Fine-Grained Auditing in PeopleSoft

by David Kurtz, Go-Faster Consultancy Ltd.

Recently, somebody asked me whether it is possible to audit the fact that someone has queried a piece of data in PeopleSoft. It led to an interesting piece of research.

It is a fairly easy matter to configure Fine-Grained Auditing (FGA) in Oracle, you have to create a policy with the DBMS_FGA package. You can choose which columns to audit, and a logical audit condition to determine whether the rows should be audited. This example will log users who query salary data from the JOB record in PeopleSoft.

```
begin
 sys.dbms_fga.add_policy
 (object_schema=>'SYSADM'
 ,object_name=>'PS_JOB'
 ,policy_name=>'JOB_SALARY'
 ,audit_condition=>'ANNUAL_RT != 0 OR
MONTHLY_RT != 0 OR DAILY_RT != 0 OR
HOURLY_RT != 0 OR SHIFT_RT != 0 OR
SHIFT_FACTOR != 0'
 ,audit_column=>'ANNUAL_RT, MONTHLY_RT,
DAILY_RT, HOURLY_RT, SHIFT_RT,
SHIFT_FACTOR'
 ,enable=>TRUE
 ,statement_types=>'SELECT'
 ,handler_schema=>'SYS'
 ,handler_module=>'dmk_fga_hand(
object_schema, object_name, policy_name)'
 ,audit_trail=>DBMS_FGA.DB +
DBMS_FGA.EXTENDED
,audit_column_opts=>DBMS_FGA.ANY_COLUMNS)
;
end;
/
```

If the audit condition is matched for any rows in a query, and the query selects one of the audit columns, a single audit row is produced, irrespective of the number of rows that match.

Be aware that if the audit condition raises an error for any inspected row (such as an error caused by coercion), the audited query will also fail. There will be nothing in the message to suggest that it is caused by FGA, and so it can be difficult to diagnose and debug.

I chose to save the audit data to a table in the database, but alternatively, the audit data can be written as an XML data structure in a flat file written by setting:

```
, audit_trail=>DBMS_FGA.XML
```

Oracle writes a file to the directory indicated by the AUDIT_FILE_DEST parameter.

And this is the data the Oracle records in the table.

<insert example that is in footnote into text, but needs to be across two columns – let me know if can be made wider than 60 columns, and I will reformat it>[1]

It tells us is that a process, running as OS user *David* on node *GO-FASTER-4*,

1

```
SESSIONID DBUID    OSUID              OSHST
--------- -------- ------------------ ----------------------
EXTID                  OBJ$SCHEMA OBJ$NAME   POLICYNAME
---------------------- ---------- ---------- ----------
      SCN LSQLTEXT                              STMT_TYPE
--------- ------------------------------------ ----------
NTIMESTAMP#                INSTANCE# PROCESS#      STATEMENT
-------------------------- ---------- ------------ ----------
   ENTRYID LSQLBIND
---------- ------------------------------------
     1927 SYSADM   GO-FASTER-4\David  GO-FASTER\GO-FASTER-4
GO-FASTER-4\David      SYSADM     PS_JOB     JOB_SALARY
  6375305 SELECT EMPLID, EMPL_RCD, EFFDT, T            1
23-JUL-07 15.47.21.429000          0 6856:11944        12104
         2  #1(6):K0G004 #2(1):0
```

connected to the database as *SYSADM* queried some audited data on table *PS_JOB*. But this node is the application server, and it will look the same for all access to PeopleSoft via the Application Server or any other program.

We need to collect some more information from the session.

If you use database triggers to audit updates in PeopleSoft, they obtain the PeopleSoft operator ID from the client information string that is set by a call to the DBMS_APPLICATION_INFO package at the start of every service in the application server. They extract the operator ID with a function called GET_PS_OPRID (delivered in *getpsoprid.sql*)

You can't put a trigger on the audit table (SYS.FGA_LOG$) because it is owned by SYS, but we could use the error handler to call a custom PL/SQL procedure. The handler module is fired every time an audit record is written. It is intended to permit a notification to be sent on audit to the DBA when a policy is breached. But we could use it for a different purpose. The handler runs in the same session as the process that performs the audited action. So I can collect information about the session and write it to another audit table.

```
CREATE TABLE sys.dmk_fga
(timestamp# DATE
,object_schema VARCHAR2(30)
,object_name VARCHAR2(128)
,policy_name VARCHAR2(30)
,ntimestamp# TIMESTAMP
,instance# INTEGER
,sessionid INTEGER
,entryid INTEGER
,scn INTEGER
,action VARCHAR2(32)
,module VARCHAR2(48)
,client_info VARCHAR2(64)
);
```

This is my handler procedure

<example code in footer>[2]

And this is the information that is inserted into my new logging table

<example code in footer>[3]

The SCNs do not match between these two tables. There is a difference of at least 3, but the tables can be joined on SESSIONID and ENTRYID (so you will probably need an index on these columns).

```
set long 5000
SELECT --b.client_info,
```

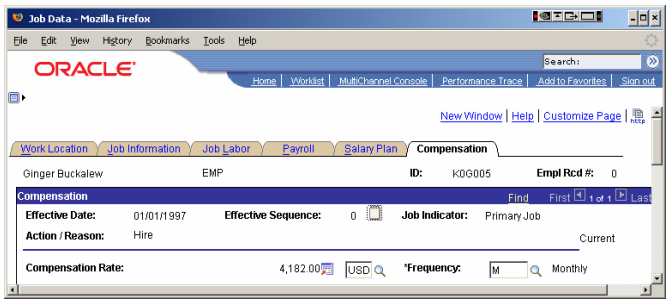
---

[2]
```
CREATE OR REPLACE PROCEDURE dmk_fga_hand (
object_schema VARCHAR2, object_name VARCHAR2,
policy_name VARCHAR2 ) AS
 l_client_info VARCHAR2(64);
 l_action varchar2(32);
 l_module varchar2(48);
BEGIN
 sys.dbms_application_info.read_client_info(l_client_info);
 sys.dbms_application_info.read_module(l_module, l_action);
 INSERT INTO dmk_fga
(timestamp#, ntimestamp#, object_schema, object_name,
policy_name, instance#, sessionid, entryid, scn, client_info, action,
module)
 SELECT sysdate, systimestamp, object_schema, object_name,
policy_name, i.instance_number, USERENV('SESSIONID'),
USERENV('ENTRYID'), d.current_scn, l_client_info, l_action,
l_module
 FROM v$database d, v$instance i
 WHERE rownum <= 1;
 EXCEPTION WHEN OTHERS THEN NULL;
end;
/
```

[3]
```
TIMESTAMP#          OBJECT_SCH OBJECT_NAM POLICY_NAM
------------------- ---------- ---------- ----------
NTIMESTAMP#                INSTANCE# SESSIONID ENTRYID
-------------------------- --------- --------- -------
      SCN ACTION               MODULE
--------- -------------------- --------------------
CLIENT_INFO
--------------------------------------------------------
23:00:54 23/07/2007 SYSADM     PS_JOB     JOB_SALARY
23-JUL-07 23.00.54.015000         1      1927       6
  6390779                      PSAPPSRV.exe
PS,,go-faster-4.london.go-
faster.co.uk,HCM89,PSAPPSRV.exe,
```

```
sysadm.get_ps_oprid(b.client_info)
oprid,a.ntimestamp#, a.POLICYNAME
,a.lsqltext, a.lsqlbind
FROM  sys.fga_log$ a, sys.dmk_fga b
WHERE a.sessionid = b.sessionid
AND   a.entryid = b.entryid
;
```

So, if I now when I open a component in PeopleSoft which queries somebody's compensation rate, the FGA policy generates a record.



I can use the above query to combine the audit data. It this example it shows that operator PS queried salary data for employee K0G005.

<insert query results>[4]

Note that the data was queried from the database when the component was opened, and that is when the audit occurred. The operator would have to navigate to the 'Compensation' tab to see the data, so the audit does not absolutely prove the data was displayed to the operator, but that is only a semantic difference.

Overhead

The ability to audit is all very well, but it comes at a price. It is not difficult to generate a large number of audit rows. In the following PL/SQL block, an audit row is generated for each time the query inside the loop is executed (2050 times on my HR demo database).

```
declare
 l_annual_rt NUMBER;
begin
FOR i IN (SELECT DISTINCT emplid,
empl_rcd FROM ps_job WHERE effdt <=
SYSDATE) loop
  SELECT j.annual_rt
  INTO l_annual_rt
  FROM ps_job j
  WHERE j.emplid = i.emplid
  AND j.empl_rcd = i.empl_rcd
  AND j.effdt = (SELECT MAX(j1.effdt)
FROM ps_job j1 WHERE j1.emplid = i.emplid
AND j1.empl_rcd = i.empl_rcd AND j1.effdt
<= SYSDATE)
  AND j.effseq = (SELECT MAX(j2.effseq)
FROM ps_job j2 WHERE j2.emplid = i.emplid
AND j2.empl_rcd = i.empl_rcd AND j2.effdt
= j.effdt);
 END LOOP;
END;
/
```

This can have a dramatic effect on performance. I ran the above PL/SQL on my laptop with trace.

Operator	Without FGA	With FGA
Driving Query	0.46s	0.43s
Inner Query	0.92s	3.43s
Insert into FGA_LOG$		1.68s
Insert into DMK_FGA		6.48s
Other time in Exception Handler		0.72s
Total	1.38s	12.75s

FGA causes a huge increase in the response time of this test. Most of the additional time is spent inserting the audit rows my handler procedure. The performance of the single disk in my laptop is probably magnifying the effect.

4

```
OPRID        NTIMESTAMP#                 POLICYNAME
------------ --------------------------- ----------
LSQLTEXT
--------------------------------------------------
LSQLBIND
--------------------------------------------------
PS           23-JUL-07 22.00.53.968000 JOB_SALARY
SELECT EMPLID, EMPL_RCD, EFFDT, TO_CHAR(EFFDT,'YYYY-
...
PL_RCD, EFFDT DESC, EFFSEQ DESC
 #1(6):K0G005 #2(1):0
```

Conclusion

I have demonstrated that it is possible to use FGA to monitor who is looking at what in PeopleSoft as well as detecting other forms of break-in. However, it can introduce a significant run-time overhead. The ability to incorporate a custom PL/SQL notification procedure suggests that it is designed with the intention of logging exceptional activities rather than routine ones that users are permitted to perform. While FGA could be used to audit access to sensitive data, I think that row-level security should be setup correctly in PeopleSoft in the first place, so that only duly authorised users can access sensitive data. FGA should only be used conservatively to monitor exceptional events.

David Kurtz is a performance specialist working Enterprise PeopleSoft applications and Oracle RDBMS. David is the author of 'PeopleSoft for the Oracle DBA', published by Apress (www.psftdba.com). Since December 2006, he has been a UKOUG Director with responsibility for representing the PeopleSoft community.