TECHNICAL NOTE

GENERIC APPROACH TO NVISION PERFORMANCE TUNING

©David Kurtz
Tuesday 19 April 2016
Technical Note
Version 1.2
(E-mail: david.kurtz@go-faster.co.uk, telephone +44-7771-760660)
File: nVision Tuning.Generic.docx, 19 April 2016

Contents

Introduction

This document sets out a number of techniques that can be used to improve and stabilise the performance of PeopleSoft nVision reports. It is based on practical experience on real productions systems at a number of customer sites over a number of years.

However, it is a generic document, and makes general statements. Every customer is different, because their data is different and often their method of analysis differs.

Some of the techniques discussed are specific to the Oracle database, although in some cases, such as partitioning, similar features are available on other databases.

We will look at

- Indexing
  - Effective indexing of LEDGER and LEDGER_BUDG tables to match the analysis criteria of the reports.
  - Enhanced indexing of the PSTREESELECT tables, so that the indexes fully satisfy the queries without the need to visit the tables.
- Collection of statistics and extended statistics on the PSTREESELECT tables.
- Using the nVision performance options to
  - use static selectors instead of dynamic selectors. They make it difficult to maintain up-to-date optimizer statistics on the selector tables
  - simplify SQL statements by replacing joins with literal crtieria
  - I also suggest use of Oracle Fine Grained Auditing to
    - enhance instrumentation,
    - detect the use of dynamic selectors.
- Appropriate partitioning of the LEDGER and LEDGER_BUDG tables.
  - If the partitioning option is not available, then I strongly recommended that as much historical data as possible is purged from the LEDGER and LEDGER_BUDG tables.
- Archiving.

Ledger and Tree Selector Indexing

PeopleSoft delivers a one-size-fits-all approach to indexing, while the shape of customer data and to some extent the way it is used varies from system to system. This leads to the need for different customers using of the same PeopleSoft module to sometimes require different indexes.

Ledger Indexing

The indexing on the PS_LEDGER and PS_LEDGER_BUDG tables needs to match the analysis criteria used in the nVision report. There may be other custom ledger tables that also need to be indexed appropriately. This will vary from system to system. So the indexing of these tables is also a matter for each customer. It is usually necessary to add additional indexes to these tables for the benefit of nVision.

In general

- columns with single value equality criteria are at the front of the index,
- columns with multiple criteria or inequality criteria are next,
- colums with inequality crtieria are last

Let's look at some queries produced by nVision

```
SELECT A.ACCOUNT, SUM( A.POSTED BASE AMT)
FROM PS LEDGER A
WHERE (A.LEDGER IN ('LOCAL','OTHER')
AND A.FISCAL_YEAR = 2014
AND A.ACCOUNTING_PERIOD = 5
AND (A.ACCOUNT='4xxxxxx' OR A.ACCOUNT='4xxxxxx' …)
AND (A.OPERATING UNIT='USxxx' OR A.OPERATING UNIT='USxxx' OR …)
AND A.BUSINESS UNIT = 'USxxx')
GROUP BY A.ACCOUNT
```

I would think of building and index the following columns: LEDGER, FISCAL_YEAR, ACCOUNTING_PERIOD, BUSINESS_UNIT, OPERATING_UNIT, ACCOUNT. This is a single period query so ACCOUNTING_PERIOD is near the front of the index, immediately after fiscal year.

ACCOUNTING_PERIOD is never used without using FISCAL_YEAR, though occasionaly you might see a query by FISCAL_YEAR that does not reference ACCOUNTING_PERIOD. So ACCOUNTING_PERIOD should never preceed FISCAL_YEAR in the list of indexed columns.

LEDGER was put at the front of the index because there are fewer values than there are for fiscal year.

```
SELECT B.TREE NODE NUM, SUM( A.POSTED BASE AMT)
FROM PS LEDGER A, PSTREESELECT10 B
WHERE ( A.LEDGER IN ('LOCAL','OTHER')
AND A.FISCAL YEAR = 2014
AND A.ACCOUNTING PERIOD BETWEEN 1 AND 5
AND B.SELECTOR_NUM=2xxxxxx
AND A.ACCOUNT BETWEEN B.RANGE_FROM_10 AND B.RANGE_TO_10
AND B.TREE NODE NUM BETWEEN 1500000000 AND 1749999999
AND (A.OPERATING_UNIT='USxxx' OR A.OPERATING_UNIT='USxxx' OR …)
GROUP BY B.TREE_NODE_NUM
```

This time I would suggest an index on the following columnd: LEDGER, FISCAL_YEAR, OPERATING_UNIT, ACCOUNT, ACCOUNTING_PERIOD. This is a year-to-date query, so this time ACCOUNTING_PERIOD has been pushed to the back of the list of columns.

It is often necessary to build pairs of indexes for each set of analysis criteria. One for single period queries, one for year to date queries.

Sometimes, I have added columns that are not selective, but which are referenced by the query, so that the index can satisfy the query without needing to visit the table.

Consider the following query.

```
SELECT A.ACCOUNT,SUM(A.POSTED TOTAL AMT)
FROM  PS LEDGER A
WHERE A.LEDGER='ACTUALS'
AND   A.FISCAL_YEAR=2016
AND   A.ACCOUNTING_PERIOD BETWEEN 1 AND 12
AND   A.CURRENCY_CD='GBP'
AND   A.STATISTICS_CODE=' '
AND  (A.BUSINESS_UNIT=…)
GROUP BY A.ACCOUNT
```

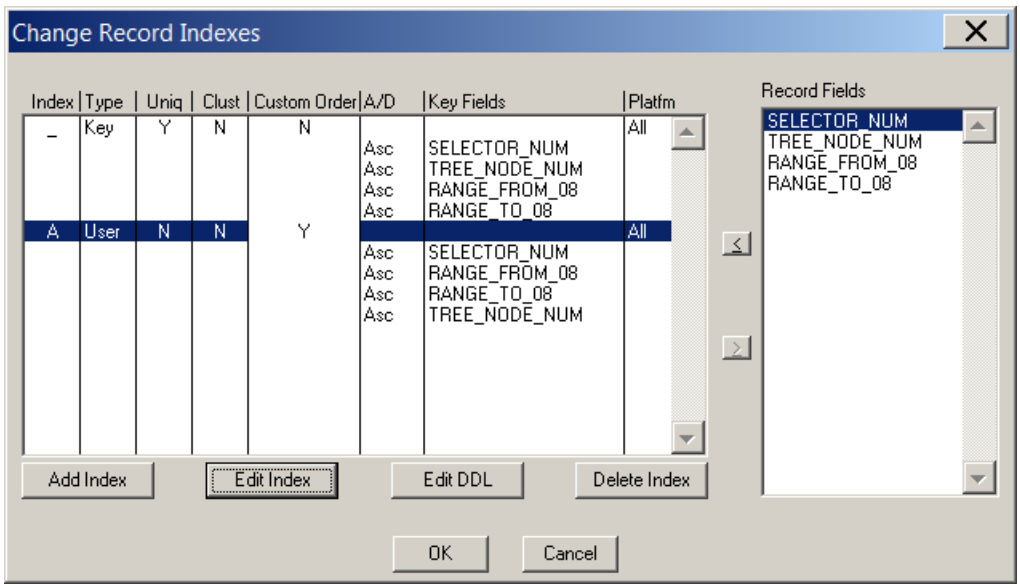
- Currency code will not be a selective column in a single currency system.
- Statistics code is rarely selective.
- There is no criteria on POSTED_TOTAL_AMT.

However, I could put all three columns into the index, so that there is no need to look up the row on the table in order to obtain the values.

Tree Selector Indexing

All the tree selector tables should have the same indexes on the following columns

- PS_PSTREESELECT*nn* (where *nn* is in the range 01 to 30):
  - The columns on the unique index are unaltered as: SELECTOR_NUM, TREE_NODE_NUM, RANGE_FROM_*nn*, RANGE_TO_*nn*.
- PSAPSTREESELECT*nn*.
  - SELECTOR_NUM, RANGE_FROM_*nn*, RANGE_TO_*nn*, TREE_NODE_NUM
    - Additional columns have be added to this delivered index. The idea is to be able to fully satisfy the queries on the tree selector tables without need to visit the table.



Optimal Index Compression

Oracle can compress more rows into a single index leaf block by storing each distinct combination of compressed column values found within a specific index leaf block in a "prefix" table within that block, and referring to it rather that storing the column values in each index entry[1]. This form of compression is not subject to a special Oracle licence.

The compression prefix length (the number of leading columns to be considered for compression) can be specified when the index is created. Columns with few distinct values result in better compression. Putting columns with lower cardinality at the front of the index can result in better compression, but this must not be done at the expense of the needs of the application. If too many columns are specified in the prefix length the prefix table could contain one entry for every row in the block and the compressed index would be larger than the uncompressed index. There is therefore an optimal compression for every index that depends on the data in the indexed columns. Oracle can calculate that optimal value for you with the ANALYZE INDEX VALIDATE STRUCTURE command. I have written a script[2] to do this for each index and index partition on a given table.

The indexes on these tables should be compressed.

- PS_LEDGER
- PS_LEDGER_BUDG
- PSTREESELECT__

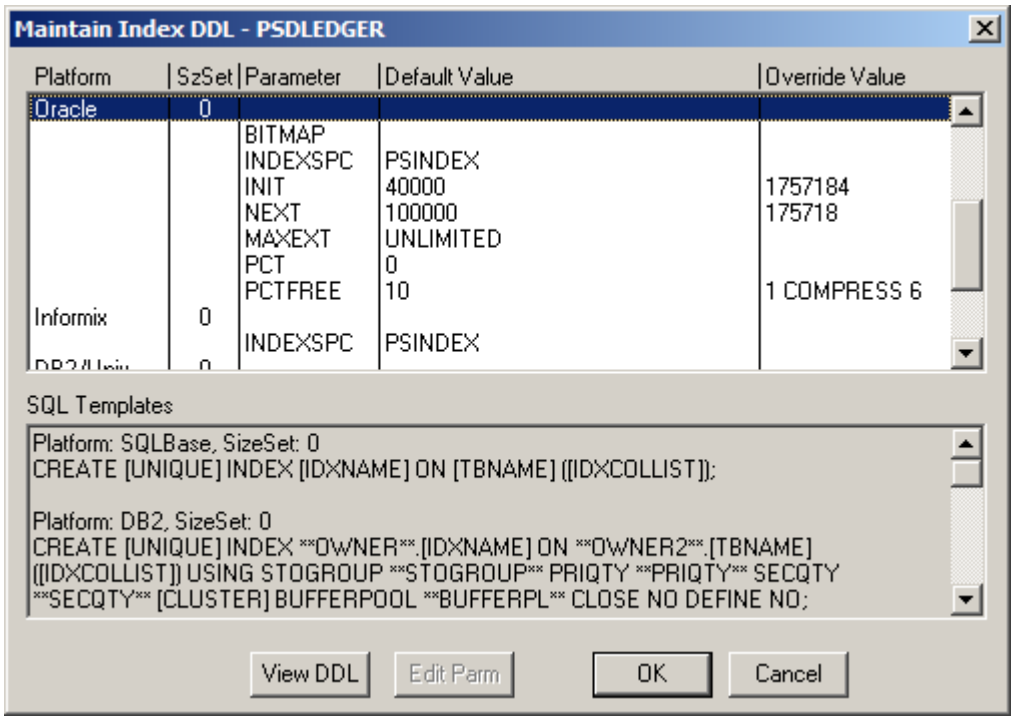
NB: If there is a recommended non-zero prefix length but a zero saving, then the compression has already been applied.

Table Name	Index Name	Partition Name	Optimal Compression Prefix Length	BLOCKS	Saving %
…					
PS_LEDGER	PSELEDGER		6	42624	.0
	PSFLEDGER		4	21504	.0
	PSGLEDGER		**6**	**29184**	**.0**
	PSHLEDGER		5	32768	.0
	PS_LEDGER		19	81280	.0
…					

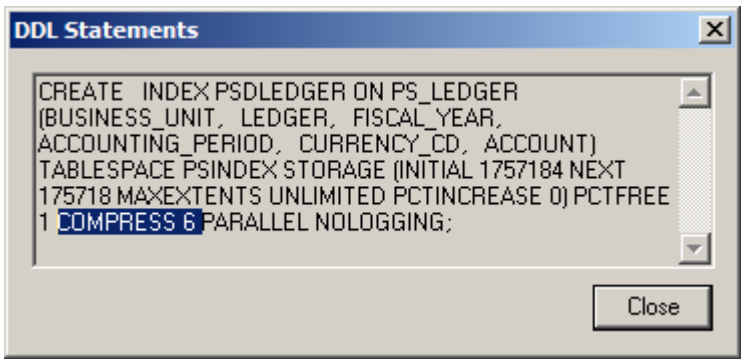
[1] See https://richardfoote.wordpress.com/2008/02/17/index-compression-part-i-low/

[2] See calc_opt_comp.sql script at http://www.go-faster.co.uk/scripts.htm#calc_opt_comp.sql

The recommended prefix lengths are specified in the create index DDL by adding the keyword COMPRESS followed by the prefix length. They should be applied in PeopleTools Application Designer as index overrides to the PCTFREE parameter as shown below[3].



They appear in the DDL thus

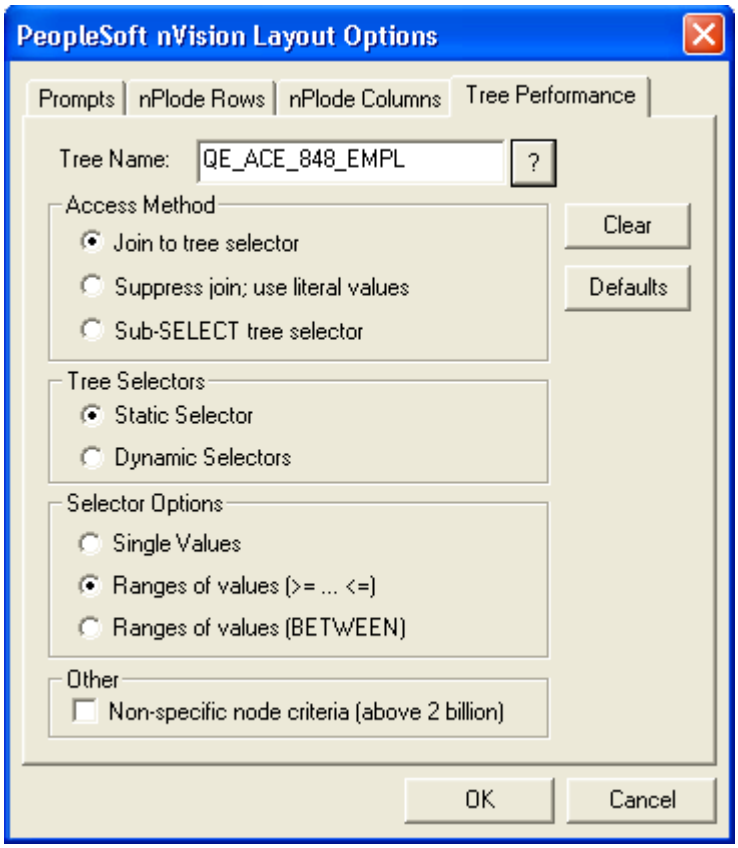


[3] See also Implementing Index Compression (and other Physical Storage Options) via Application Designer (see http://blog.psftdba.com/2016/02/implementing-index-compression-and.html)

nVision Tree Performance Options

PS/nVision has always had 'Tree Performance Options; panel that can be accessed in the nVision add-in to Excel. The settings change the SQL generated by nVision without any functional change to the report.

From PeopleTools 8, Tree Peformance Options can be set on the tree definition in the database. This setting can then be overridden in individual nVision layout files. This screenshot is from the nVision PeopleBook[4]. Note that the operator must have security access to the Performance Options in order that the 'Tree Performance' tab is visible



4 http://docs.oracle.com/cd/E41509_01/pt852pbh2/eng/psbooks/tnvs/htm/tnvs15.htm#_4ecd4b88_13f1d8f8763__650b

Generally Recommended Tree Performance Options

I would generally recommend that they should be set as follows:

- Suppress join; use literal values. nVision breaks the queries down by executing some tree queries separately and putting literal values into the final query.
  - For very large trees (>3000 tree leaves) the resulting SQL can contain very many predicates and the parse time can be significant. 'Join' should be used for such trees.
- Use Static Selectors.
  - When an nVision report uses dynamic selectors, it copies the part of the tree it wants to use into one of the PSTREESELECT tables using a new selector number, and deletes them at the end. If an nVision using dynamic selectors crashes then these rows can be left behind in the table and over time this can build up and affect table size and statistics on the table.
  - Because dynamic selectors continues alters the contents of the PSTREESELECT tables, and continues selects increasing values for SELECTOR_NUM, it is always a challenge to keep the statistics up to date on these tables. Out of date statistics, particularly the column high value of SELECTOR_NUM can result Oracle miscasting the selectivity of cirteria, and lead to sub-optimal execution plans.
  - When using static selectors, the whole of each tree required is copied into the tree selector table, but it is only copied again when the tree changes.
- Use Ranges of Values BETWEEN
  - There is no difference between the two range options in Oracle. BETWEEN criteria are expanded during SQL parse to 2 inequalities during SQL optimisation, but the SQL generated by nvision is smaller.when using the BETWEEN option. Fewer predicates are likely to be generated with this options than with single value options

From PeopleTools 8.x, nVision Tree Performance Options can be set on the tree definition stored in the database. This will be used in all nVision layouts unless specifically overridden in the layout.

Implementation

1. **Action:** Update nVision Tree Performance Options on All Trees by running nvperfopts-6trees.sql scripts in SQL*Plus connected as SYSADM.
    a. **NB: Going forward, any new trees must be created using the Performance Options as recommended. Those people who create trees should be informed of this. However, this script can always be run again to reset all trees.**
    b. This script sets the performance options access method to 'Use Literal Values'
2. Any nVision layouts which specify nVision performance options will override the settings on the tree and will need to be changed to confirm to these settings described above[5].
    c. Should it be necessary to reverse these changes out, the option will be to restore previous versions of the layouts. So backups should be made before making any changes.

[5] The PeopleSoft VB utility has been fixed so that it works on the current version of PeopleSoft/MS Office. It can do this in bulk (per directory).

```
REM nvperfopts.sql
REM (c)2013 David Kurtz

spool nvperfopts
set pages 99
break on setid on tree_name skip 1
rollback
/
SELECT T.SETID, T.TREE NAME, T.EFFDT
, T.TREE ACC SELECTOR --static selctors
, (SELECT X1.XLATSHORTNAME FROM PSXLATITEM X1 WHERE X1.FIELDNAME =
'TREE_ACC_SELECTOR' AND X1.FIELDVALUE = TREE_ACC_SELECTOR)
, T.TREE_ACC_SEL_OPT --between
, (SELECT X2.XLATSHORTNAME FROM PSXLATITEM X2 WHERE X2.FIELDNAME =
'TREE ACC SEL OPT' AND X2.FIELDVALUE = TREE ACC SEL OPT)
, T.TREE ACC METHOD --literals
, (SELECT X3.XLATSHORTNAME FROM PSXLATITEM X3 WHERE X3.FIELDNAME =
'TREE_ACC_METHOD' AND X3.FIELDVALUE = TREE_ACC_METHOD)
FROM PSTREEDEFN T
WHERE (T.TREE_ACC_SELECTOR != 'S'
OR T.TREE ACC SEL OPT != 'B'
OR T.TREE ACC METHOD !='L')
AND X1.FIELDNAME = 'TREE_ACC_SELECTOR'
ORDER BY 1,2,3
/

/*increment the version numbers[6]*/
UPDATE PSLOCK
SET VERSION = VERSION + 1
WHERE OBJECTTYPENAME IN('SYS','TDM')
/

UPDATE PSVERSION
SET VERSION = VERSION + 1
WHERE OBJECTTYPENAME IN('SYS','TDM')
/

/*update nvision flags and version number on trees*/
/*This is the general setting[7]*/
UPDATE PSTREEDEFN
SET TREE_ACC_SELECTOR = 'S' --static selctors
, TREE ACC SEL OPT = 'B' --between
, TREE ACC METHOD = 'L' --literals
, VERSION = (SELECT VERSION FROM PSLOCK WHERE OBJECTTYPENAME = 'TDM')
, lastupddttm = SYStimestamp
, lastupdoprid = 'DAVID.KURTZ'
WHERE (TREE_ACC_SELECTOR != 'S'
OR TREE ACC SEL OPT != 'B'
OR TREE ACC METHOD != 'L')
AND tree strct id IN(SELECT tree strct id FROM pstreestrct WHERE node fieldname
= 'TREE_NODE')
/

/*This is the exception for a very large tree that is resulting in massive
SQL[8]*/
UPDATE PSTREEDEFN
SET TREE_ACC_SELECTOR = 'S' --static selctors
, TREE_ACC_SEL_OPT = 'B' --between
, TREE ACC METHOD = 'J' --join
, VERSION = (SELECT VERSION FROM PSLOCK WHERE OBJECTTYPENAME = 'TDM')
, lastupddttm = SYStimestamp
, lastupdoprid = 'DAVID.KURTZ'
WHERE setid = 'SHARE'
```

[6] Note that this script updates the PeopleTools version numbers on the trees that it updates causing them to be automatically recached by PeopleTools.

[7] This update statement updated the nVision options on all trees to the default settings.

[8] This second update statement updates the nVision performance options to slightly different for very large trees – it should be used for tree with more than 5000 nodes. The list of trees should be hard coded, there is only likely to be a few.

```
AND tree_strct_id IN(SELECT tree_strct_id FROM pstreestrct WHERE node_fieldname
= 'TREE_NODE')
AND tree name IN ('LIST NAMES OF VERY LARGE TREES');
AND 1=2
/

SELECT T.SETID, T.TREE_NAME, T.EFFDT
, T.TREE_ACC_SELECTOR --static selctors
, (SELECT X1.XLATSHORTNAME FROM PSXLATITEM X1 WHERE X1.FIELDNAME =
'TREE ACC SELECTOR' AND X1.FIELDVALUE = TREE ACC SELECTOR)
, T.TREE_ACC_SEL_OPT --between
, (SELECT X2.XLATSHORTNAME FROM PSXLATITEM X2 WHERE X2.FIELDNAME =
'TREE_ACC_SEL_OPT' AND X2.FIELDVALUE = TREE_ACC_SEL_OPT)
, T.TREE_ACC_METHOD --literals
, (SELECT X3.XLATSHORTNAME FROM PSXLATITEM X3 WHERE X3.FIELDNAME =
'TREE ACC METHOD' AND X3.FIELDVALUE = TREE ACC METHOD)
FROM PSTREEDEFN T
WHERE (T.TREE_ACC_SELECTOR = 'S'
OR T.TREE_ACC_SEL_OPT = 'B'
OR T.TREE_ACC_METHOD IN('J','L'))
AND t.tree strct id IN(SELECT tree strct id FROM pstreestrct WHERE
node fieldname = 'TREE NODE')
ORDER BY 1,2,3
/

select lastrefreshdttm from psstatus
/
UPDATE psstatus
SET lastrefreshdttm = SYSTIMESTAMP
/
select lastrefreshdttm from psstatus
/

select l.setid, l.tree_name, l.effdt, x.selector_num, count(*)
from pstreeleaf l, pstreeselctl x
where l.setid = x.setid
and l.tree name = x.tree name
and l.effdt = x.effdt
and l.setcntrlvalue = x.setcntrlvalue
group by x.selector_num, l.setid, l.setcntrlvalue, l.tree_name, l.effdt
order by 1,2,3
/
spool off
```

nvperfopts.sql

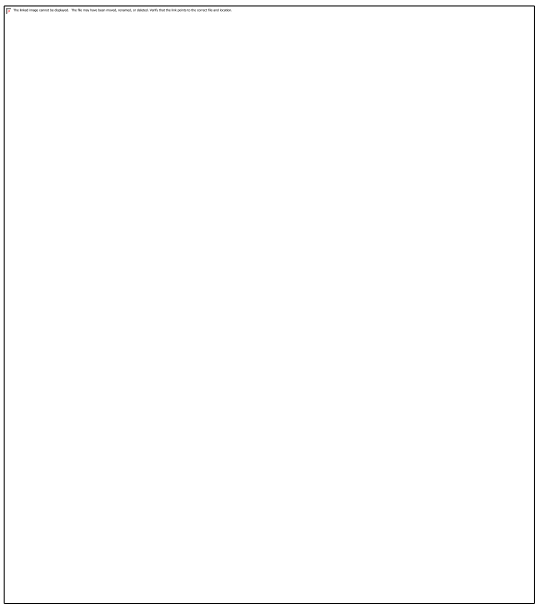
Reversal

To be able to reverse this change out it will be necessary to have a backup copy of the table PSTREEDEFN from which to restore the values of the tree TREE_ACC% columns. Remember also to update the version number on the trees as follows.

```
UPDATE PSLOCK SET VERSION = VERSION + 1 WHERE OBJECTTYPENAME IN('SYS','TDM');
UPDATE PSVERSION SET VERSION = VERSION + 1 WHERE OBJECTTYPENAME
IN('SYS','TDM');
UPDATE PSTREEDEFN a
SET VERSION = (SELECT version FROM pslock WHERE objecttypename = 'TDM')
, (tree acc selector, tree acc sel opt, tree acc method, lastupdoprid,
lastupddttm) = (
        SELECT b.tree_acc_selector, b.tree_acc_sel_opt, b.tree_acc_method,
b.lastupdoprid, b.lastupddttm
        FROM   <backup of pstreenode> b
        WHERE  a.setid = b.setid
        AND    a.tree name = b.tree name
        AND    a.effdt = b.effdt)
/
```

Tree Performance Overrides in Layouts

Tree performance options set at tree level can be overridden in the layouts[9]. Any overrides in layouts should be removed, although we may encounter specific cases where they are beneficial.



PeopleSoft created a utility to bulk update performance options in nVision layouts that may need to be used. I have had some help getting it working again on modern versions of MS Office Excel.

PerfOptionsAnalysisToolDocumentation.zip

PerfOptionSettingAnalysis.Fixed.zip

[9] Historical footnote: In fact, originally in PeopleTools tree performance options could only be set in the layouts. The options at tree level were added in PT8.4

Coalescing Tree Leaves

A consequence of the Tree Performance Access Method 'suppress join; use literal values' is that the resulting SQL in nVision will have a criteria for every leaf on each of the selected nodes on that tree. There will be an equality condition for each single value leaf. I normally set Selector Options to 'Ranges of values (BETWEEN)', so I get a between condition for each ranged leaf. Behind the scenes, Oracle rewrites between as a pair of inequalities, so there is no difference, but the SQL generated by nVision is slightly shorter.

The following is typical of nVision SQL with these performance options set.

```
SELECT A.ACCOUNT,SUM(A.POSTED_TOTAL_AMT)
FROM
 PS_LEDGER A WHERE A.LEDGER='ACTUALS' AND A.FISCAL_YEAR=2015 AND
  A.ACCOUNTING PERIOD BETWEEN 1 AND 12 AND A.CURRENCY CD='GBP' AND
  A.STATISTICS CODE=' ' AND (A.BUSINESS UNIT=
…
) AND (
  A.DEPTID='C500' OR A.DEPTID='C512' OR A.DEPTID='C117' OR A.DEPTID='C157' OR
  A.DEPTID='C340' OR A.DEPTID='C457' OR A.DEPTID='C510' OR A.DEPTID='A758' OR
  A.DEPTID='8220' OR A.DEPTID='A704' OR A.DEPTID='A121' OR A.DEPTID='A110' OR
  A.DEPTID BETWEEN 'A153' AND 'A154' OR A.DEPTID BETWEEN '1151' AND '1152' OR
  A.DEPTID='A724' OR A.DEPTID BETWEEN '1131' AND '1133' OR A.DEPTID='A733' OR
  A.DEPTID='A217' OR A.DEPTID='A437' OR A.DEPTID='A130' OR A.DEPTID='A134' OR
  A.DEPTID='A703' OR A.DEPTID='A714' OR A.DEPTID='A218' OR A.DEPTID='A226' OR
  A.DEPTID BETWEEN 'A135' AND 'A138'
…
```

Although this access method can result in good execution time, it can also increase the time taken for Oracle to parse the SQL statements such that this can become significant. It may only be a few seconds in the case of a single SQL statement, but an nVision report book can consist of thousands of SQL statements.

One way to reduce the parse time is simply to reduce the number of criteria in the SQL statement by reducing the number of leaves on the tree. Tree leaves can be single values or ranges of values, by coalescing adjacent leaves into ranged leaves the number of leaves can be reduced.

The *leafcoal.sql* script seeks to address this by repeatedly merging two consecutive leaves on the same tree node into a single ranged leaf where possible. It performs two checks before merging adjacent leaves on the same tree node:

- There is not an intermediate value on the detail field defined in the tree structure record. So if the detail field was DEPT_TBL.DEPTID, the script checks that there are no values of DEPTID on PS_DEPT_TBL that are not currently selected by existing leaves that would be included in the merged leaf.
- There is not another leaf on another node on the tree that would intersect with the merged leaf.

Instructions for running leafcoal.sql

The utility is written in PL/SQL and runs as an anonymous blocks so there is nothing to install. It should be run in SQL*Plus connected as the PeopleSoft owner ID (usually SYSADM).

It is expected that there are some adjustments to the script that the user may need to make.

```
REM leafcoal.sql
REM (c)David Kurtz 2016
REM 06.04.2016 detect overlapping ranges on other tree nodes
REM 07.04.2016 added update leaf count

rollback;
spool leafcoal
----------------------------------------------------------------------------------------------------
set serveroutput on termout off trimspool on
DECLARE
k_testmode     CONSTANT BOOLEAN := TRUE[10]; /*set this false to perform update*/
k_dateformat   CONSTANT VARCHAR2(10) := 'yymmdd';
k_oprid        CONSTANT VARCHAR2(20) := 'david.kurtz'[11];
l_module       VARCHAR2(48);
l_action       VARCHAR2(32);
l_debug_level  INTEGER := 4;[12]
l_debug_indent INTEGER := 0;
----------------------------------------------------------------------------------------------------
no_structure_found EXCEPTION;
PRAGMA EXCEPTION_INIT(no_structure_found,-20001);
----------------------------------------------------------------------------------------------------
PROCEDURE debug_msg(p_text VARCHAR2 DEFAULT ''
                   ,p_debug_level INTEGER DEFAULT 5) IS
BEGIN
  IF p_debug_level <= l_debug_level AND p_text IS NOT NULL THEN
    sys.dbms_output.put_line(LPAD('.',l_debug_indent,'.')||'('||p_debug_level||')'||p_text);
  END IF;
END debug_msg;
----------------------------------------------------------------------------------------------------
PROCEDURE set_action(p_action_name VARCHAR2 DEFAULT ''
                    ,p_debug_level INTEGER DEFAULT 5) IS
BEGIN
  l_debug_indent := l_debug_indent + 1;
  dbms_application_info.set_action(action_name=>p_action_name);
  debug_msg(p_text=>'Setting action to: '||p_action_name,p_debug_level=>p_debug_level);
END set_action;
----------------------------------------------------------------------------------------------------
PROCEDURE unset_action(p_action_name VARCHAR2 DEFAULT ''
                      ,p_debug_level INTEGER DEFAULT 7) IS
BEGIN
  IF l_debug_indent > 0 THEN
    l_debug_indent := l_debug_indent - 1;
  END IF;
  dbms_application_info.set_action(action_name=>p_action_name);
  debug_msg(p_text=>'Resetting action to: '||p_action_name,p_debug_level=>p_debug_level);
END unset_action;
----------------------------------------------------------------------------------------------------
FUNCTION show_bool(p_bool BOOLEAN) RETURN VARCHAR IS
BEGIN
  IF p_bool THEN
    RETURN 'TRUE';
  ELSE
    RETURN 'FALSE';
  END IF;
END show_bool;
----------------------------------------------------------------------------------------------------
--identify the table, record and field referred to in the tree
----------------------------------------------------------------------------------------------------
```

[10] As delivered, *leafcoal.sql* runs in a test mode that does not update the database but reports on what it would do. Change *k_testmode* to FALSE to have script update the PeopleTools tables.

[11] When the script updates the tree leaves it also updates the tree definition, and updates the last operation ID to this value. Change this as necessary.

[12] The level of output written to the spool file can be controlled by changing the value of the variable *l_debug_level*.

1. end of processing message
2. start of processing for tree
3. number of leaves in tree and number of leaves coalesced
4. details of leaves being compressed
5. start and end of each procedure
6. parameters passed to functions
7. number of rows updated/deleted during coalesce
8. dynamic SQL statement

```
PROCEDURE get_structure_field
(p_setid         IN VARCHAR2 DEFAULT ' '
,p_setcntrlvalue IN VARCHAR2 DEFAULT ' '
,p tree name     IN VARCHAR2 DEFAULT ' '
,p effdt         IN DATE
,p dtl tabname   OUT VARCHAR2
,p dtl recname   OUT VARCHAR2
,p dtl fieldname OUT VARCHAR2
) IS
  l_module VARCHAR2(48);
  l action VARCHAR2(32);
BEGIN
  dbms application info.read module(module name=>l module, action name=>l action);
  set action(p action name=>'get structure field('||p tree name||'/'||TO CHAR(p effdt,k dateformat)||')');
  debug msg('get structure field('||p setid||'/'||p setcntrlvalue||'/'||p tree name
                                  ||'/'||TO CHAR(p effdt,k dateformat)||')',6);
  SELECT DECODE(r.sqltablename,' ','PS '||r.recname,r.sqltablename) dtl tabname
                   , s.dtl_recname, s.dtl_fieldname
  INTO   p_dtl_tabname, p_dtl_recname, p_dtl_fieldname
  FROM   pstreedefn d
  ,      pstreestrct s
  ,      psrecdefn r
  WHERE  d.setid = p setid
  AND    d.setcntrlvalue = p setcntrlvalue
  AND    d.tree name = p tree name
  AND    d.effdt = p_effdt
  AND    d.tree_strct_id = s.tree_strct_id
  AND    s.node_fieldname = 'TREE_NODE'
  AND    r.recname = s.dtl_recname
  ;
  debug msg('structure field: '||p dtl tabname||'.'||p dtl fieldname,5);
  unset action(p action name=>l action);

EXCEPTION
  WHEN no_data_found THEN
    debug_msg('No tree structure record found',3);
    unset_action(p_action_name=>l_action);
    RAISE_APPLICATION_ERROR(-20001,'No structure record found for tree '
          ||p_setid||'/'||p_setcntrlvalue||'/'||p_tree_name||'/'||TO_CHAR(p_effdt,k_dateformat));
END get_structure_field;
--------------------------------------------------------------------------------------------------------------
--process one named tree
--------------------------------------------------------------------------------------------------------------
PROCEDURE one tree
(p_setid         IN VARCHAR2 DEFAULT ' '
,p_setcntrlvalue IN VARCHAR2 DEFAULT ' '
,p_tree_name     IN VARCHAR2 DEFAULT ' '
,p_effdt         IN DATE
) IS
  l dtl tabname   psrecdefn.sqltablename%TYPE;
  l dtl recname   pstreestrct.dtl recname%TYPE;
  l dtl fieldname pstreestrct.dtl fieldname%TYPE;
  l sql CLOB;

  TYPE refcur IS REF CURSOR;
  c_treeleaf refcur;

  l tree node num   pstreeleaf.tree node num%TYPE;
  l range from      pstreeleaf.range from%TYPE;
  l range to        pstreeleaf.range to%TYPE;
  l next range from pstreeleaf.range from%TYPE;
  l next range to   pstreeleaf.range to%TYPE;
  l coal range from pstreeleaf.range from%TYPE := '';
  l_coal_range_to   pstreeleaf.range_to%TYPE   := '';
  l_tree_branch     pstreeleaf.tree_branch%TYPE;

  l leaf count INTEGER;
  l coal count INTEGER := 0;
  l node count INTEGER;

  l module VARCHAR2(48);
  l_action VARCHAR2(32);
BEGIN
  set_action(p_action_name=>'one_tree('||p_tree_name||'/'||TO_CHAR(p_effdt,k_dateformat)||')');
  debug_msg('Processing '||p_setid||'/'||p_setcntrlvalue||'/'||p_tree_name
                                      ||'/'||TO CHAR(p effdt,k dateformat),2);
  get structure field(p setid, p setcntrlvalue, p tree name, p effdt, l dtl tabname, l dtl recname, l dtl fieldname);

  SELECT COUNT(*)
  INTO   l_node_count
  FROM   pstreenode
  WHERE  setid = p_setid
  AND    setcntrlvalue = p_setcntrlvalue
  AND    tree_name = p_tree_name
  AND    effdt = p_effdt;

  SELECT COUNT(*)
  INTO   l leaf count
  FROM   pstreeleaf
  WHERE  setid = p_setid
  AND    setcntrlvalue = p_setcntrlvalue
  AND    tree_name = p_tree_name
  AND    effdt = p_effdt;

  debug msg(l node count||' nodes, '||l leaf count||' leaves',4);
  --dynamic sql to find coalescible leaves.  No ref values between ranges or single values
  --do not process branched trees
  --Dynamic because structure field is configurable
  --assume SETID in key field
  --ignore EFFDT processing.  Do not coalesce even if not effective as at tree date
  l_sql := 'WITH x AS (
SELECT l.*
,      LEAD(range_from,1) OVER (PARTITION BY setid, setcntrlvalue, effdt, tree_node_num, tree_branch ORDER BY range_from)
next_range_from
,      LEAD(range_to,1) OVER (PARTITION BY setid, setcntrlvalue, effdt, tree_node_num, tree_branch ORDER BY range_from)
next_range_to
FROM   pstreeleaf l
WHERE  l.setid = :p1
AND    l.setcntrlvalue = :p2
AND    l.tree name = :p3
AND    l.effdt = :p4
), y AS (
SELECT  x.*
```

[13]

[13] This section builds the SQL for a dynamic query identifies leaves on the same tree node for which there are no intermediate values on column referred to by the tree structure, nor on any leaf on a different tree node

```
,       (SELECT count(*)
        FROM    '||l_dtl_tabname||' d
        WHERE   ';

  FOR i IN (
    SELECT fieldname
    FROM   psrecfielddb
    WHERE  recname = l_dtl_recname
    AND    MOD(useedit,2) = 1
    AND    fieldname = 'SETID'
  ) LOOP
    l_sql := l_sql||'d.'||i.fieldname||' = x.'||i.fieldname||' AND ';
  END LOOP;

  l_sql := l_sql||'d.'||l_dtl_fieldname||' > x.range_to
            AND   d.'||l_dtl_fieldname||' < x.next_range_from) intermediates
,       (SELECT count(*)
        FROM    PSTREELEAF f
        WHERE   f.setid = x.setid
        AND     f.setcntrlvalue = x.setcntrlvalue
        AND     f.tree_name = x.tree_name
        AND     f.effdt = x.effdt
        AND     f.tree_node_num != x.tree_node_num
        AND     f.range_from <= x.next_range_to
        AND     f.range_to >= x.range_from
        ) overlapers FROM    x
WHERE   next_range_From is not null
)
select tree_node_num, range_from, range_to, next_range_from, next_range_to, tree_branch
from y
where intermediates=0
and   overlapers=0
order by 1,2';
  debug_msg('SQL:'||l_sql,8);

  OPEN c_treeleaf FOR l_sql USING p_setid, p_setcntrlvalue, p_tree_name, p_effdt;
  LOOP
    FETCH c_treeleaf INTO l_tree_node_num, l_range_from, l_range_to, l_next_range_from, l_next_range_to, l_tree_branch;
    EXIT WHEN c_treeleaf%NOTFOUND;
    IF l_coal_count = 0 THEN
      IF k_testmode THEN --update tree version prior to first coalesce, but not in test mode
        debug_msg('Running in Test Mode - No Update',1);
      ELSE[14]
        UPDATE psversion
        SET    version = version+1
        WHERE  objecttypename IN('SYS','TDM');

        UPDATE pslock
        SET    version = version+1
        WHERE  objecttypename IN('SYS','TDM');
      END IF;
    END IF;
    l_coal_count := l_coal_count + 1;

    IF l_range_to = l_coal_range_to THEN
      l_coal_range_to := l_next_range_to;
    ELSE
      l_coal_range_from := l_range_from;
      l_coal_range_to := l_next_range_to;
    END IF;

    --coalesce two leaves - update range on one leaf
    IF k_testmode THEN
      debug_msg(l_range_from||'-'||l_range_to||' + '||l_next_range_from||'-'||l_next_range_to||' => '||l_coal_range_from||'-
'||l_coal_range_to,4);
    ELSE
      debug_msg(l_range_from||'-'||l_range_to||' + '||l_next_range_from||'-'||l_next_range_to||' => '||l_coal_range_from||'-
'||l_coal_range_to,6);

      UPDATE pstreeleaf[15]
      SET    range_from = l_coal_range_from
      ,      range_to = l_coal_range_to
      WHERE  setid = p_setid
      AND    setcntrlvalue = p_setcntrlvalue
      AND    tree_name = p_tree_name
      AND    effdt = p_effdt
      AND    tree_node_num = l_tree_node_num
      AND    range_from = l_next_range_from
      AND    range_to = l_next_range_to
      AND    tree_branch = l_tree_branch;
      debug_msg(SQL%ROWCOUNT||' rows updated',7);

      --delete the other leaf
      DELETE pstreeleaf
      WHERE  setid = p_setid
      AND    setcntrlvalue = p_setcntrlvalue
      AND    tree_name = p_tree_name
      AND    effdt = p_effdt
      AND    tree_node_num = l_tree_node_num
      AND    range_from = l_coal_range_from
      AND    range_to = l_range_to
      AND    tree_branch = l_tree_branch;
      debug_msg(SQL%ROWCOUNT||' rows deleted',7);
    END IF;
  END LOOP;

  IF NOT k_testmode AND l_coal_count > 0 THEN
    UPDATE pstreedefn d[16]
    SET    d.version = (SELECT version FROM pslock WHERE objecttypename = 'TDM')
    ,      d.lastupdoprid = k_oprid
    ,      d.lastupddttm = SYSTIMESTAMP
    ,      d.leaf_count = (SELECT COUNT(*)
```

[14] When the first change on a tree is detected, the version number for tree definitions is incremented.

[15] The script merges two leaves into one. One is updated, the other deleted.

[16] The tree definition is updated at the end with the new version number, and the new number of leaves on the tree. The number of nodes on the tree is unchanged.

```
                        FROM   pstreeleaf l
                        WHERE  l.setid = p_setid
                        AND    l.setcntrlvalue = p_setcntrlvalue
                        AND    l.tree name = p tree name
                        AND    l.effdt = p effdt)
    WHERE  d.setid = p setid
    AND    d.setcntrlvalue = p setcntrlvalue
    AND    d.tree name = p tree name
    AND    d.effdt = p_effdt;
  END IF;

  CLOSE c treeleaf;
  debug msg(l coal count||' leaves coalesced ('||LTRIM(TO CHAR(100*l coal count/NULLIF(l leaf count,0),'90'))||'%)',3);
  unset action(p action name=>l action);
EXCEPTION
  WHEN no structure found THEN
    debug msg('Skipping tree',3);
    unset_action(p_action_name=>l_action);
END one_tree;
-----------------------------------------------------------------------------------------------
BEGIN
  dbms application info.read module(module name=>l module, action name=>l action);
  dbms application info.set module(module name=>'LEAFCOALESCE', action name=>'MAIN');
  FOR i IN (17
    SELECT DISTINCT d.setid, d.setcntrlvalue, d.tree_name, d.effdt
    FROM   pstreeselctl d
/*  FROM   pstreedefn d
    ,      pstreestrct s
    ,      psrecfielddb f
    WHERE  d.tree_strct_id = s.tree_strct_id
    AND    s.node_fieldname = 'TREE_NODE'
--  AND    d.TREE_ACC_METHOD = 'L' --literal values
    AND    s.dtl recname = f.recname
    AND    s.dtl fieldname = f.fieldname*/
--  AND    tree name = 'NAME OF TREE'
  ) LOOP
    one tree(i.setid, i.setcntrlvalue, i.tree name, i.effdt);
    debug msg(' ',4);
  END LOOP;
  unset_action(p_action_name=>l_action);
  IF k_testmode THEN
    debug msg('Running in Test Mode - No Update',1);
  ELSE
    debug msg('Commit changes or rollback',1);
  END IF;
END;
/
spool off
set termout on
```

The spool file output reports on the number of leaves coalesced. If the script does update the PeopleTools tree tables, it does not commit the update. It is left for the user to commit or rollback.

```
.(3)Processing SHARE, ,XXX ACCOUNT,141201
.(4)634 nodes, 2636 leaves
.(4)1358 leaves coalesced (52%)
…
(1)Commit changes or rollback
```

[17] This query identifies the trees to be processed. It can be changed as necessary. For example, you might process

- specific trees,
- most recent effective dated trees
- trees with literal values performance option
- trees with a tree structure record.

Clearing Selector Control Records for which there is no Tree Definition

1. This statement deletes rows from PSTREESELCTL for which no tree is defined in PSTREEDEFN.

```
DELETE FROM   pstreeselctl l
WHERE NOT EXISTS(
      SELECT 'x'
      FROM   pstreedefn d
      WHERE  d.setid = l.setid
      AND    d.setcntrlvalue = l.setcntrlvalue
      AND    d.tree_name = l.tree_name
      AND    d.effdt = l.effdt)
/
```

The next script that clears dynamic selectors from the PSTREESELECT tables will also clear any rows orphaned by this delete.

Clear Dynamic Selectors from Tree Selector Tables

2. This script deletes all rows from all of the PSTREESELECT tables which relate for which there is no corresponding data in PSTREESELCTL because they are dynamic selectors.
    d. **Action:** Run treeselectorclup2.sql in SQL*Plus connected as SYSADM.
    e. There is no need to be able reverse this. Rows relating to dynamic selectors can be left in the tree selector tables when an nVision crashes without cleaning up after itself. Clearing debris from the tree selector tables is something that should be done regularly regardless. In a system that still uses dynamic selector I would suggest doing this daily.

```
REM treeselectorclup2.sql
set termout on head off echo off feedback off timi off trimspool on trimout on pages 0
serveroutput on lines 200
spool treeselectorclup2

DECLARE
  l_sqlc CLOB;
  l_sqld CLOB;
  l_numrows INTEGER;
  l_numsels INTEGER;
BEGIN
  FOR i IN (
    SELECT table_name
    FROM   user_tables
    where table_name LIKE 'PSTREESELECT__'
    ORDER BY 1
  ) LOOP
    l_sqlc := 'SELECT COUNT(DISTINCT selector_Num), COUNT(*) FROM '||i.table_name;
    l_sqld := 'DELETE FROM '||i.table_name||' t WHERE NOT t.selector_num IN(SELECT
DISTINCT selector_num FROM pstreeselctl)';

    EXECUTE IMMEDIATE l_sqlc INTO l_numsels, l_numrows;
    dbms_output.put_line('Table '||i.table_name||':'||l_numsels||' sectors,'||l_numrows||'
rows');

    IF l_numrows > 0 THEN
      dbms_output.put_line('SQL:'||l_sqld);
      EXECUTE IMMEDIATE l_sqld;
      dbms_output.put_line(SQL%ROWCOUNT||' rows deleted.');

      EXECUTE IMMEDIATE l_sqlc INTO l_numsels, l_numrows;
      dbms_output.put_line('Table '||i.table_name||':'||l_numsels||'
sectors,'||l_numrows||' rows');

      COMMIT;
    END IF;
  END LOOP;
END;
/
spool off
show errors
```

The script produces a report of what it did. This is typical output.

```
Table PSTREESELECT01:0 sectors,0 rows
Table PSTREESELECT02:0 sectors,0 rows
Table PSTREESELECT03:1 sectors,85 rows
SQL:DELETE FROM PSTREESELECT03 t WHERE NOT t.selector_num IN(SELECT DISTINCT
selector_num FROM pstreeselctl)
0 rows deleted.
Table PSTREESELECT03:1 sectors,85 rows
Table PSTREESELECT04:0 sectors,0 rows
Table PSTREESELECT05:2 sectors,117 rows
SQL:DELETE FROM PSTREESELECT05 t WHERE NOT t.selector_num IN(SELECT DISTINCT
selector_num FROM pstreeselctl)
0 rows deleted.
Table PSTREESELECT05:2 sectors,117 rows
Table PSTREESELECT06:13 sectors,2385 rows
SQL:DELETE FROM PSTREESELECT06 t WHERE NOT t.selector_num IN(SELECT DISTINCT
selector_num FROM pstreeselctl)
0 rows deleted.
Table PSTREESELECT06:13 sectors,2385 rows
Table PSTREESELECT07:0 sectors,0 rows
Table PSTREESELECT08:328 sectors,87798 rows
SQL:DELETE FROM PSTREESELECT08 t WHERE NOT t.selector_num IN(SELECT DISTINCT
selector_num FROM pstreeselctl)
26633 rows deleted.
Table PSTREESELECT08:133 sectors,61165 rows
Table PSTREESELECT09:0 sectors,0 rows
Table PSTREESELECT10:450 sectors,66891 rows
```

```
SQL:DELETE FROM PSTREESELECT10 t WHERE NOT t.selector_num IN(SELECT DISTINCT
selector_num FROM pstreeselctl)
45376 rows deleted.
Table PSTREESELECT10:74 sectors,21515 rows
Table PSTREESELECT11:0 sectors,0 rows
Table PSTREESELECT12:0 sectors,0 rows
Table PSTREESELECT13:0 sectors,0 rows
Table PSTREESELECT14:0 sectors,0 rows
Table PSTREESELECT15:0 sectors,0 rows
Table PSTREESELECT16:0 sectors,0 rows
Table PSTREESELECT17:0 sectors,0 rows
Table PSTREESELECT18:0 sectors,0 rows
Table PSTREESELECT19:0 sectors,0 rows
Table PSTREESELECT20:3 sectors,130 rows
SQL:DELETE FROM PSTREESELECT20 t WHERE NOT t.selector num IN(SELECT DISTINCT
selector num FROM pstreeselctl)
104 rows deleted.
Table PSTREESELECT20:1 sectors,26 rows
Table PSTREESELECT21:0 sectors,0 rows
Table PSTREESELECT22:0 sectors,0 rows
Table PSTREESELECT23:0 sectors,0 rows
Table PSTREESELECT24:0 sectors,0 rows
Table PSTREESELECT25:0 sectors,0 rows
Table PSTREESELECT26:0 sectors,0 rows
Table PSTREESELECT27:0 sectors,0 rows
Table PSTREESELECT28:0 sectors,0 rows
Table PSTREESELECT29:0 sectors,0 rows
Table PSTREESELECT30:0 sectors,0 rows
```

Maintaining Statistics on Tree Selector Tables

First Time Generation of Statistics

Script *treeselect_stat_prefs.sql* sets table preferences on all 30 tree selector tables.

- Histograms are collected on SELECTOR_NUM and TREE_NODE_NUM. Oracle may choose to collect histograms on other columns if it judges it appropriate.
- have also created extended statistics on the combination of SELECTOR_NUM and TREE_NODE_NUM
- Statistics on the tree selector tables are locked to prevent the overnight statistics job from updating them.

```
REM treeselect_stat_prefs.sql
REM (c)2013 Go-Faster Consultancy Ltd.
REM histograms

spool treeselect stat perfs
set serveroutput on


REM set table prefs and unlock stats
BEGIN
 FOR i IN (
  SELECT table name FROM user Tables WHERE table name like 'PSTREESELECT  '
 ) LOOP
  dbms_stats.lock_table_stats
  (ownname=>user
  ,tabname=>i.table_name
  );[18]
  dbms output.put line('Table:'||i.table name);
  dbms_stats.set_table_prefs
  (ownname=>user
  ,tabname=>i.table_name
  ,pname=>'METHOD_OPT'
--,pvalue=>'FOR ALL COLUMNS SIZE AUTO FOR COLUMNS SIZE 254 SELECTOR_NUM TREE_NODE_NUM'
  ,pvalue=>'FOR ALL COLUMNS SIZE 1 FOR COLUMNS SIZE 254 SELECTOR NUM'[19]
  dbms_stats.set_table_prefs
  (ownname=>user
  ,tabname=>i.table_name
  ,pname=>'STALE_PERCENT'
  ,pvalue=>'1' --1pct stale threshold to force regular stats refresh by the maintenance
job
  );
END LOOP;
END;
/

REM create extended stats[20]
DECLARE
 l_clob CLOB;
 e_extension_exists EXCEPTION;
 PRAGMA EXCEPTION_INIT(e_extension_exists,-20007);
BEGIN
 FOR i IN (
  SELECT table name FROM user Tables
  where table name like 'PSTREESELECT  '
) LOOP
  BEGIN
    l_clob:=sys.dbms_stats.create_extended_stats
    (ownname=>user
    ,tabname=>i.table_name
    ,extension=>'(SELECTOR_NUM, TREE_NODE_NUM)'
    );
    dbms_output.put_line('Extended stats:'||i.table_name||':'||l_clob);
   EXCEPTION
     WHEN e_extension_exists THEN
       dbms_output.put_line('Extended stats already exist on :'||
                            i.table name||':'||l clob);
   END;
 END LOOP;
END;
/
```

[18] Statistics on the tree selectors are locked because a manual adjustment is made to two densities immediately after stats are calculated.

[19] Note that histograms are only built on SELECTOR_NUM which should have a frequency histogram.

[20] Extended statistics are collected on the combination of SELECTOR_NUM and TREE_NODE_NUM.

```
REM gather stats
BEGIN
 FOR i IN (
  SELECT table name, SUBSTR(table name,-2) suffix
  FROM user Tables WHERE table name like 'PSTREESELECT  '
 ) LOOP
  dbms_output.put_line('Table Stats:'||i.table_name);
  sys.dbms_stats.gather_table_stats
  (ownname=>user
  ,tabname=>i.table_name
  ,force=>TRUE
  );
  --set density to 1 for range from[21]
  dbms_stats.set_column_stats
  (ownname=>user
  ,tabname=>i.table_name
  ,colname=>'RANGE FROM '||i.suffix
  ,density=>1
  ,force=>TRUE);
  dbms_stats.set_column_stats
  (ownname=>user
  ,tabname=>i.table_name
  ,colname=>'RANGE_TO_'||i.suffix
  ,density=>1
  ,force=>TRUE);
 END LOOP;
END;
/


select table name, stattype locked, num Rows, last analyzed, stale Stats
from dba_tab_statistics
where table_name like 'PSTREESELECT__'
order by 1
/

set pages 99 lines 200 trimspool on
break on table_name skip 1
select   s.*, e.extension
from dba_tab_Col_statistics s
         left outer join dba_stat_extensions e
         on e.owner = s.owner
         and e.table name = s.table name
         and e.extension name = s.column name
--where table_name = 'PS_LEDGER'
where s.table_name LIKE 'PSTREESELECT__'
and s.num_distinct > 0
ORDER by 1,2,3
/

spool off
```

treeselect_stat_prefs.sql

[21] After collecting statistics on the PSTREESELECTnn tables the density on the RANGE_FROM_nn and RANGE_TO_nn columns is set to 1. This prevents Oracle from under estimating the cost of looking up this table which can occur when the LEDGER table is joined before the PSTREESELECTnn table.

Implementation

3. **Action:** Run *treeselect_stat_perfs.sql* in SQL*Plus connected as SYSADM.

The script produces a report of what it has updated (not shown) and the statistics that have been collected (shown below)

```
OWNER    TABLE NAME        COLUMN NAME                    NUM DISTINCT LOW VALUE           HIGH VALUE             DENSITY  NUM NULLS NUM BUCKETS LAST ANAL SAMPLE SIZE GLO USE
-------- ----------------- ------------------------------ ------------ ------------------- ------------------- ---------- ---------- ----------- --------- ----------- --- ---
NOTES                          AVG COL LEN HISTOGRAM       SCOPE   EXTENSION
------------------------------ ----------- --------------- ------- ------------------------------------------------------------------------------------------
SYSADM   PSTREESELECT03    RANGE FROM 03                            85 414257              5A4D42                1                0           1 13-JAN-16          85 YES YES
                                       4 NONE              SHARED

SYSADM                     RANGE TO 03                              85 414257              5A4D42                1                0           1 13-JAN-16          85 YES YES
                                       4 NONE              SHARED

SYSADM                     SELECTOR NUM                              1 C30F361A            C30F361A              .005882353       0           1 13-JAN-16          85 YES NO
                                       5 FREQUENCY         SHARED

SYSADM                     SYS_STUSRD6Y1KXDCW#4RDM0NN3A4I            9 CA051413330F47455C25 CA112712290147 2b5515 .111111111       0           1 13-JAN-16          85 YES NO
                                                                      29                     55
                                      12 NONE              SHARED  ("SELECTOR NUM","TREE NODE NUM")

SYSADM                     TREE NODE NUM                             9 C5134C              C514644C3B3C          .111111111       0           1 13-JAN-16          85 YES NO
                                       6 NONE              SHARED
```

On-Going Maintenance of Statistics on Tree Selector Tables

Statistics on the tree selectors are locked because the density was manually reset, so a process needs to be put in place to maintain the statistics as the static selectors are populated when trees are used for the first time.

PeopleSoft inserts a row into PSTREESELCTL for each statistic selector. A database trigger can be used to submit a database job to gather statististics in the same manner as the previous script.

Action: Implement trigger by running *pstreeselector_stats.sql* in SQL*Plus connected as SYSADM.

```
rem pstreeselector_stats.sql
rem trigger to update stats on selector table as pstreeselctl is populated.
rem NB stats job is only fired on commit.
set echo on feedback on verify on termout on
spool pstreeselector_stats
ROLLBACK;

CREATE OR REPLACE TRIGGER sysadm.pstreeselector_stats
BEFORE INSERT OR UPDATE ON sysadm.pstreeselctl
FOR EACH ROW
DECLARE
  l_jobno      NUMBER;
  l_cmd        VARCHAR2(1000);
  l_table_name VARCHAR2(18);
  l_suffix     VARCHAR2(2);
BEGIN
  l table name := 'PSTREESELECT'||LTRIM(TO CHAR(:new.length,'00'));
  l suffix     := SUBSTR(l table name,-2);
  l_cmd := 'dbms_stats.gather_table_stats(ownname=>user,tabname=>'''
||l_table_name||''',force=>TRUE);'
||'dbms_stats.set_column_stats(ownname=>user,tabname=>'''
||l table name||''',colname=>''RANGE FROM '||l suffix||''',density=>1,force=>TRUE);'
||'dbms stats.set column stats(ownname=>user,tabname=>'''
||l_table_name||''',colname=>''RANGE_TO_'||l_suffix||''',density=>1,force=>TRUE);';
  dbms_output.put_line(l_cmd);
  dbms_job.submit(l_jobno,l_cmd);
EXCEPTION WHEN OTHERS THEN NULL;
END;
/

set serveroutput on
show errors

rollback;[22]
alter session set nls date format='hh24:mi:ss dd.mm.yyyy';
INSERT INTO sysadm.pstreeselctl VALUES('GFC',' ','GFC TEST',sysdate,42,42,sysdate,'R',1);
commit;
delete from sysadm.pstreeselctl where setid = 'GFC';
commit;
pause

select table_name, num_rows, last_analyzed from user_tables where table_name like
'PSTREESELECT  ' order by 1,2,3;
select table_name, column_name, num_distinct, density from user_tab_columns where
table_name = 'PSTREESELECT01';

spool off
```

pstreeselector_stats.sql

[22] This part of the script is a test. It is necessary to pause for a few seconds to allow time for the job to run, before running the queries to see if the stats have been updated correctly on PSTREESELECT01 (which is not used by the application).

```
SQL> INSERT INTO sysadm.pstreeselctl VALUES('GFC',' ','GFC_TEST',sysdate,42,42,sysdate,'R',1);[23]
dbms_stats.gather_table_stats(ownname=>user,tabname=>'PSTREESELECT01',force=>TRUE);
dbms_stats.set_column_stats(ownname=>user,tabname=>'PSTREESELECT01',colname=>
'RANGE FROM 01',density=>1,force=>TRUE);
dbms stats.set column stats(ownname=>user,tabname=>'PSTREESELECT01',colname=>'RANGE TO 01',density=>1,force=>TRUE);
…
SQL> select table_name, column_name, num_distinct, density from user_tab_columns where
table_name = 'PSTREESELECT01';

TABLE_NAME                     COLUMN_NAME                    NUM_DISTINCT    DENSITY
------------------------------ ------------------------------ ------------ ----------
PSTREESELECT01                 SELECTOR_NUM                              0          0
PSTREESELECT01                 TREE_NODE_NUM                             0          0
PSTREESELECT01                 RANGE_FROM_01                             0          1[24]
PSTREESELECT01                 RANGE_TO_01                               0          1
```

[23] When an insert or update is made to PSTREESELCTL the trigger submits a PL/SQL command string to the job scheduler and also prints the command out to the standard output channel. In nVision this output is simply lost, but can be seen when in SQL*Plus with serveroutput enabled

[24] Note that the density is 1 even though the number of distinct rows is 0. Normally density would be 0 in this case.

Database Partitioning of Ledger tables

Database partitioning is not an essential part of the nVision tuning solution, but it is extremely effective. It allows the database to eliminate partitions that cannot contain data of interest without having to scan them.

Most of the large queries in nVision are on the PS_LEDGER and PS_LEDGER_BUDG tables. They always apply to a specific finanacial year, and often to a single accounting period or they year to data. Therefore, these queries always contain an equi-join on the column FISCAL_YEAR, and will contain either an equi-join or between clause on ACCOUNTING_PERIOD. For example

```
SELECT A.ACCOUNT,SUM(A.POSTED TOTAL AMT)
FROM PS LEDGER A, PSTREESELECT05 L1
WHERE A.LEDGER='ACTUALS'
AND A.FISCAL_YEAR=2016 AND
A.ACCOUNTING_PERIOD=3
AND …
```

A single period query

```
SELECT A.ACCOUNT,SUM(A.POSTED TOTAL AMT)
FROM PS LEDGER A, PSTREESELECT05 L1
WHERE A.LEDGER='ACTUALS'
AND A.FISCAL YEAR=2015 AND
A.ACCOUNTING_PERIOD BETWEEN 1 AND 9
AND …
```

A year-to-date query

These queries will always benefit on from partitioning on FISCAL_YEAR. There is often some benefit to partitioning on ACCOUNTING_PERIOD, particularly the queries that compare with the same period to date in the pervious fiscal year.

Oracle can only partition in two dimensions.

Range Partitioning

I always recommend range partitioning the PS_LEDGER and PS_LEDGER_BUG tables in a single dimension on the combination of FISCAL_YEAR and ACCOUNTING PERIOD to produce single period partitions in the current and previous fiscal year. There should be a separate partition for period 0, which is used to bring entries forward from the previous year, and another partition for periods 998 and 999 which are used to carry entries forward to the next year.

I would create the use something along the following lines to recreate the tables (comments in footnotes)

```
CREATE TABLE ps ledger
(business unit VARCHAR2(5) NOT NULL
,ledger VARCHAR2(10) NOT NULL
…
)
TABLESPACE GLLARGE
PCTFREE 10 PCTUSED 80
PARTITION BY RANGE(FISCAL YEAR,ACCOUNTING PERIOD)[25]
(PARTITION ledger 2012 VALUES LESS THAN (2013,0) PCTFREE 0[26] COMPRESS[27]
,PARTITION ledger_2013 VALUES LESS THAN (2014,0) PCTFREE 0 COMPRESS
,PARTITION ledger_2014 VALUES LESS THAN (2015,0)[28] PCTFREE 0 COMPRESS
,PARTITION ledger 2015 bf VALUES LESS THAN (2015,1) PCTFREE 0 COMPRESS
…
,PARTITION ledger 2015 11 VALUES LESS THAN (2015,12)
,PARTITION ledger 2015 12 VALUES LESS THAN (2015,13)
,PARTITION ledger_2015_cf VALUES LESS THAN (2016,0)[29]
,PARTITION ledger_2016_bf VALUES LESS THAN (2016,1)
…
,PARTITION ledger 2016 12 VALUES LESS THAN (2016,13)
,PARTITION ledger 2016 cf VALUES LESS THAN (2017,0)
)
ENABLE ROW MOVEMENT
PARALLEL
NOLOGGING
/
```

---

[25] This is a single range partition but on the combination of two columns. This only works because there is never a predicate on ACCOUNTING_PERIOD without there also being a predicate on FISCAL_YEAR. It is the combination of these two columns that identify a single accounting period.

[26] Note that PCTFREE is set to 0 on these paratitions because the data is static, there is no need to reserve free space for updates.

[27] Simple compression has been specified so that when the table is rebuilt the data is compressed. This is only applied to historical partitions, because it will not apply to rows inserted in convential path mode by parts of the Oracle application. If you are licenced for other forms of compression you could employ them on all partitions, and different levels of compression on different partitons.

[28] There is no need to create single period partitions for older fiscal years. This data is static and is usually rarely accessed. Here, data for fiscal year 2014 is going to be in a partition called LEDGER_2014, and all the data values are less than 2015. Each year, as a maintenance task the partitions for the oldest year with periodic partitioning could be merged into a single partition, They could also be compressed during that operation.

[29] Single period partitions have been defined for the current and previous fiscal year.

Sub-partitioning

It is possible to sub-partition tables in Oracle. This is partitioning in a separate dimensions. Partitions can be eliminated independently in each dimension. Partitioning was introduced in 8i. Range-hash subpartitioning was available in 8i, range-list was introduced in 10g, and range-range partitioning in 11g.

Some, but not all, PeopleSoft GL systems will benefit from sub-partitioning. The partitioning column may also vary. It depends upon the data and the data volumes. There is no point having 90% of the data in one partition.

For example, a customer with several ledgers created separate list sub-partitions for each of the larger partitions.

```
CREATE TABLE ps_ledger
(business_unit VARCHAR2(5) NOT NULL
,ledger VARCHAR2(10) NOT NULL
...)
TABLESPACE GLLARGE
PCTFREE 10 PCTUSED 80
PARTITION BY RANGE(FISCAL_YEAR,ACCOUNTING_PERIOD)
SUBPARTITION BY LIST (LEDGER)
(PARTITION ledger_2012 VALUES LESS THAN (2013,0) PCTFREE 0 COMPRESS
(SUBPARTITION ledger 2012 act eur
 VALUES ('ACT EUR')
,SUBPARTITION ledger_2012_act_usd
 VALUES ('ACT_USD')
,SUBPARTITION ledger_2012_z_others
 VALUES (DEFAULT)
)
…
```

At another customer where different regions processed their data at different times with different batch and reporting processs, list sub-partitions were proposed for the largest business units in each region.

```
CREATE TABLE ps ledger
(business unit VARCHAR2(5) NOT NULL
,ledger VARCHAR2(10) NOT NULL
...)
TABLESPACE GLLARGE
PCTFREE 10 PCTUSED 80
PARTITION BY RANGE(FISCAL_YEAR,ACCOUNTING_PERIOD)
SUBPARTITION BY LIST (BUSINESS UNIT)
(PARTITION ledger 2012 VALUES LESS THAN (2013,0) PCTFREE 0 COMPRESS
(SUBPARTITION ledger 2012 bu eur
 VALUES ('BEL01','NLD01','FRA01')
,SUBPARTITION ledger_2012_bu_usa
 VALUES ('NY001','CA001','TX001',…)
,SUBPARTITION ledger 2012 z others
 VALUES (DEFAULT)
)
…
```

Tuning nVision Without Partitioning

The partitioning option is licenced option of Oracle Enterprise Edition. So it costs money.

The advantage of partitioning the ledger tables is that you can eliminate historical data, i.e. previous fiscal years, without having to scan them. As data builds up over time this saving is more significant. If you can't partition those tables you will have to scan through that historical data every time you query the data.

It is also common to have a long narrower tail of data for previous fiscal years due to migration. This skew can cause Oracle to miscalculate execution plans.

Even with partitioning it is important to purge unneeded fiscal years out of the LEDGER and LEDGER_BUG tables, but without it is essential[30].

[30] Business users are almost universally averse to letting go of data, and most countries have legislation that requires retention of data for several years. I always say that I am happy for the data to be retained in the database, just not in the LEDGER and LEDGER_BUG tables. These are for recent data that is regularly in use.

SQL Outlines/Baselines/Profiles/Patches

I don't normally recommend any of the Oracle plan stability technologies in conjuction with nVision due to the dynamic nature of the SQL. However, in some very limited cases, it is the only way I have found to consistently control the execution plan of this statement.

Additional nVision Monitoring with Fine Grain Audit

I use Oracle Fine Grain Audit (FGA) to capture access to certain nVision control tables in order to:

- Set the session's ACTION attribute using DBMS_APPLICATION_INFO[31] to the values on the NVS_REPORT table. That value will visible on v$session and will be captured by AWR and ASH.
- Identify the report that is running and whether it is using dynamic selectors[32]. This can identify layouts with tree performance overrides.

It should be implemented as follows:

Granted Privileges

- **Action:** Explicitly grant (not via a role) select privilege on audit table to SYSADM

```
GRANT SELECT ON sys.fga_log$ TO SYSADM;
GRANT EXECUTE ON sys.dbms_fga TO SYSADM;
```

Audit Handler Procedure

- **Action:** Script fga_hander.sql (called from fga_audit_create.sql) will create a PL/SQL package to act as an error handler for the audit policy on PS_NVS_REPORT.
  - It will extract the business unit and name of the report from the third and forth parameters in the audit data and set the action of the session accordingly[33].

```
REM fga_handler.sql
spool fga_handler
REM requires psftapi[34]

REM requires explicit
GRANT SELECT ON sys.fga log$ TO SYSADM;
```

[31] Use of this package does not require any special licence.

[32] Acknowledgements to Tyler McPheeters for the original idea and implementation.

[33] Although this might change from customer to customer, and some further customisation may be required.

[34] Available from http://www.go-faster.co.uk/scripts.htm#psftapi.sql

```
CREATE OR REPLACE PROCEDURE sysadm.aeg_fga_nvision_handler
(object_schema VARCHAR2
,object_name   VARCHAR2
,policy name   VARCHAR2)
AS
  l_sqlbind VARCHAR2(4000);
  l_parm1   VARCHAR2(30);
  l_parm2   VARCHAR2(30);
  l_parm3   VARCHAR2(30);
  l_parm4   VARCHAR2(30);
BEGIN
  BEGIN
    SELECT x.lsqlbind
    ,      SUBSTR(x.lsqlbind,x.start1,NVL(x.end1,x.lensqlbind+1)-x.start1) parm1
    ,      SUBSTR(x.lsqlbind,x.start2,NVL(x.end2,x.lensqlbind+1)-x.start2) parm2
    ,      SUBSTR(x.lsqlbind,x.start3,NVL(x.end3,x.lensqlbind+1)-x.start3) parm3
    ,      SUBSTR(x.lsqlbind,x.start4,NVL(x.end4,x.lensqlbind+1)-x.start4) parm4
    INTO   l sqlbind, l parm1, l parm2, l parm3, l parm4
    FROM   (
      SELECT l.*
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,1,1,'i'),0) start1
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,2,0,'i'),0) end1
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,2,1,'i'),0) start2
      ,      NULLIF(REGEXP INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,3,0,'i'),0) end2
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,3,1,'i'),0) start3
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,4,0,'i'),0) end3
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,4,1,'i'),0) start4
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,5,1,'i'),0) end4
      ,      LENGTH(lsqlbind) lensqlbind
      FROM   sys.fga log$ l
    ) x
    WHERE  x.sessionid = USERENV('SESSIONID')
    AND    x.entryid   = USERENV('ENTRYID')
    AND    x.obj$name  = 'PS_NVS_REPORT';
  EXCEPTION
    WHEN no_data_found THEN
      RAISE_APPLICATION_ERROR(-20000,'AEG_FGA_NVISION_HANDER: No Audit Row');
  END;

  IF l_parm4 IS NULL THEN
    l_parm4 := l_parm3;
    l_parm3 := l_parm2;
    l parm2 := l parm1;
  END IF;

  IF l_parm4 IS NULL THEN
    l_parm4 := l_parm3;
    l_parm3 := l_parm2;
  END IF;

  IF l_parm4 IS NULL THEN
    l_parm4 := l_parm3;
  END IF;

  dbms_output.put_line(l_sqlbind);
  dbms output.put line(l parm1);
  dbms output.put line(l parm2);
  dbms output.put line(l parm3);
  dbms_output.put_line(l_parm4);


dbms_application_info.set_action(SUBSTR('PI='||psftapi.get_prcsinstance()||':'||l_parm4||'
:'||l parm3,1,64));
--EXECUTE IMMEDIATE 'ALTER SESSION SET
TRACEFILE_IDENTIFIER=''PI='||psftapi.get_prcsinstance()||':'||l_parm4||':'||l_parm3||''''';
35
END;
/
show errors
```

[35] It could also set the name of the session trace file to the same value as action. That is currently commented out because even if not using trace, it can generate lots of small trace files with just header information.

```
rem this is a test - expected output[36]
rem ERROR at line 1:
rem ORA-20000: AEG FGA NVISION HANDER: No Audit Row
rem ORA-06512: at "SYSADM.AEG_FGA_NVISION_HANDLER", line 28
rem ORA-06512: at line 1
exec aeg_fga_nvision_handler('SYSADM','PS_NVS_REPORT','PS_NVS_REPORT_SEL');
spool off
```

[36] The script includes a test of the error handle. It will produce an error message similar to that in the comments in the script. However, if the privileges on the FGA table and package are not in place you will get other errors.

Audit Policies

- **Action:** Script fga_audit_create.sql will implement FGA policies on
  - PS_NVS_REPORT. Audit Select. To capture name of report.
  - PSTREESELNUM. Audit Select. To detect allocation of new selector numbers when tree selector tables are maintained.
  - PSTREESELECT*nn* (where *nn* between 01 and 30). Audit insert and delete on tree selector tables. To detect use of dynamic selectors and introduction of new trees.

```
REM fga_audit_create.sql
REM (c)2016 David Kurtz
REM (c)2014 T McPheeters
set echo on serveroutput on
spool fga_audit_create

/*************************************************************
 * track population/delete of tree selectors.
 * detect dynamic selectors, detect dynamic selectors that were not deleted.
/*************************************************************/
BEGIN
  FOR i IN(
    SELECT r.recname, t.table_name
    FROM   psrecdefn r
    ,      user_tables t
    WHERE  r.rectype = 0
    AND    t.table name = DECODE(r.sqltablename,' ','PS ')||r.recname
    AND  (        r.recname like 'PSTREESELECT__')
    AND NOT EXISTS(
           SELECT 'x'
           FROM   user audit policies p
           WHERE  p.object name = t.table name
           AND    p.policy name = r.recname)
    ORDER BY 1
  ) LOOP
    dbms_output.put_line('Creating FGA Policy '||i.recname||' on insert and delete on
'||i.table_name);
    DBMS_FGA.ADD_POLICY(
     object schema      => 'SYSADM',
     object name        => i.table name,
     policy_name        => i.recname,
     enable             => TRUE,
     statement_types    => 'INSERT, DELETE',
     audit trail        => DBMS FGA.DB + DBMS FGA.EXTENDED);
  END LOOP;
END;
/

/*************************************************************
 * track allocation of new selector num.  Static selectors appear in PSTREESELCTL.
/*************************************************************/
BEGIN
  DBMS_FGA.ADD_POLICY(
   object_schema      => 'SYSADM',
   object_name        => 'PSTREESELNUM',
   policy_name        => 'PSTREESELNUM',
   enable             =>  TRUE,
   statement types    => 'SELECT',
   audit trail        =>  DBMS FGA.DB + DBMS FGA.EXTENDED);
END;
/


/*************************************************************
 * This policy logs query on the control table from which nVision parameters are read
 * The error handle is used to name the oracle trace file.  NB creates 0 length file even
if trace not invoked
/*************************************************************/
BEGIN
  DBMS FGA.ADD POLICY(
   object schema      => 'SYSADM',
   object name        => 'PS NVS REPORT',
   policy_name        => 'PS_NVS_REPORT_SEL',
   handler_module     => 'AEG_FGA_NVISION_HANDLER'
   enable             =>  TRUE,
   statement_types    => 'SELECT',
   audit_trail        =>  DBMS_FGA.DB + DBMS_FGA.EXTENDED);
END;
/

/*************************************************************
 * This policy is used to log LEDGER queries
/*************************************************************
BEGIN
  DBMS_FGA.ADD_POLICY(
```

```
   object_schema      => 'SYSADM',
   object_name        => 'PS_LEDGER',
   policy_name        => 'PS_LEDGER_NVS',
   enable             =>  TRUE,
   statement types    => 'SELECT',
   audit_condition    => 'STATISTICS_CODE = '' ''',
   audit_trail        =>  DBMS_FGA.DB + DBMS_FGA.EXTENDED);
END;
/
/************************************************************/
spool off
```

Audit Archive/Purge

The audit records need to be archived and purged. I would also recommend setting the audit purge job as described in the Oracle Database Security Guide, 23: Administering the Audit Trail (https://docs.oracle.com/database/121/DBSEG/audit_admin.htm#DBSEG1026). I suggest matching the retention period to that of the ASH data.

I would suggest moving SYS.FGA_LOG$ out of the SYSTEM tables to another tablespace[37].

```
BEGIN
  DBMS_AUDIT_MGMT.set_audit_trail_location(
    audit_trail_type           => DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD,
    audit_trail_location_value => 'AUDIT AUX');
END;
/
```

[37] See https://oracle-base.com/articles/11g/auditing-enhancements-11gr2

Detecting Use of Dynamic Selectors

When nVision runs with dynamic selectors it populate PSTREESELECT. This triggers an FGA. The following query reports the FGA on PSTREESELECT and determines the report by finding the corresponding FGA for select on PS_NVL_REPORT

```
set pages 99 lines 200 long 4000
column policyname format a20
column oshst format a30
column parm2 format a20
column clientid format a20
column lsqltext format a60
column lsqlbind format a30
column stmt_type format a8
column ntimestamp# format a25
column mintimestamp format a25
column maxtimestamp format a25
with x as (
      SELECT l.*
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,1,1,'i'),0) start1
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,2,0,'i'),0) end1
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,2,1,'i'),0) start2
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,3,0,'i'),0) end2
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,3,1,'i'),0) start3
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,4,0,'i'),0) end3
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,4,1,'i'),0) start4
      ,      NULLIF(REGEXP_INSTR(lsqlbind,' #[0-9]+\([0-9]+\)\:',1,5,0,'i'),0) end4
      ,      LENGTH(lsqlbind) lensqlbind
      FROM   sys.fga_log$ l
), y as (
    SELECT x.*
    ,      CAST(SUBSTR(x.lsqlbind,x.start1,NVL(x.end1,x.lensqlbind+1)-x.start1) AS VARCHAR2(32)) parm1
    ,      CAST(SUBSTR(x.lsqlbind,x.start2,NVL(x.end2,x.lensqlbind+1)-x.start2) AS VARCHAR2(32)) parm2
    ,      CAST(SUBSTR(x.lsqlbind,x.start3,NVL(x.end3,x.lensqlbind+1)-x.start3) AS VARCHAR2(32)) parm3
    ,      CAST(SUBSTR(x.lsqlbind,x.start4,NVL(x.end4,x.lensqlbind+1)-x.start4) AS VARCHAR2(32)) parm4
    FROM   x
)
select DISTINCT y.parm2
,      a.policyname
,      MIN(a.ntimestamp#) mintimestamp
,      MAX(a.ntimestamp#) maxtimestamp
--,      DECODE(a.stmt_type,1,'SELECT',2,'INSERT',8,'DELETE',a.stmt_type) stmt_type
,      CAST(a.lsqltext AS VARCHAR2(1000)) lsqltext
FROM   sys.fga_log$ a
       LEFT OUTER JOIN pstreeselctl s
       ON CAST(a.lsqltext AS VARCHAR2(1000)) LIKE '%'||s.selector_num||'%'
,      y
WHERE  a.policyname  LIKE 'PSTREESELECT__'
and    a.ntimestamp# > sysdate-1
and    a.oshst like 'WINADROOT%'
and    a.oshst = y.oshst
and    a.sessionid = y.sessionid
and    y.policyname  = 'PS_NVS_REPORT_SEL'
and    y.ntimestamp# = (
        SELECT MAX(b1.ntimestamp#)
        FROM   sys.fga_log$ b1
        WHERE  b1.policyname  = y.policyname
        and    b1.oshst = y.oshst
        and    b1.sessionid = y.sessionid
        and    b1.ntimestamp# <= a.ntimestamp#)
--and    a.stmt_type = 2 --delete statements only
and    s.selector_num IS NULL --dynamic selector
GROUP BY a.policyname, y.parm2
,      CAST(a.lsqltext AS VARCHAR2(1000))
/
```

PARM2 is the name of the report.

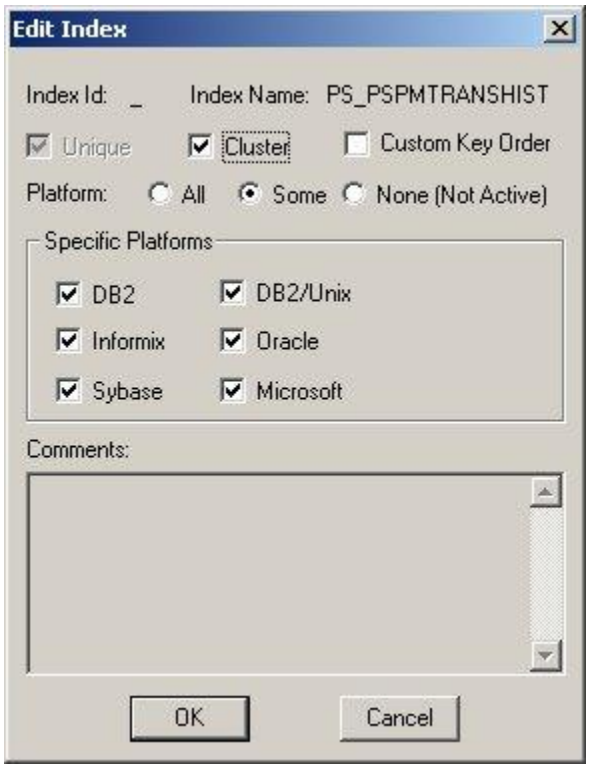
LSQL_TEXT shows the query that triggered the fine-grained audit and hence we can see the name of the tree involved.

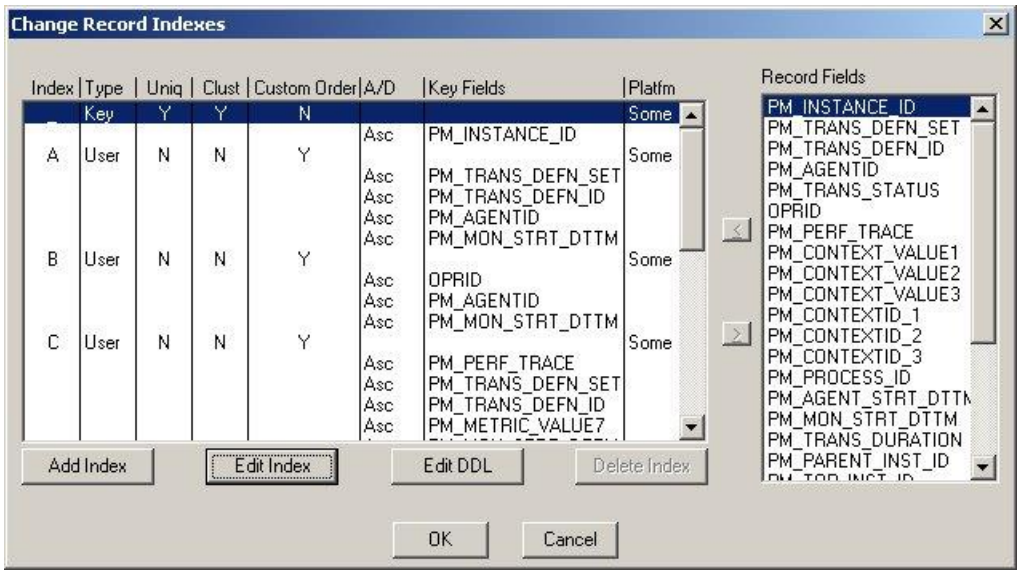
```
PARM2                POLYCYNAME           LSQLTEXT
-------------------- -------------------- ----------------------------------------------------------------

TR_RV_OP             PSTREESELECT10       DELETE FROM PSTREESELECT10 WHERE SELECTOR_NUM=279034

TR_RV_OP             PSTREESELECT10       DELETE FROM PSTREESELECT10 WHERE SELECTOR_NUM=279035

TR_RV_OP             PSTREESELECT10       INSERT INTO PSTREESELECT10(SELECTOR_NUM,TREE_NODE_NUM,RANGE_
                                          FROM_10,RANGE_TO_10) SELECT DISTINCT 279030,L.TREE_NODE_NUM,
                                          SUBSTR(L.RANGE_FROM,1,10), SUBSTR(L.RANGE_TO,1,10) FROM PSTR
                                          EELEAF L WHERE L.SETID='SHARE' AND L.SETCNTRLVALUE=' ' AND L
                                          .TREE_NAME='ACCTROLLUP' AND L.EFFDT=TO_DATE('2015-01-01','YY
                                          YY-MM-DD') AND L.TREE_NODE_NUM BETWEEN 1625000000 AND 168749
```

Other Recommended Changes

PeopleTools Index Platform Flags

Since PeopleTools 8.45 the platform radio button in PeopleTools is frequently set to 'some' though all the platform flags are ticked. This problem is explained at http://blog.psftdba.com/2006/08/peopletools-platform-flags-on-indexes.html





Though not part of the nVision performance work, it makes it much easier to review indexes in Application Designer if the platform button is set to ALL when all the flags are ticked.

Action: I recommend that it is applied to all PeopleSoft databases.

```
UPDATE PSVERSION
SET VERSION = VERSION + 1
WHERE OBJECTTYPENAME IN('SYS','RDM')
/

UPDATE PSLOCK
SET VERSION = VERSION + 1
WHERE OBJECTTYPENAME IN('SYS','RDM')
/

UPDATE PSRECDEFN
SET VERSION = (
  SELECT VERSION
  FROM PSVERSION
  WHERE OBJECTTYPENAME = 'RDM')
WHERE RECNAME IN (
  SELECT RECNAME
  FROM PSINDEXDEFN
  WHERE PLATFORM_DB2=PLATFORM_DBX
  AND PLATFORM_DBX=PLATFORM_INF
  AND PLATFORM_INF=PLATFORM_ORA
  AND PLATFORM_ORA=PLATFORM_SYB
  AND ( PLATFORM_ORA!=PLATFORM_SBS
    OR PLATFORM_ORA!=PLATFORM_ALB
    OR PLATFORM_ORA!=PLATFORM_DB4)
)
/

UPDATE psindexdefn
SET PLATFORM_DB4=PLATFORM_ORA
WHERE PLATFORM_DB2=PLATFORM_DBX
AND PLATFORM_DBX=PLATFORM_INF
AND PLATFORM_INF=PLATFORM_ORA
AND PLATFORM_ORA=PLATFORM_SYB
AND PLATFORM_SYB=PLATFORM_MSS
AND PLATFORM_ORA!=PLATFORM_DB4
/

UPDATE psindexdefn
SET PLATFORM_ALB=PLATFORM_ORA
WHERE PLATFORM_DB2=PLATFORM_DBX
AND PLATFORM_DBX=PLATFORM_INF
AND PLATFORM_INF=PLATFORM_ORA
AND PLATFORM_ORA=PLATFORM_SYB
AND PLATFORM_SYB=PLATFORM_MSS
AND PLATFORM_ORA!=PLATFORM_ALB
/

UPDATE psindexdefn
SET PLATFORM_SBS=PLATFORM_ORA
WHERE PLATFORM_DB2=PLATFORM_DBX
AND PLATFORM_DBX=PLATFORM_INF
AND PLATFORM_INF=PLATFORM_ORA
```

```
AND PLATFORM_ORA=PLATFORM_SYB
AND PLATFORM_SYB=PLATFORM_MSS
AND PLATFORM_ORA!=PLATFORM_SBS
/

COMMIT
/
```

Platformfix.sql