Example 2: Implementing row-level access policies

Overview

Implementing row-level access is probably the most common form of security controls applied using VPD. It prevents rows from being returned that do not meet the condition defined in policy function, and is activated in any condition regardless of the columns participating in the statement.

In this recipe we will create a new table EMPLOYEES_REG_DATA_VPD in the HR schema, based on the VIEW_REG_DATA definition created in the previous recipe. Next, we will create a policy function that will limit the data that is returned by dynamically applying a region restriction through the application context HR_REGVIW_CONTEXT.

Basically we recreate the scenario used in the previous recipe, but this time using VPD components.

Workflow

1. As user HR create a table EMPLOYEES_REG_DATA_VPD as follows:

```
CREATE TABLE EMPLOYEES_REG_DATA_VPD
AS
  SELECT E.FIRST_NAME,
    E.LAST_NAME,
    E.EMAIL,
    E.PHONE_NUMBER,
    E.HIRE_DATE,
    J.JOB_TITLE,
    E.SALARY,
    E.COMMISSION_PCT,
    D.DEPARTMENT_NAME,
    L.STATE_PROVINCE,
    L.CITY,
    L.POSTAL_CODE,
    C.COUNTRY_NAME,
    C.REGION_ID
  FROM HR.EMPLOYEES E
  JOIN HR.DEPARTMENTS D
  ON E.DEPARTMENT_ID=D.DEPARTMENT_ID
  JOIN HR.JOBS J
  ON E.JOB_ID=J.JOB_ID
  JOIN HR.LOCATIONS L
  ON D.LOCATION_ID=L.LOCATION_ID
  JOIN HR.COUNTRIES C
  ON L.COUNTRY_ID=C.COUNTRY_ID
```

2. Grant select on table EMPLOYEES_REG_DATA_VPD to vw_america and vw_europe users as follows:

```
GRANT SELECT ON EMPLOYEES_REG_DATA_VPD TO VW_AMERICA,VW_EUROPE;
```

3. Connect as user system and create region_id_plc_func VPD policy function as follows:

```
conn system
Enter password:

 CREATE OR REPLACE
  FUNCTION region_id_plc_func
    (
      schema_v IN VARCHAR2,
      tbl_v VARCHAR2)
    RETURN VARCHAR2
  IS
    ret_val VARCHAR2(200);
  BEGIN
    ret_val := 'region_id = sys_context(''hr_regviw_context'',''region_id'')';
    RETURN ret_val;
  END;
```

4. Next, define a policy named SELECT_REGIONS _POLICY , defined on object EMPLOYEES_REG_DATA_VPD from schema HR, and applicable only for select statements as follows:

```
BEGIN
  DBMS_RLS.ADD_POLICY ( object_schema => 'HR', object_name =>
'EMPLOYEES_REG_DATA_VPD', policy_name => 'SELECT_REGIONS_POLICY', function_schema
=> 'SYSTEM', policy_function => 'region_id_plc_func', statement_types =>
'select');
END;
```

5. Connect as users vw_europe and vw_america, and issue a select distinct statement to see if the SELECT_REGIONS_POLICY VPD policy is correctly applied:

```
conn vw_europe
Enter password:

select distinct country_name from hr. EMPLOYEES_REG_DATA_VPD;
4

conn vw_america
Enter password:
Connected.

select distinct country_name from hr. EMPLOYEES_REG_DATA_VPD;
```

6. If you try to select from table EMPLOYEES_REG_DATA_VPD connected as other users that have select privileges, no rows will be returned. Connect as system and reissue the previous select distinct statement as follows:

```
conn system
Enter password:
Connected.

select distinct country_name from hr. EMPLOYEES_REG_DATA_VPD;
```

7. Next we will create a policy function for inserts and delete. In the following steps we will create an empty table and a VPD policy applicable for insert statements. Connect as user HR and create an empty table named EMPLOYEES_REG_DATA_VPD_EU based on the structure of EMPLOYEES_REG_DATA_VPD as follows:

```
create table EMPLOYEES_REG_DATA_EU_VPD as select * from EMPLOYEES_REG_DATA_VPD;
```

8. As system, create a policy name REGION_ID_EU_PLC_FNC, which will allow inserts only for rows corresponding to region 1 or Europe as follows:

```
CREATE OR REPLACE
      FUNCTION region_id_EU_plc_func
        (
          schema_v IN VARCHAR2,
          tbl_v VARCHAR2)
        RETURN VARCHAR2
      IS
        ret_val VARCHAR2(200);
      BEGIN
        ret_val := 'region_id = 1';
        RETURN ret_val;
      END;
```

9. Create a insert policy called INSERT_EU_POLICY as follows:

```
BEGIN
    DBMS_RLS.ADD_POLICY (
object_schema => 'HR',
object_name => 'EMPLOYEES_REG_DATA_VPD_EU',
policy_name => 'INSERT_EU_POLICY',
function_schema => 'SYSTEM',
policy_function => 'region_id_eu_plc_func',
statement_types => 'insert');
END;
```

10. Next, try to insert some values that are not compliant with INSERT_EU_POLICY as follows:

```
conn HR/HR
 INSERT INTO EMPLOYEES_REG_DATA_EU_VPD values
('Donald','OConnell','DOCONNEL','650.507.9833',to_timestamp('21-06-2007','DD-MM-R
RRR HH24:MI:SSXFF'),'Shipping
 Clerk',4100,null,'Shipping','California','South San Francisco','99236','United
States of America',2);

1 row created.
```

11. Surprisingly it works. To enforce VPD policy on insert statement we must to enable update_check parameter. Connect as system, drop the policy and recreate it with update_check=>true, and reissue the previous INSERT statement as follows:

```
conn system
Enter password:
Connected.

execute dbms_rls.drop_policy(
object_schema =>'HR',
policy_name => 'INSERT_EU_POLICY',
object_name => 'EMPLOYEES_REG_DATA_EU_VPD'
);

PL/SQL procedure successfully completed.

BEGIN
DBMS_RLS.ADD_POLICY
(
object_schema => 'HR',
object_name => 'EMPLOYEES_REG_DATA_EU_VPD',
policy_name => 'INSERT_EU_POLICY',
function_schema => 'SYSTEM',
policy_function => 'region_id_eu_plc_func',
statement_types => 'insert',
update_check => true
);
END;
```

12. Connect as HR and reissue the previous insert as follows:

```
conn HR
Enter password:

INSERT INTO EMPLOYEES_REG_DATA_EU_VPD values
('Donald','OConnell','DOCONNEL','650.507.9833',to_timestamp('21-06-2007','DD-MM-R
RRR HH24:MI:SSXFF'),'Shipping
 Clerk',4100,null,'Shipping','California','South San Francisco','99236','United
States of America',2);
INSERT INTO EMPLOYEES_REG_DATA_EU_VPD values
('Donald','OConnell','DOCONNEL','650.507.9833',to_timestamp('21-06-2007','DD-MM-R
RRR HH24:MI:SSXFF'),'Shipping Cler
k',4100,null,'Shipping','California','South San Francisco','99236','United States
of America',2);

ERROR at line 1:
ORA-28115: policy with check option violation
```

13. This time the policy is enforced. Now, insert values that are compliant with the policy:

```
Insert into EMPLOYEES_REG_DATA_VPD values
('Hermann','Baer','HBAER','515.123.8888',to_timestamp('07-06-2002','DD-MM-RRRR
HH24:MI:SSXFF'),'Public Relations
Representative',10000,null,'Public
Relations','Bavaria','Munich','80925','Germany',1);

commit;
```

14. Next, in the following steps, we will create a policy function and VPD policy to deal with delete statement. Connect as user HR and create a table EMPLOYEES_SAL_COMPCT_VPD, which will contain first_name,last_name, salary, and commission_pct columns, as follows:

```
create table employees_sal_cmpct_vpd as select
first_name,last_name,salary,commission_pct from employees;
```

15. Create a policy function named COST_REDUCTION_PLC_FUNC, which will be applied for salaries over 5000 and commissions over 0.1 as follows:

```
CREATE OR REPLACE
     FUNCTION cost_reduction_plc_func
       (
         schema_v IN VARCHAR2,
         tbl_v VARCHAR2)
       RETURN VARCHAR2
     IS
       ret_val VARCHAR2(200);
     BEGIN
       ret_val := 'salary > 5000 and commission_pct > 0.1';
       RETURN ret_val;
     END;
```

16. Next, create a policy applicable on the EMPLOYEES_SAL_CMPCT_VPD table using COST_REDUCTION_PLC_FUNC for delete statements, named COST_REDUCTION_POLICY, as follows:

```
BEGIN
 DBMS_RLS.ADD_POLICY (
  object_schema => 'HR',
object_name => 'EMPLOYEES_SAL_CMPCT_VPD',
policy_name => 'COST_REDUCTION_POLICY',
function_schema => 'SYSTEM' ,
policy_function => 'COST_REDUCTION_PLC_FUNC',
statement_types => 'delete'
);
END;
```

17. Issue a count to check the number of rows for value of salary greater than 5000 and the value of commission_pct greater than 0.1:

```
select count(*) from employees_sal_cmpct_vpd where salary > 5000 and
commission_pct > 0.1;

COUNT(*)
----------
29
```

18. Connect as HR and issue a delete command on employees_sal_cmpct_vpd table as follows:

```
conn HR
Enter password:

delete employees_sal_cmpct_vpd;

commit;
```

19. If you issue the delete command again, no rows will be deleted because no one fits in the policy check:

```
delete employees_sal_cmpct_vpd;

0 rows deleted.
```

How it works

As a table, view, or synonym is associated with a policy, all statements that are found in the category defined in the policy will be dynamically rewritten to apply the policy condition when they are executed. The statement types are defined within the policy by using the statement_type input variable of package DBMS_RLS. As mentioned before, there could be defined policies on SELECT, UPDATE, DELETE, INSERT, and INDEX statements. The default is all but INDEX.

If the statement issued against an object has a WHERE clause, then the policy predicate will be added to the clause. When there is no WHERE clause one will be added in order to apply the policy predicate to the statement. The policy function must have the function arguments declared as object_name and object_schema, and the return value should always be varchar2 type. The predicate returned must form a valid WHERE clause. There must not be a circular reference for the object defined in the policy. In other words, you cannot use the protected object to generate the policy predicate.

There's more…

The Execute privilege on DBMS_RLS should be granted to the security administrator user and not to application users. In this way the VPD policies will be controlled only by a privileged user, which will be audited. There is a special policy parameter named UPDATE_CHECK. When this parameter is set to TRUE, the policy will check the after values and the before values issued from an UPDATE or INSERT statement. More information about VPD policies can be found in the ALL_POLICIES and DBA_POLICIES dictionary views.

Performance implications

In most cases, using VPD can lead to increase in performance because the final result set is decreased in size. However in some cases using complex queries having several tables with VPD policies enabled can lead to performance degradation. To find out the predicates used for query rewrite you may use event 10730. For more information check oracle support note [ID 967042.1] - How to Investigate Query Performance Regressions Caused by VPD (FGAC) Predicates?