TECHNICAL PAPER

IMPROVING THE PERFORMANCE OF SECURITY VIEWS IN PEOPLESOFT HCM8.8

Prepared By David Kurtz, Go-Faster Consultancy Ltd.

Technical Paper

Version 0.1

Monday 2 May 2005

(E-mail: david.kurtz@go-faster.co.uk, telephone +44-7771-760660)

File: hcm88.security.views.doc, 2 May 2005

Contents

Introduction

Row-level security in the PeopleSoft Enterprise HCM product is based on a set of security views that determine whether an employee is, or will be, in a department that is at or below a point on the Department Security Tree to which the operator has been granted access. However, the SQL to achieve this is highly complex and does not always perform well.

To combat the performance problems, PeopleSoft have introduced what they call 'Fast Security'. There are two variants that apply to employee security.

- In Fast Security 1 they simply generate a table from the security view. This gives excellent performance, but has a number of problems.
  - The table is not updated when changes are made and so if a new employee is hired they are not visible to the system until the table is regenerated. Similarly if an employee changes department, they will not be visible to an operator who has access to the new department but not the old one until the table is regenerated.
  - The generated table can be very large and can take a significant time to refresh.
- In Fast Security 2 the Department Security table is converted into a matrix. This is effective because security access is only changed as a part of a process. This table only needs to be refreshed when the tree changes or when a new row-level security class is added.
  - Although this method does not suffer from the latency problems in method 1, it does not provide the same level of performance.

This paper discusses enhancing method to provide better performance by using a second pregenerated table, and how to keep that table up-to-date and thus resolve the latency problems.

Delivered Search Views

EMPLMT_SRCH_GBL is typical of the security views is HR. It includes

- Two correlated sub-queries of the JOB table to get the current effective dated row and the max effective dated row for that date. However any future dated rows on the JOB table are included.
- A WHERE EXISTS sub-query of the Department Security Tree on the PSTREENODE table to get the departments to which the user has access. Note that the department on the JOB table is referenced by this sub-query. A further WHERE NOT EXISTS sub-query to exclude any departments below the points of access on the security tree that to which the operator class has be excluded.

You will see how each of these areas are replaced with a pre-generated table.

```
SELECT A.EMPLID
 ,JOB.EMPL_RCD
 ,SEC.ROWSECCLASS
 ,SEC.ACCESS_CD
 ,B.NAME
 ,B.LAST_NAME_SRCH
 ,B.NAME_AC
 ,A.PER_STATUS
FROM PS_PERSON A
  ,PS_PERSON_NAME B
  ,PS_JOB JOB
  ,PS_SCRTY_TBL_DEPT SEC
WHERE A.EMPLID=B.EMPLID
AND   B.EMPLID=JOB.EMPLID
AND   (    JOB.EFFDT>=TO_DATE(TO_CHAR(SYSDATE,'YYYY-MM-DD'),'YYYY-MM-DD')
       OR (    JOB.EFFDT= (
                  SELECT MAX(JOB2.EFFDT)
                  FROM   PS_JOB JOB2
                  WHERE  JOB.EMPLID=JOB2.EMPLID
                  AND    JOB.EMPL_RCD=JOB2.EMPL_RCD
                  AND    JOB2.EFFDT<=TO_DATE(TO_CHAR(SYSDATE,'YYYY-MM-DD'),'YYYY-MM-DD')
                  )
           AND JOB.EFFSEQ= (
                  SELECT MAX(JOB3.EFFSEQ)
                  FROM PS_JOB JOB3
                  WHERE JOB.EMPLID=JOB3.EMPLID
                  AND JOB.EMPL_RCD=JOB3.EMPL_RCD
                  AND JOB.EFFDT=JOB3.EFFDT
                  )
          )
      )
AND   JOB.APPT_TYPE <> '1'
AND   SEC.ACCESS_CD='Y'
AND EXISTS (
      SELECT 'X'
      FROM   PSTREENODE TN
      WHERE  TN.SETID = SEC.SETID
      AND    TN.SETID = JOB.SETID_DEPT
      AND    TN.TREE_NAME='DEPT_SECURITY'
      AND    TN.EFFDT= SEC.TREE_EFFDT
      AND    TN.TREE_NODE=JOB.DEPTID
      AND    TN.TREE_NODE_NUM BETWEEN SEC.TREE_NODE_NUM AND SEC.TREE_NODE_NUM_END
      AND    NOT EXISTS (
             SELECT 'X'
             FROM   PS_SCRTY_TBL_DEPT SEC2
             WHERE  SEC.ROWSECCLASS = SEC2.ROWSECCLASS
             AND    SEC.SETID = SEC2.SETID
             AND    SEC.TREE_NODE_NUM <> SEC2.TREE_NODE_NUM
             AND    TN.TREE_NODE_NUM BETWEEN SEC2.TREE_NODE_NUM AND SEC2.TREE_NODE_NUM_END
             AND    SEC2.TREE_NODE_NUM BETWEEN SEC.TREE_NODE_NUM AND SEC.TREE_NODE_NUM_END
             )
      )
```

Identifying Search Views

The first task is to identify which security views are in use and to determine which of them can be improved by the Fast Security technique. The following queries interrogate the PeopleTools tables and report the security views in use. The views that can be tuned have been indicated with asterisks (*).

Component Search Records

Component Definition

Component search records are used to identify the key values of the data to be queried into a component. In PeopleTools 7.x components were called Panel Groups and the underlying PeopleTools tables have not been renamed.

```
SELECT searchrecname, COUNT(*)
FROM pspnlgrpdefn
GROUP BY searchrecname
ORDER BY 2 DESC
;

SEARCHRECNAME     COUNT(*)
--------------- ----------
PRCSRUNCNTL           1247
INSTALLATION           214
PERS_SRCH_GBL          106 *
EMPLMT_SRCH_GBL        104 *
HR_SS_PERS_SRCH         79
RUN_CNTL_HR             69
EMPLMT_SRCH_US          47 *
PSOPTIONS               47
OPR_ROWS_VW             42
EMPL_COMP_SRCH          33
RUN_CNTL_ST             28
COMPANY_TBL             23
GPDE_RC_PAYROLL         23
GPCH_RC_PAYROLL         22
PSPMSYSDEFN             21
```

```
W3EB_EMPL_SRCH            20
PA_CLC_TMP_SRCH           18
FPAEMPLOYM_SRC2           17
APPL_SRCH_GBL             16
HR_SS_EMPL_SRCH           14
PERS_SRCH_US              14 *
W3HR_PERS_SRCH            14
…
```

Menu Item Override

The component search record specified on the component can be overridden on the menuitem.

```
SELECT searchrecname, count(*)
FROM psmenuitem
GROUP BY searchrecname
ORDER BY 2 DESC
;

SEARCHRECNAME     COUNT(*)
--------------- ----------
                      6266
SUCCESSION_SRCH         17
POSN_SUCC_SRCH          11
PERS_SRCH_GBL            9 *
CAREERPLAN_SRCH          7
EMPLMT_SRCH_NLD          6
GPJP_YEA_PYSRCH          6
RUN_CNTL_HR              5
SUCCESS_TR_SRCH          4
KEY_POSN_SRCH            2
SUCCESS_EM_SRCH          2
APPL_SRCH_GBL            1
TL_TM_ESS_MPAY           1
TL_TM_ESS_FPAY           1
TL_SCH_ESS_SRCH          1
TL_MNG_ESS_PAY           1
TL_MNG_ESS_FPAY          1
TL_COMP_EES_SCH          1
PSDDLMODEL_VW            1
PERS_SRCH_NLD            1 *
CM_TREE_SRCH             1
ER_AM_RSLT_SRCH          1
DEPARTMENT_SRCH          1
EMPLMT_ADD2SRCH          1
EMPLMT_SRCH_COR          1 *
```

Query Security Records

Query security records are joined to records in PeopleSoft query tool, and hence also in Crystal reports. Query security records are recorded on the PSRECDEFN table.

```
SELECT qrysecrecname, COUNT(*)
```

```
FROM psrecdefn
GROUP by qrysecrecname
ORDER BY 2 DESC
;

QRYSECRECNAME     COUNT(*)
--------------- ----------
                     17704
EMPLMT_SRCH_QRY        562 *
PERS_SRCH_QRY          324 *
APPL_SRCH_GBL          236
EMPLMT_ST_QRY           40 *
APPL_EFFDT_SRCH         28
DEPT_SRCH_QRY           16
POSTN_SRCH_QRY          15
PERS_SRCH_ST             5 *
KEY_POSTN_QRY            3
EMPLMT_SRCH_GBL          2
APPLICANT_DATA           1
SAL_SEC_JPN_VW2          1
FPAEMPLOYM_SRC           1
JOB_REQUIS_SRCH          1
DEPENDENT_BENEF          1
```

Pre-Generated Tables

Two pregenerated tables are needed for the security views

PS_FAST_SCRTY_2

This table is delivered by PeopleSoft and is refreshed by the application engine program HR_FASTVIEW.

Table

It seems …

```
CREATE TABLE PS_FAST_SCRTY_2 (SETID VARCHAR2(5) NOT NULL,
   DEPTID VARCHAR2(10) NOT NULL,
   ROWSECCLASS VARCHAR2(30) NOT NULL,
   ACCESS_CD VARCHAR2(1) NOT NULL)
TABLESPACE HRLARGE
…
/
```

Indexes

The order of columns in the primary key index on FAST_SCRTY_2 has been changed to demote SETID to the last column. Often in HR systems there is only one SETID, and it is not specified in the queries.

```
CREATE UNIQUE INDEX PS_FAST_SCRTY_2 ON PS_FAST_SCRTY_2 (DEPTID,
   ROWSECCLASS,
   SETID)
TABLESPACE PSINDEX
…
/
```

Again, an additional unique index has been built on this table so that queries can be fully satisfied from the index without needing to visit the table.

```
CREATE UNIQUE  INDEX PSAFAST_SCRTY_2 ON PS_FAST_SCRTY_2
(ROWSECCLASS,
   DEPTID,
   SETID,
   ACCESS_CD)
TABLESPACE PSINDEX
…
/
```

PS_GEN_JOB_TBL

This is a new table that will be used to hold the values from the PS_JOB table that are used by the security views.

Table

```
CREATE TABLE PS_GEN_JOB_TBL (EMPLID VARCHAR2(11) NOT NULL,
   DEPTID VARCHAR2(10) NOT NULL,
   EMPL_RCD SMALLINT NOT NULL,
   SETID_DEPT VARCHAR2(5) NOT NULL,
   APPT_TYPE VARCHAR2(1) NOT NULL,
   EFFDT DATE)
TABLESPACE HRAPP
…
/
```

Indexes

Note that the EMPLID and DEPTID columns have been brought to the front of the index, and the less selective columns have been pushed to the end of the index.

```
CREATE UNIQUE INDEX PS_GEN_JOB_TBL ON PS_GEN_JOB_TBL (EMPLID,
   DEPTID,
   EMPL_RCD,
   SETID_DEPT,
   APPT_TYPE)
TABLESPACE PSINDEX
…
/
```

Although the first five columns form the unique key on this table, an additional unique user index on all the columns has also be created. This index will be used to satisfy the queries from the index without having to visit the table.

```
CREATE UNIQUE INDEX PSAGEN_JOB_TBL ON PS_GEN_JOB_TBL (EMPLID,
   DEPTID,
   EFFDT,
   EMPL_RCD,
   SETID_DEPT,
   APPT_TYPE)
TABLESPACE PSINDEX
…
/
```

Refreshing the Tables

The application engine program HR_FASTVIEW refreshes FAST_SCRTY_2, so it makes sense to fresh the generated job table in the same process.

A new view, PS_GEN_JOB_VW, has been created to to refresh the new table PS_GEN_JOB_TBL because there are two places where the table is maintained. One is HR_FASTVIEW, the other is the SavePostChange PeopleCode on the Job record.

```
CREATE VIEW PS_GEN_JOB_VW (EMPLID, DEPTID, EMPL_RCD, SETID_DEPT,
 APPT_TYPE, EFFDT)
AS SELECT EMPLID
 , DEPTID
 , EMPL_RCD
 , SETID_DEPT
 , APPT_TYPE
 , MIN(EFFDT)
  FROM PS_JOB JOB
 WHERE ( JOB.EFFDT>=TRUNC(SYSDATE)
    OR (JOB.EFFDT= (
 SELECT MAX(JOB2.EFFDT)
  FROM PS_JOB JOB2
 WHERE JOB.EMPLID=JOB2.EMPLID
   AND JOB.EMPL_RCD=JOB2.EMPL_RCD
   AND JOB2.EFFDT<=TRUNC(SYSDATE))
   AND JOB.EFFSEQ= (
 SELECT MAX(JOB3.EFFSEQ)
  FROM PS_JOB JOB3
 WHERE JOB.EMPLID=JOB3.EMPLID
   AND JOB.EMPL_RCD=JOB3.EMPL_RCD
   AND JOB.EFFDT=JOB3.EFFDT ) ) )
  GROUP BY EMPLID , DEPTID , EMPL_RCD , SETID_DEPT, APPT_TYPE
```

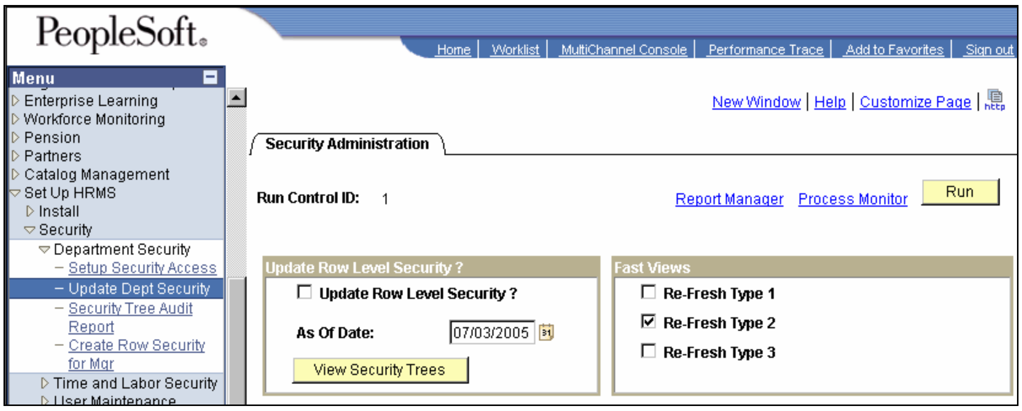
Note that this view does not return every current and future row from the JOB table. An employee may have several JOB rows that leave the employee in the same department. For security, it is only necessary to know that the employee is currently or will be in a department at some time in the future. By noting the minimum date that an employee was found in each department it is possible to distinguish current and future rows. The date on the current department will be less than or equal to the current date.

In version 8.8 PeopleSoft have referenced an extra column APPT_TYPE in some of the security views, so it has had to be added to PS_GEN_JOB_TBL and PS_GEN_JOB_VW. It must be a part of the primary key and must be included in the GROUP BY clause of the view.

Application HR_FASTVIEW

This application engine program already maintains the pre-generated security table FAST_SCRTY2_VW in Re-Fresh type 2.

Navigate to: Set Up HRMS -> Security -> Department Security -> Update Department Security ->



Run Control ID: 1

Update Row Level Security ?

- [ ] Update Row Level Security ?

As Of Date: 07/03/2005

Fast Views

- [ ] Re-Fresh Type 1
- [✓] Re-Fresh Type 2
- [ ] Re-Fresh Type 3

A new section GEN_JOB has been added to this program. It is called from a new step in section FASTVIEW

Ormal

The section GEN_JOB simply truncates and repopulates the generated JOB table. The table must be updated every day at midnight because hwne the date changes a new row on the JOB table can become effective.

```
[HR_FASTVIEW.GEN_JOB.Step01]
%TruncateTable(%Table(GEN_JOB_TBL))

[HR_FASTVIEW.GEN_JOB.Step02]
%InsertSelect(GEN_JOB_TBL,GEN_JOB_VW)
  FROM %Table(GEN_JOB_VW)

[HR_FASTVIEW.GEN_JOB.Step03]
%UpdateStats(GEN_JOB_TBL)
```

JOB.DEPTID.SavePostChange PeopleCode

The problem with introducing a pre-generated JOB table is that job changes that affect the row-level security that made during the day will not be reflected until the table is refreshed.

```
[JOB.DEPTID.SavePostChange]
/* maintain GEN_JOB_TBL whenever an update to PS_JOB is made */
SQLExec("DELETE FROM PS_GEN_JOB_TBL where EMPLID = :1 and EMPL_RCD = :2", JOB.EMPLID,
JOB.EMPL_RCD);
SQLExec("INSERT INTO PS_GEN_JOB_TBL SELECT * FROM PS_GEN_JOB_VW where EMPLID = :1 and
EMPL_RCD = :2", JOB.EMPLID, JOB.EMPL_RCD);
```

Changing the Security Views

The following security views should be changed as follows

EMPLMT_SRCH_COR

```
SELECT A.EMPLID
 ,JOB.EMPL_RCD %Sql(HRSECVIEWS_SELECT,SEC)
 ,B.NAME
 ,B.LAST_NAME_SRCH
 ,B.NAME_AC
 ,A.PER_STATUS
  FROM PS_PERSON A
  ,PS_PERSON_NAME B
  ,PS_GEN_JOB_TBL JOB
  ,PS_FAST_SCRTY_2 SEC
 WHERE A.EMPLID=B.EMPLID
   AND B.EMPLID=JOB.EMPLID
   AND JOB.EMPLID = A.EMPLID
   AND JOB.APPT_TYPE <> '1'
   AND JOB.APPT_TYPE = '0'
   AND SEC.ACCESS_CD='Y'
   AND JOB.DEPTID = SEC.DEPTID
   AND JOB.SETID_DEPT = SEC.SETID
```

EMPLMT_SRCH_GBL

```
SELECT A.EMPLID
 ,JOB.EMPL_RCD %Sql(HRSECVIEWS_SELECT,SEC)
 ,B.NAME
 ,B.LAST_NAME_SRCH
 ,B.NAME_AC
 ,A.PER_STATUS
  FROM PS_PERSON A
  ,PS_PERSON_NAME B
  ,PS_GEN_JOB_TBL JOB
  ,PS_FAST_SCRTY_2 SEC
 WHERE A.EMPLID=B.EMPLID
   AND B.EMPLID=JOB.EMPLID
   AND JOB.EMPLID = A.EMPLID
   AND JOB.APPT_TYPE <> '1'
   AND SEC.ACCESS_CD='Y'
   AND JOB.DEPTID = SEC.DEPTID
   AND JOB.SETID_DEPT = SEC.SETID
```

PERS_SRCH_GBL

```
SELECT A.EMPLID %Sql(HRSECVIEWS_SELECT,SEC)
 ,JOB.EMPL_RCD
 ,B.NAME
 ,B.LAST_NAME_SRCH
 ,JOB.SETID_DEPT
 ,JOB.DEPTID
 ,B.NAME_AC
 ,A.PER_STATUS
  FROM PS_PERSON A
  ,PS_PERSON_NAME B
  ,PS_GEN_JOB_TBL JOB
  ,PS_FAST_SCRTY_2 SEC
 WHERE A.EMPLID=B.EMPLID
   AND B.EMPLID=JOB.EMPLID
   AND JOB.EMPLID = A.EMPLID
   AND JOB.APPT_TYPE <> '1'
   AND SEC.ACCESS_CD='Y'
   AND JOB.DEPTID = SEC.DEPTID
   AND JOB.SETID_DEPT = SEC.SETID
```

EMPLMT_SRCH_QRY

Only the ROWSECCLASS and key columns are needed in Query Security Views. So only the two pregenerated tables are needed. This simlication produces further performance improvements.

Note that the condition on EFFDT restricts the query to current onlt effect security.

```
SELECT JOB.EMPLID
 ,JOB.EMPL_RCD
 ,SEC.ROWSECCLASS
  FROM PS_GEN_JOB_TBL JOB
  ,PS_FAST_SCRTY_2 SEC
 WHERE SEC.ACCESS_CD='Y'
   AND JOB.EFFDT <= SYSDATE
   AND JOB.DEPTID = SEC.DEPTID
   AND JOB.SETID_DEPT = SEC.SETID
```

PERS_SRCH_QRY

This view has been similarly simplified, but because EMP_RCD has not been referenced the DISTINCT is required.

```
SELECT DISTINCT JOB.EMPLID
 ,SEC.ROWSECCLASS
  FROM PS_GEN_JOB_TBL JOB
  ,PS_FAST_SCRTY_2 SEC
 WHERE SEC.ACCESS_CD='Y'
   AND JOB.EFFDT <= SYSDATE
   AND JOB.DEPTID = SEC.DEPTID
   AND JOB.SETID_DEPT = SEC.SETID
```