set OPTIMIZER_ADAPTIVE_REPORTING_ONLY=TRUE

  1. Join Methods - the optimizer may decide to do a nested loop instead of hash join on the fly or vise versa. To permanently disable Adaptive plans, including both Join Methods and Parallel Distribution Methods, set the hidden parameter **`_optimizer_adaptive_plans = false;`**

  2. Parallel Distribution Methods are very useful for parallel execution. The parallel distribution method is determined by the number of records to be returned along the number of parallel processes.

- If only certain queries and views are experiencing performance problems after upgrading, you can try using hint – **`/*+ NO_ADAPTIVE_PLAN */`** which disables the use of adaptive plans for a particular SQL statement

- To disable only one of these methods, use hidden parameters:
  `_optimizer_nlj_hj_adatpive_join = false;` --disables only the adaptive join from nested loops to hash join
  `_px_adaptive_dist_method = off;` --disables the adaptive parallel distribution methods

However, if the initial join method is a sort merge join no adaptation will take place.

2. ADAPTIVE STATISTICS -

- **Adaptive Statistics** are used when complex predicate statistics are not sufficient to generate good execution plans.

- There are three types of adaptive statistics – **1. Dynamic Statistics**
  2. Automatic Re-optimization
  3. SQL Plan Directives.

1. Dynamic Statistics was called dynamic sampling in 11g. 12c Dynamic Statistics augment normal statistics. They help the optimizer improve plans so it can better estimate predicate selectivity. The optimizer looks at available stats (called default stats) and then determines if the default stats need to be improved with dynamic statistics. 12c retains the results of dynamic stats and then re-uses them in subsequent queries.

- **Dynamic Statistics** in 12c are now performed on statements doing joins, group by, and non-parallel statements, unlike 11g and lower.

- **Dynamic Statistics** in 12c are NOT enabled by default. You must set **OPTIMIZER_DYNAMIC_SAMPLING=11** from the default of 2. The default setting of 2 means the optimizer will gather dynamic stats if at least one table in a sql statement is missing stats. When set to 11, the optimizer will use dynamic sampling when ever it determines it is necessary, because of missing stats, stale stats or insufficient stats(i.e.. data skew, missing extended stats, complex predicates).

- **Dynamic Statistics** can be turned off by setting **OPTIMIZER_DYNAMIC_SAMPLING=0.**

  -per OCP Upgrade to Oracle Database 12c Exam Guide
  This is the recommendation from Oracle for unrepeated OLTP queries, since there is overhead gathering stats on the fly.

2. ADAPTIVE STATISTICS - Continued

2. Automatic Re-optimization – During execution, the optimizer compares statistics to execution statistics. If there is a difference, the optimizer may modify execution plans for the next execution. It will continually optimize queries, learning more and improving the plan.

- **Automatic Re-optimization** uses two modes for optimization - **1. Statistics Feedback**
  2. Performance Feedback

  1. Statistics Feedback – also called Cardinality feedback's goal is to improve the execution plans for frequently executing queries with cardinality misestimates**.** **It** enables monitoring when tables have no statistics, multiple filter predicates on a table, and predicates containing complex operators. The optimizer then compares the stats improving the plans. After the 1st execution, the optimizer disable statistics feedback, and stores the information for future use and may create a sql plan directive.

  2. Performance Feedback – is for improving the degree of parallelism. After the 1st execution, the optimizer compares the DOP to the actual DOP used by query and makes adjustments for the next execution. Performance feedback is affected greatly by parameter **PARALLEL_DEGREE_POLICY**

3. SQL Plan Directives - the optimizer collects additional instructions during compilation or at the execution stage when it find missing stats or misestimated cardinalities. These directives are collected every 15 minutes, stored in the shared pool and then are written to the SYSAUX tablespace. If a plan is not used in 52 weeks, it is automatically purged, or you can flush them manually.

-

THE COMPLEXITY

- The definition of parameter OPTIMIZER_ADAPTIVE_FEATURES from Oracle 12c DB ADMIN BOOK: OPTIMIZER_ADAPTIVE_FEATURES - enables or disables all of the adaptive optimizer features, including adaptive plan (adaptive join methods and bitmap plans), automatic re-optimization, SQL plan directives, and adaptive distribution methods.

- The above definition says it disables all adaptive features, but does not mention dynamic statistics(dynamic sampling).

- DEFAULT SETTINGS: OPTIMIZER_ADAPTIVE_FEATURES = **TRUE** and OPTIMIZER_DYNAMIC_SAMPLING = 2

- When OPTIMIZER_ADAPTIVE_FEATURES = **FALSE** and OPTIMIZER_DYNAMIC_SAMPLING = 2 (default), then Automatic Dynamic Statistics (ADS) will not happen. However, the default dynamic sampling level is still used.

- If OPTIMIZER_DYNAMIC_SAMPLING is set to 11, then Automatic Dynamic Statistics is enabled regardless of the setting for parameter OPTIMIZER_ADAPTIVE_FEATURES. Also, possibly consider below recommendation for setting for this parameter.

  per OCP Upgrade to Oracle Database 12c Exam Guide
  This is the recommendation from Oracle for unrepeated OLTP queries, since there is overhead gathering stats on the fly.

Default Init Settings	**Value**
optimizer_adaptive_features	TRUE
optimizer_adaptive_reporting_only	FALSE
optimizer_features_enable	12.1.0.1
optimizer_dynamic_sampling	2

How to Verify Adaptive Optimization:

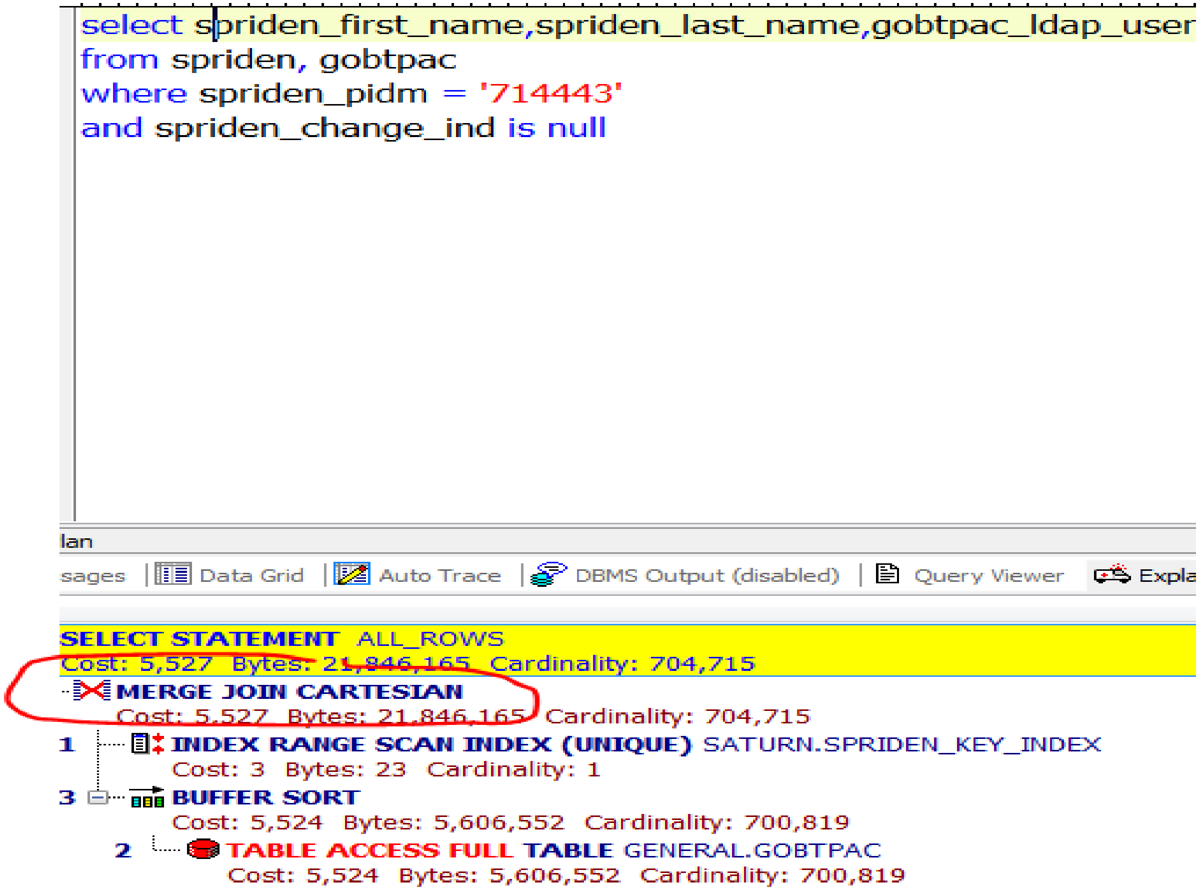
SELECT sql_id, child_number, sql_text,
IS_RESOLVED_ADAPTIVE_PLAN,
IS_REOPTIMIZABLE FROM v$sql;

IS_REOPTIMIZABLE is for next execution

- Y - the next execution will trigger a re-optimization
- R – has re-optimization info but won't trigger due to reporting mode
- N -the child cursor has no re-optimization info

5. Proper Table Joins using conventional/ANSI SQL

What is wrong with this query?



```
select spriden_first_name,spriden_last_name,gobtpac_ldap_user
from spriden, gobtpac
where spriden_pidm = '714443'
and spriden_change_ind is null
```

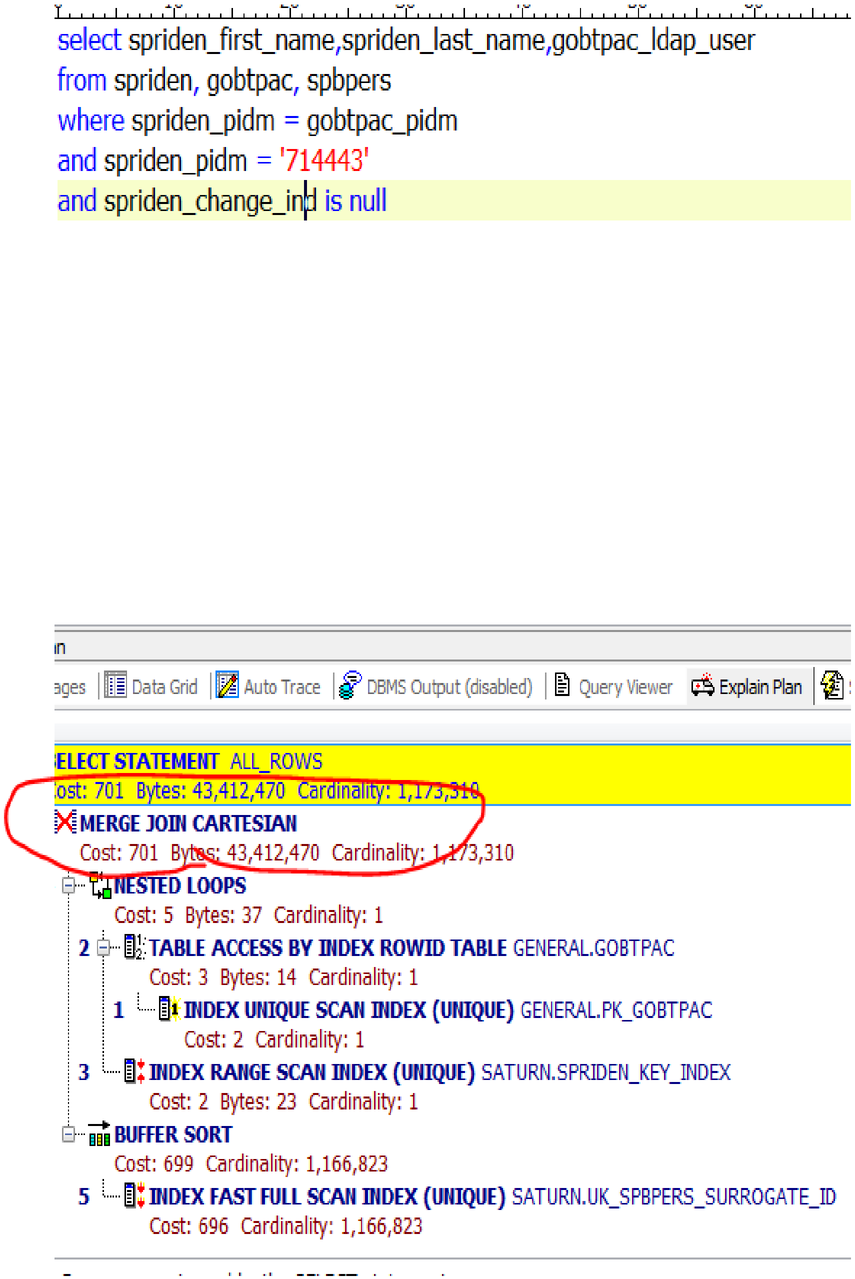
```
SELECT STATEMENT  ALL_ROWS
Cost: 5,527  Bytes: 21,846,165  Cardinality: 704,715
  MERGE JOIN CARTESIAN
    Cost: 5,527  Bytes: 21,846,165  Cardinality: 704,715
1     INDEX RANGE SCAN INDEX (UNIQUE) SATURN.SPRIDEN_KEY_INDEX
        Cost: 3  Bytes: 23  Cardinality: 1
3     BUFFER SORT
        Cost: 5,524  Bytes: 5,606,552  Cardinality: 700,819
  2     TABLE ACCESS FULL TABLE GENERAL.GOBTPAC
          Cost: 5,524  Bytes: 5,606,552  Cardinality: 700,819
```

Cartesian merge joins can also be caused by:

- Forgot to add a table join condition to WHERE clause
- Missing join indexes
- Bad/stale schema statistics (reanalyze with dbms_stats)

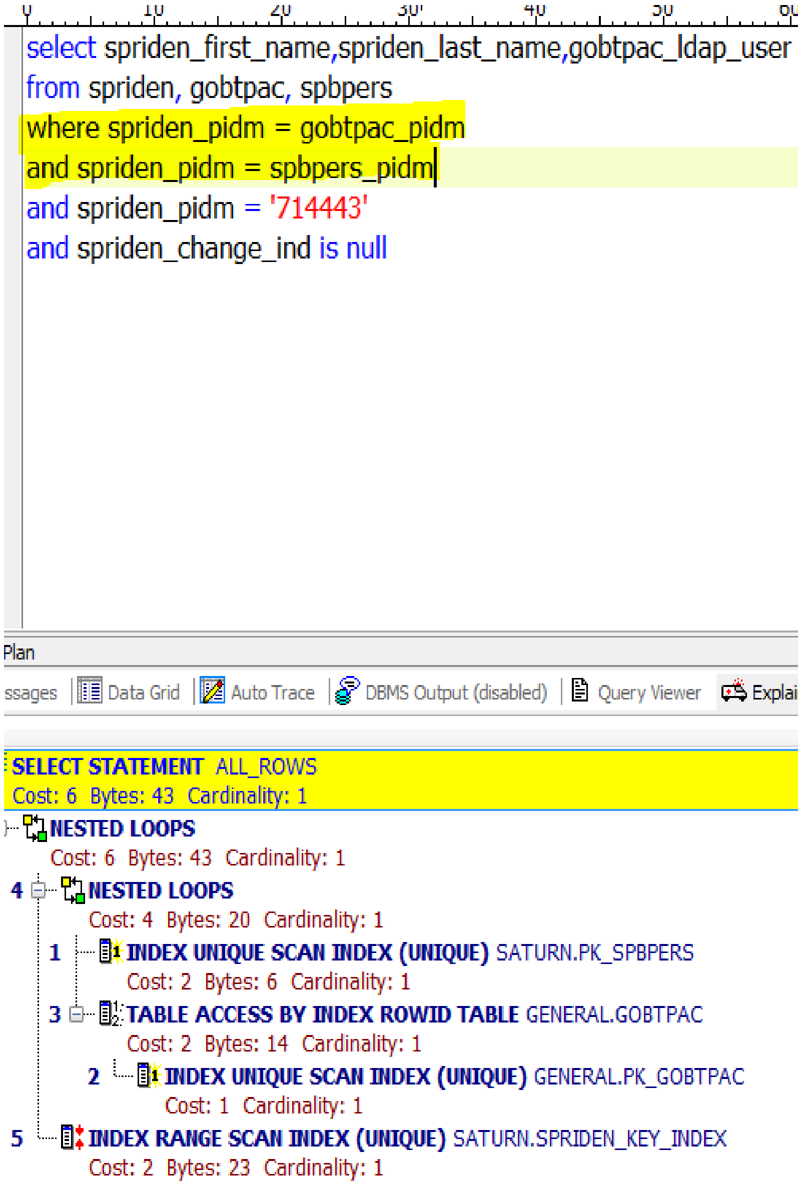
Missing joins – common mistake

Hint: The number of tables in a query needs the same number of joins minus 1



```
select spriden_first_name,spriden_last_name,gobtpac_ldap_user
from spriden, gobtpac, spbpers
where spriden_pidm = gobtpac_pidm
and spriden_pidm = '714443'
and spriden_change_ind is null
```

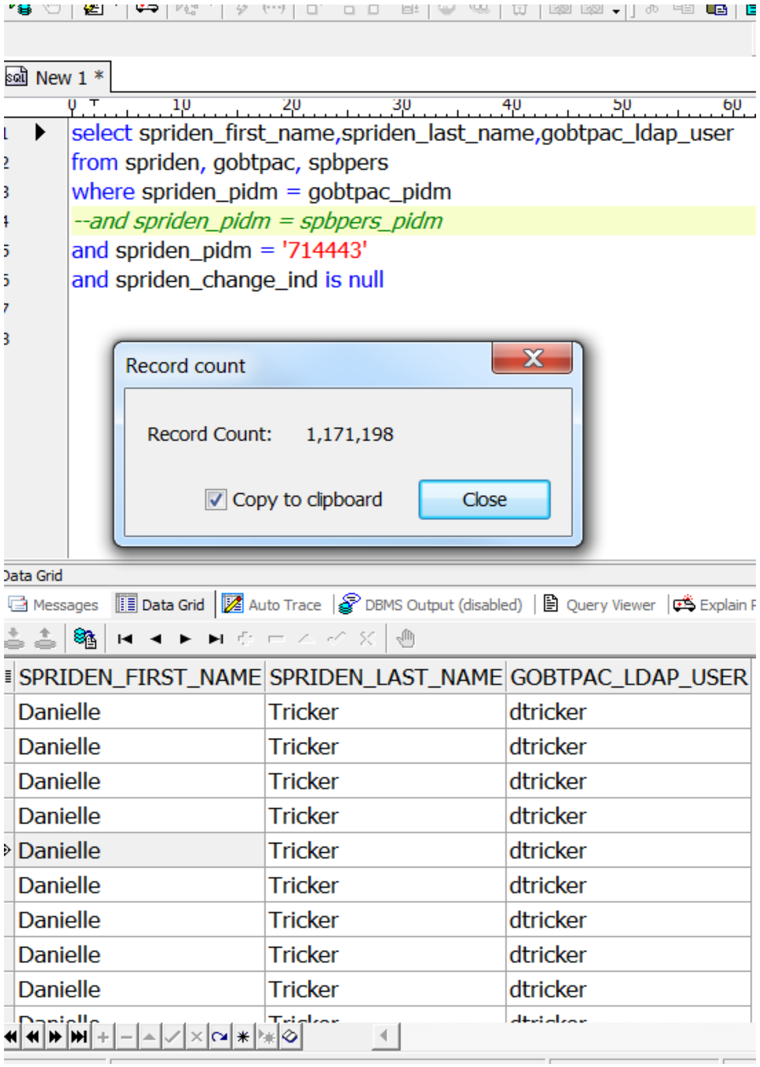
```
ELECT STATEMENT  ALL_ROWS
ost: 701  Bytes: 43,412,470  Cardinality: 1,173,310
MERGE JOIN CARTESIAN
   Cost: 701  Bytes: 43,412,470  Cardinality: 1,173,310
      NESTED LOOPS
         Cost: 5  Bytes: 37  Cardinality: 1
   2     TABLE ACCESS BY INDEX ROWID TABLE GENERAL.GOBTPAC
            Cost: 3  Bytes: 14  Cardinality: 1
      1     INDEX UNIQUE SCAN INDEX (UNIQUE) GENERAL.PK_GOBTPAC
               Cost: 2  Cardinality: 1
   3     INDEX RANGE SCAN INDEX (UNIQUE) SATURN.SPRIDEN_KEY_INDEX
            Cost: 2  Bytes: 23  Cardinality: 1
      BUFFER SORT
         Cost: 699  Cardinality: 1,166,823
   5     INDEX FAST FULL SCAN INDEX (UNIQUE) SATURN.UK_SPBPERS_SURROGATE_ID
            Cost: 696  Cardinality: 1,166,823
```



```
select spriden_first_name,spriden_last_name,gobtpac_ldap_user
from spriden, gobtpac, spbpers
where spriden_pidm = gobtpac_pidm
and spriden_pidm = spbpers_pidm
and spriden_pidm = '714443'
and spriden_change_ind is null
```

```
SELECT STATEMENT  ALL_ROWS
Cost: 6  Bytes: 43  Cardinality: 1
   NESTED LOOPS
      Cost: 6  Bytes: 43  Cardinality: 1
4     NESTED LOOPS
         Cost: 4  Bytes: 20  Cardinality: 1
   1     INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SPBPERS
            Cost: 2  Bytes: 6  Cardinality: 1
   3     TABLE ACCESS BY INDEX ROWID TABLE GENERAL.GOBTPAC
            Cost: 2  Bytes: 14  Cardinality: 1
      2     INDEX UNIQUE SCAN INDEX (UNIQUE) GENERAL.PK_GOBTPAC
               Cost: 1  Cardinality: 1
5     INDEX RANGE SCAN INDEX (UNIQUE) SATURN.SPRIDEN_KEY_INDEX
         Cost: 2  Bytes: 23  Cardinality: 1
```

Cartesian joins are expensive and can give you the wrong answer.



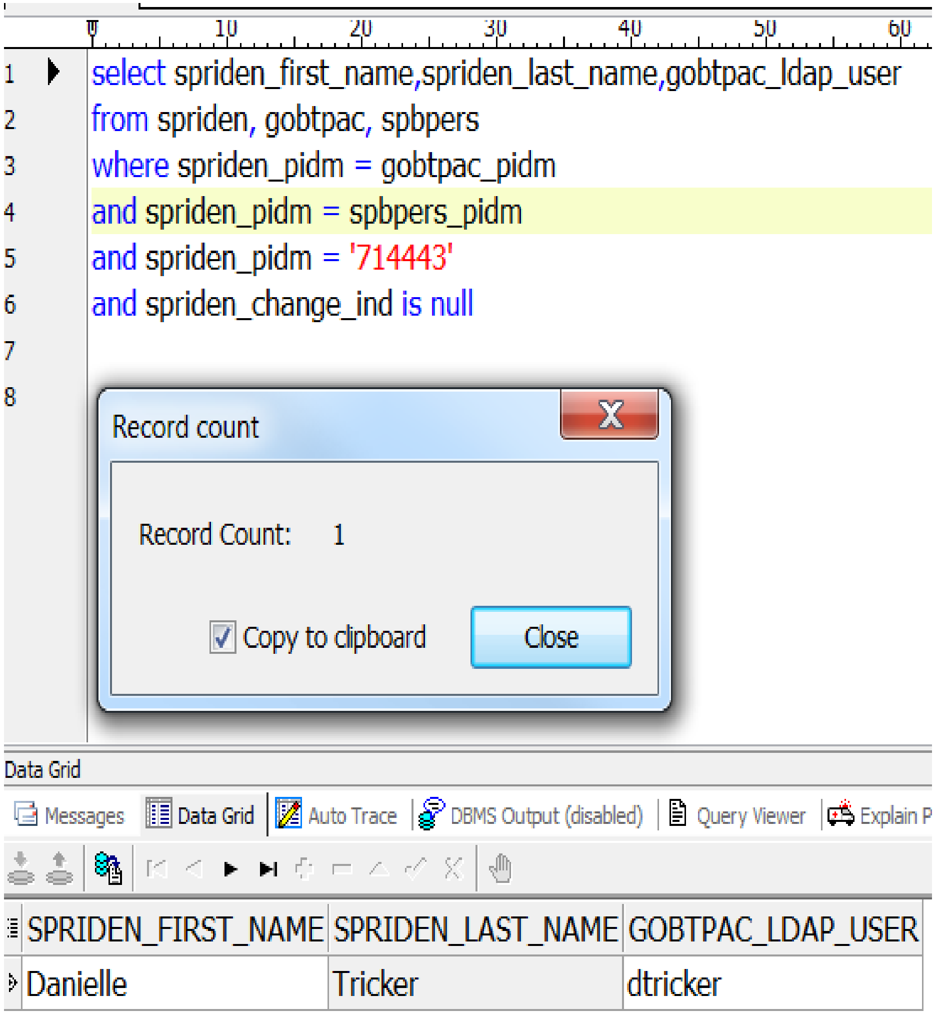
```
select spriden_first_name,spriden_last_name,gobtpac_ldap_user
from spriden, gobtpac, spbpers
where spriden_pidm = gobtpac_pidm
--and spriden_pidm = spbpers_pidm
and spriden_pidm = '714443'
and spriden_change_ind is null
```

Record count

Record Count: 1,171,198

☑ Copy to clipboard   Close

SPRIDEN_FIRST_NAME	SPRIDEN_LAST_NAME	GOBTPAC_LDAP_USER
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker
Danielle	Tricker	dtricker



```
select spriden_first_name,spriden_last_name,gobtpac_ldap_user
from spriden, gobtpac, spbpers
where spriden_pidm = gobtpac_pidm
and spriden_pidm = spbpers_pidm
and spriden_pidm = '714443'
and spriden_change_ind is null
```

Record count

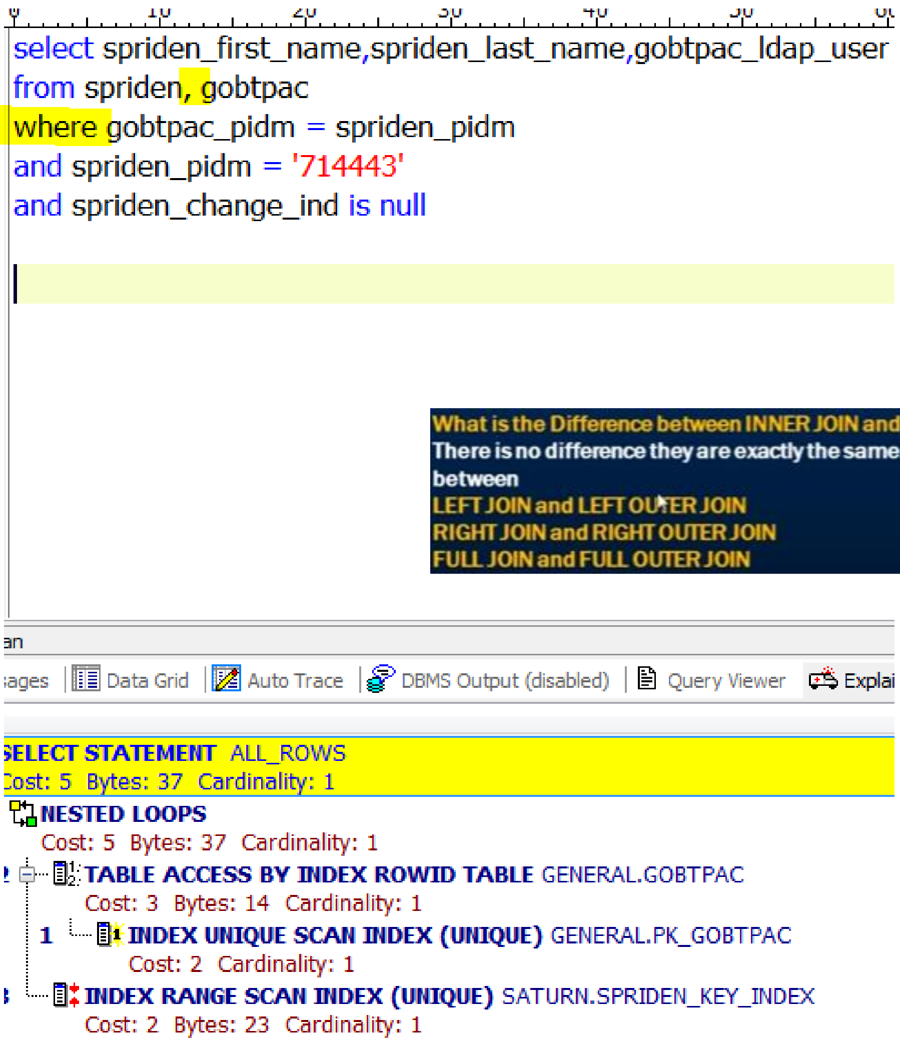
Record Count: 1

☑ Copy to clipboard   Close

SPRIDEN_FIRST_NAME	SPRIDEN_LAST_NAME	GOBTPAC_LDAP_USER
Danielle	Tricker	dtricker

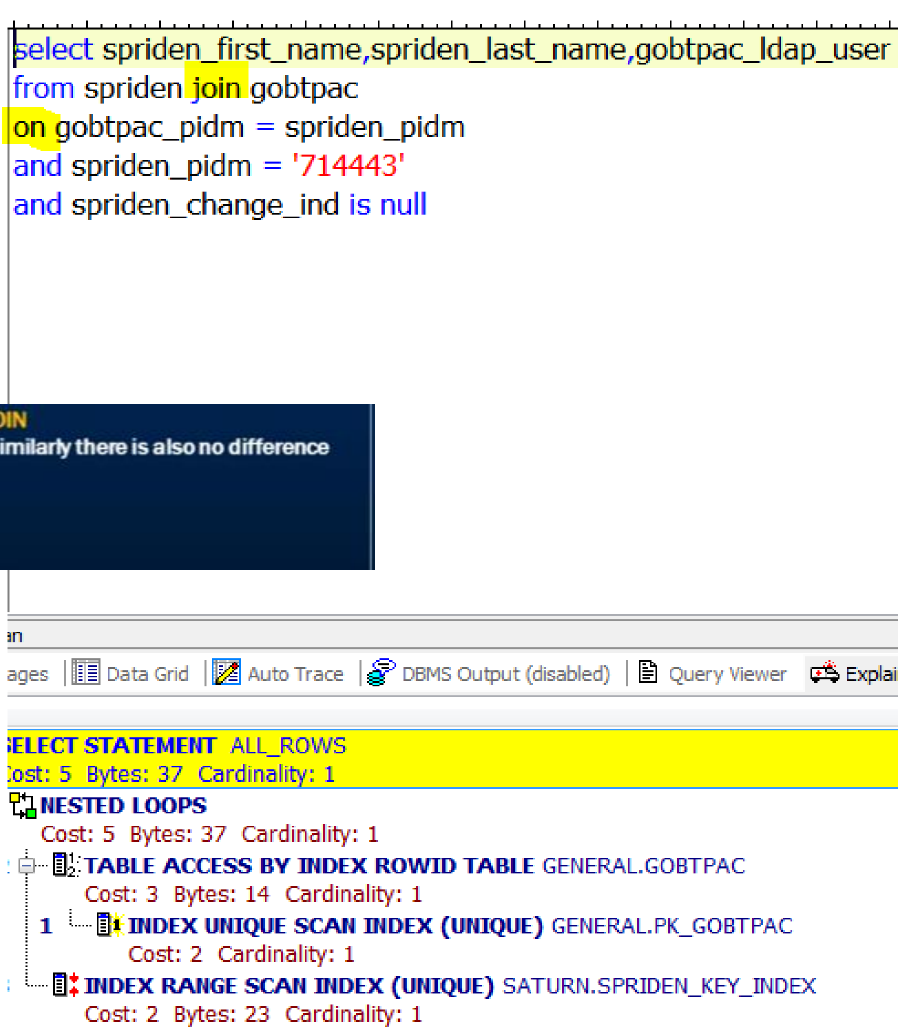
Traditional equality join vs. ANSI Join Method

(change commas to the word join, and change the where keyword to the on keyword)



```
select spriden_first_name,spriden_last_name,gobtpac_ldap_user
from spriden, gobtpac
where gobtpac_pidm = spriden_pidm
and spriden_pidm = '714443'
and spriden_change_ind is null
```

```
ELECT STATEMENT  ALL_ROWS
ost: 5  Bytes: 37  Cardinality: 1
  NESTED LOOPS
    Cost: 5  Bytes: 37  Cardinality: 1
      TABLE ACCESS BY INDEX ROWID TABLE GENERAL.GOBTPAC
        Cost: 3  Bytes: 14  Cardinality: 1
  1       INDEX UNIQUE SCAN INDEX (UNIQUE) GENERAL.PK_GOBTPAC
            Cost: 2  Cardinality: 1
      INDEX RANGE SCAN INDEX (UNIQUE) SATURN.SPRIDEN_KEY_INDEX
        Cost: 2  Bytes: 23  Cardinality: 1
```



```
select spriden_first_name,spriden_last_name,gobtpac_ldap_user
from spriden join gobtpac
on gobtpac_pidm = spriden_pidm
and spriden_pidm = '714443'
and spriden_change_ind is null
```

```
ELECT STATEMENT  ALL_ROWS
ost: 5  Bytes: 37  Cardinality: 1
  NESTED LOOPS
    Cost: 5  Bytes: 37  Cardinality: 1
      TABLE ACCESS BY INDEX ROWID TABLE GENERAL.GOBTPAC
        Cost: 3  Bytes: 14  Cardinality: 1
  1       INDEX UNIQUE SCAN INDEX (UNIQUE) GENERAL.PK_GOBTPAC
            Cost: 2  Cardinality: 1
      INDEX RANGE SCAN INDEX (UNIQUE) SATURN.SPRIDEN_KEY_INDEX
        Cost: 2  Bytes: 23  Cardinality: 1
```

What is the Difference between INNER JOIN and JOIN

There is no difference they are exactly the same. Similarly there is also no difference between

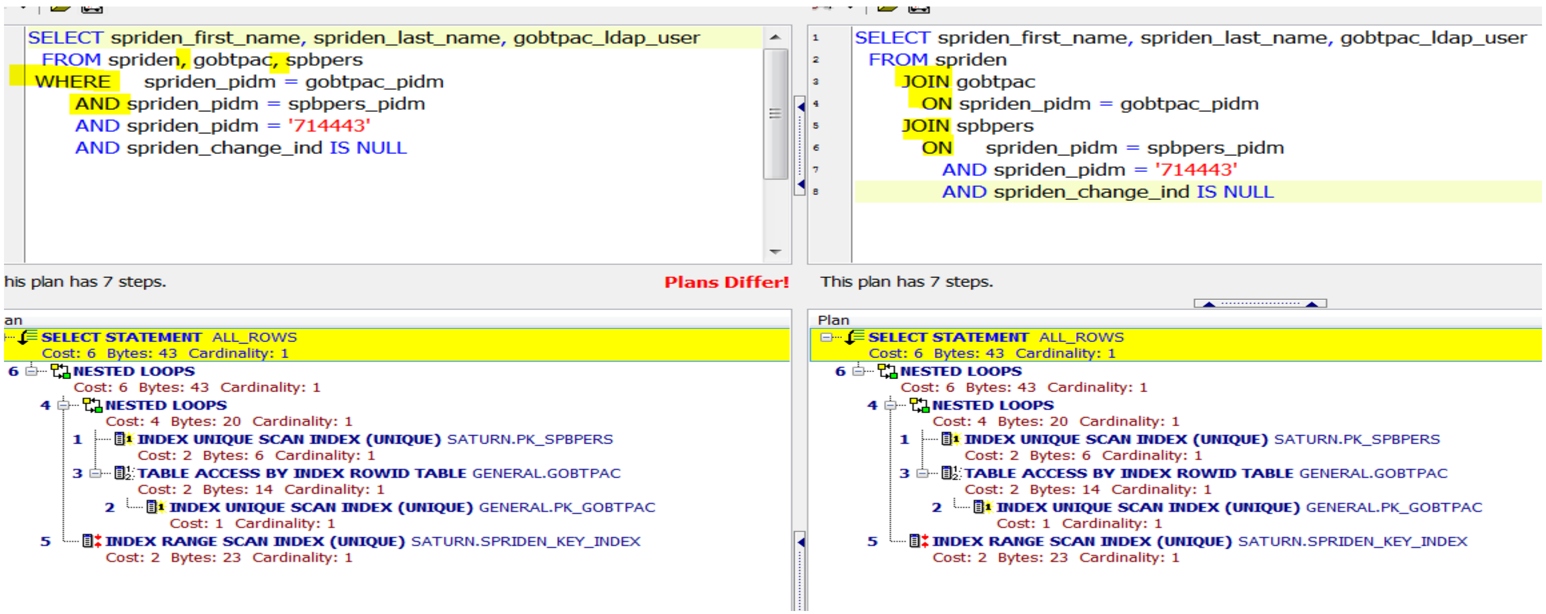
LEFT JOIN and LEFT OUTER JOIN

RIGHT JOIN and RIGHT OUTER JOIN

FULL JOIN and FULL OUTER JOIN

ANSI Join Method Syntax

1. Natural join – automatic join using matching column names
2. Using – Join on columns with the same name
3. ON – Join on columns with different names – Banner code can only use ON
   Since there are no common column names that exist in different tables



```
SELECT spriden_first_name, spriden_last_name, gobtpac_ldap_user
  FROM spriden, gobtpac, spbpers
 WHERE     spriden_pidm = gobtpac_pidm
       AND spriden_pidm = spbpers_pidm
       AND spriden_pidm = '714443'
       AND spriden_change_ind IS NULL
```

his plan has 7 steps.   **Plans Differ!**

an

```
SELECT STATEMENT  ALL_ROWS
  Cost: 6  Bytes: 43  Cardinality: 1
6 NESTED LOOPS
    Cost: 6  Bytes: 43  Cardinality: 1
  4 NESTED LOOPS
      Cost: 4  Bytes: 20  Cardinality: 1
    1 INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SPBPERS
        Cost: 2  Bytes: 6  Cardinality: 1
    3 TABLE ACCESS BY INDEX ROWID TABLE GENERAL.GOBTPAC
        Cost: 2  Bytes: 14  Cardinality: 1
      2 INDEX UNIQUE SCAN INDEX (UNIQUE) GENERAL.PK_GOBTPAC
          Cost: 1  Cardinality: 1
  5 INDEX RANGE SCAN INDEX (UNIQUE) SATURN.SPRIDEN_KEY_INDEX
      Cost: 2  Bytes: 23  Cardinality: 1
```

```
SELECT spriden_first_name, spriden_last_name, gobtpac_ldap_user
  FROM spriden
     JOIN gobtpac
       ON spriden_pidm = gobtpac_pidm
     JOIN spbpers
       ON     spriden_pidm = spbpers_pidm
          AND spriden_pidm = '714443'
          AND spriden_change_ind IS NULL
```

This plan has 7 steps.

Plan

```
SELECT STATEMENT  ALL_ROWS
  Cost: 6  Bytes: 43  Cardinality: 1
6 NESTED LOOPS
    Cost: 6  Bytes: 43  Cardinality: 1
  4 NESTED LOOPS
      Cost: 4  Bytes: 20  Cardinality: 1
    1 INDEX UNIQUE SCAN INDEX (UNIQUE) SATURN.PK_SPBPERS
        Cost: 2  Bytes: 6  Cardinality: 1
    3 TABLE ACCESS BY INDEX ROWID TABLE GENERAL.GOBTPAC
        Cost: 2  Bytes: 14  Cardinality: 1
      2 INDEX UNIQUE SCAN INDEX (UNIQUE) GENERAL.PK_GOBTPAC
          Cost: 1  Cardinality: 1
  5 INDEX RANGE SCAN INDEX (UNIQUE) SATURN.SPRIDEN_KEY_INDEX
      Cost: 2  Bytes: 23  Cardinality: 1
```

6. BANNER TIPS

- Do Ellucian DatabaseHealthCheck – download DatabaseHealthCheckV3.sql and run it
- From the above Database HealthCheck – verify good statistics are being done.
- How to find missing statistics:

```
select distinct 'exec dbms_stats.gather_table_stats(ownname=>"SATURN", tabname=>'''||table_name||''', method_opt=>"for all columns size 1", cascade=> TRUE)'
from dba_tables a, dba_objects b where a.owner = 'SATURN' and a.last_analyzed is null
and a.table_name = b.object_name and object_type = 'TABLE'
```

- Cleanup data – See Ellucian **Article 000010562 Banner Tables which should be routinely monitored and purged**
- Especially **Regular** cleanup registration tables sftregs, sfraccl, sftarea
- Do FAQ: **1-15PT6UP: Banner Student SFRFASC Batch Fee Assessment poor performance**
- Review other Ellucian performance docs such as: **Article 000034125 Banner enq TX row lock contention on SSBSECT TWGBWSES when PCTFREE INITRANS at default value in 8K Block Size tablespace and Article 000009369 Banner Performance Tuning Oracle DB**
- Monitor for Locks – see script
- Monitor for Stuck SSB session – see script
- Monitor, monitor, monitor….
- Ellucian Live 2016 Highlights

Monitor Locks

Run the below in your cron every 20 minutes and email yourself when a blocking lock occurs. This is critical when a INB user is locking out all WWW2_USER connections.

```
set feedback off;
set echo on;
connect / as sysdba
select * from dba_blockers;
exit
```

Monitor for Stuck SSB Sessions

Run this every 20 minutes in your cron. SSB sessions should never stay in an active state. Email yourself when they are stuck in a active state.

Then use a tool to show the calling sql to see where the hang up is.

Sometimes stuck sessions are due to lock, other times check for defects, such as CR-000133774- Poor performance with bwckctlg.p_display_courses after WebTailor 8.7 upgrade and twbksecr.f_escape.

```
select sid,logon_time, (last_call_et/60)/60,
  last_call_et,
 substr('0'||trunc(LAST_CALL_ET/86400),-2,2)  || ':'  ||
  -- hours
  substr('0'||trunc(mod(LAST_CALL_ET,86400)/3600),-2,2) || ':' ||
  -- minutes
  substr('0'||trunc(mod(mod(LAST_CALL_ET,86400),3600)/60),-2,2) || ':' ||
  --seconds
  substr('0'||mod(mod(mod(LAST_CALL_ET,86400),3600),60),-2,2) lastcall
from v$session s
where status = 'ACTIVE'
and username = 'WWW2_USER'
and (last_call_et/60)/60 > .05;
spool off
exit
```

7. Ellucian Live 2016

Topics of Interest	Description
ESM – Ellucian Solutions Manager	Auto installer for Banner upgrades and installs – big time saver
Application Navigator	Navigation between INB and new Java Admin pages - required
Transformation JAVA pages	INB replacement
XE pages	SSB replacement
Extensibility for XE and Transformation Pages	Allows custom modifications
ODI for ODS	Oracle Data Integrator – required purchase for next ODS upgrade
Cloud Options	Amazon attended, and Ellucian SAAS offerings

***From Ellucian: Technology Planning and Readiness Top 5 - All of the Banner XE administrative modules are scheduled for final delivery in December 2016. A transition to the Banner XE modules fully removes the dependency on any version of Oracle Forms and Reports.