

Basic Database Security

Creating a password profile

1. Connect to the database as a user who has create profile privilege:

```
sqlplus /
```

2. Create a password profile:

```
create profile userprofile limit
```

```
failed_login_attempts 4
```

```
password_lock_time 2
```

```
password_life_time 180;
```

3. Alter the user to use a newly created password profile:

```
alter user scott profile userprofile;
```

4. Alter the default password profile:

```
alter profile default limit
```

```
failed_login_attempts 4;
```

Creating password-authenticated users

1. Connect to the database as a user who has create user privilege:

```
$ sqlplus /
```

2. Create a password-authenticated user (for example, username: jessica, password: oracle_1) as follows:

```
SQL> create user jessica identified by oracle_1;
```

3. Create a password-authenticated user with a more complex password:

```
SQL> create user tom identified by "Qax7UnP!123*";
```

4. Create a user that uses a specific password profile:

```
SQL> create user steve identified by test1 profile  
userprofile;
```

5. Create a user and force it to change password upon the first login:

```
SQL> create user john identified by password1  
password expire;
```

6. Create a user richard, whose default tablespace is users, temporary tablespace is temp, and who has their quota set to unlimited on the users tablespace:

```
SQL> create user richard identified by oracle_2 default  
tablespace users temporary tablespace temp quota unlimited  
on users;
```

Changing a user's password

1. Connect to the database as a user who has alter user privilege:

```
$ sqlplus /
```

2. Change the password for user jessica:

```
SQL> password jessica;
```

3. Enter a new password (for example, oracle_2) on a command line (note that typing will not be visible in the command line):

New password:

4. Retype the new password (for example, oracle_2) on the command line (note that typing will not be visible in the command line):

Retype new password:

5. Connect to the database as any user (for example, tom, to change their own password):

```
$ sqlplus tom/"Qax7UnP!123*"
```

6. Change the password using the following code:

```
SQL> password
```

7. Enter the old password (for example, Qax7UnP!123) on the command line (note that typing will not be visible on the command line):*

Old password:

8. Enter the new password (for example, oracle_123) on the command line (note that typing will not be visible on the command line):

New password:

9. Retype the new password (for example, oracle_123) on the command line (note that typing will not be visible on the command line):

Retype new password:

Creating a user with the same credentials on another database

1. Connect to the first database as a user who has a DBA role:

\$ sqlplus /

2. Find a Data Definition Language (DDL) statement (ddl) that is used for user creation (for example, user jessica):

```
SQL> select dbms_metadata.get_ddl('USER', 'JESSICA') from  
dual;
```

3. Connect to the second database as a user who has create user privilege:

\$ sqlplus ernesto@orcl2

4. Create a user using the value you found in step 2:

```
SQL> create user "JESSICA" identified by values
```

'S:D82E6EF961F2EA7A878BCDDBC7E5C542BC148C4759D19A7
20A96BBF65658;H:F297A50FD538EF4AB119EB0278C9E72D;
C50B1E9C9AA52EC2';

Locking a user account

1. Connect to the database as a user who has alter user privilege:

SQL> \$ sqlplus /

2. Lock the account of user steve:

SQL> alter user steve account lock;

3. Unlock the account of user steve:

SQL> alter user steve account unlock;

Expiring a user's password

1. Connect to the database as a user who has the alter user privilege:

SQL> \$ sqlplus /

2. Steve's password expires with the following command:

SQL> alter user steve password expire;

Creating and using OS-authenticated users

1. Connect to the database as a user who has a DBA role:

\$ sqlplus johndba

2. Find the prefix for operating system authentication:

SQL> show parameter os_authent_prefix

NAME	TYPE	VALUE
os_authent_prefix	string	ops\$

3. Create an OS-authenticated user:

SQL> create user ops\$ernesto identified externally;

4. Grant this user the create session privilege:

SQL> grant create session to ops\$ernesto;

5. Log in to the operating system as the user ernesto:

\$ su - ernesto

6. Connect to the database without entering a user name or password:

\$ sqlplus /

Creating and using proxy users

1. Connect to the database as a user who has a DBA role:

```
$ sqlplus /
```

2. Create a proxy user named appserver:

```
SQL> create user appserver identified by oracle_1;
```

3. Grant create session to the user appserver:

```
SQL> grant create session to appserver;
```

4. Alter the user to connect through the proxy user:

```
SQL> alter user steve grant connect through appserver;
```

5. Connect to the database through proxy user:

```
SQL> connect appserver[steve]
```

6. Enter a password for the appserver user (for example, oracle_1):

Enter password:

7. To revoke connection through the proxy user, first connect to the database as a user who has altered user privilege:

```
$ sqlplus /
```

8. Revoke connection through the proxy user appserver from user steve:

SQL> alter user steve revoke connect through appserver;

Creating and using database roles

1. Connect to the database as a user who has a dba role:

\$ sqlplus /

2. Create the role usr_role:

SQL> create role usr_role;

3. Grant system privilege to usr_role:

SQL> grant create session to usr_role;

4. Grant object privileges to usr_role:

SQL> grant select, insert on hr.employees to usr_role;

5. Create another role as follows:

SQL> create role mgr_role;

6. Grant usr_role to mgr_role:

SQL> grant usr_role to mgr_role;

7. Grant system privileges to mgr_role:

SQL> grant create table to mgr_role;

8. Grant object privileges to mgr_role:

SQL> grant update, delete on hr.employees to mgr_role;

9. Grant usr_role to user (steve):

SQL> grant usr_role to steve;

10. Grant mgr_role to user (tom):

SQL> grant mgr_role to tom;

The sysbackup privilege – how, when, and why should you use it?

Database authentication

The instructions for database authentication are as follows:

1. Connect to the database as sysdba (or another user that can grant the sysbackup privilege):

sqlplus / as sysdba

2. Grant the sysbackup privilege to user tom:

```
grant sysbackup to tom;
```

3. Verify that there is an entry in the password file that grants user tom the sysbackup administrative privilege. Select data from the v\$pwfile_users view:

```
select * from v$pwfile_users;
```

The following table is the result of the preceding command:

Username	sysdb	sysop	sysas	sysba	sysdg	syskm	con_id
sys	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	0
sysdg	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	0
sysbackup	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	0
syskm	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	0
tom	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	0

4. Test the connection using RMAN:

```
rman target ""tom/oracle_123 as sysbackup""
```

OS authentication

The instructions for OS authentication are as follows:

1. Verify that the OS user (for example, john) is a member of the backupdba OS group:

```
$ id john
```

2. Connect to the database using the sysbackup privilege (SQL*Plus or RMAN):

```
$> sqlplus / as sysbackup
```

```
$> rman target ''/ as sysbackup''
```

The syskm privilege – how, when, and why should you use it?

Database authentication

The instructions for database authentication are as follows:

1. Connect to the database as sysdba (or another user that can grant the syskm privilege):

```
sqlplus / as sysdba
```

2. Grant the syskm privilege to user jessica:

```
grant syskm to jessica;
```

3. Connect user jessica to the database as syskm:

```
SQL> connect jessica/oracle_1 as syskm
```

4. View privileges:

```
SQL> select * from user_tab_privs;
```

```
SQL> select * from session_privs;
```

OS authentication

The instructions for OS authentication are as follows:

- 1. Verify that an OS user (for example, bob) is a member of the kmdba OS group.***

\$ id bob

- 2. Connect to the database using syskm privilege:***

\$ sqlplus / as syskm

The sysdg privilege – how, when, and why should you use it?

Database authentication

The instructions for database authentication are as follows:

- 1. Connect to the database as sysdba (or another user who can grant the sysdg privilege):***

sqlplus / as sysdba

- 2. Grant SYSDG privilege to user steve:***

SQL> grant sysdg to steve;

- 3. Exit SQL*Plus, connect steve using the dgmgrl command-line interface:***

SQL> exit

\$ dgmgrl

```
DGMRRL> connect steve/test_1
```

OS authentication

The instructions for OS authentication are as follows:

- 1. Verify that the OS user (for example, kelly) is a member of the dgdba OS group:***

```
$ id kelly
```

- 2. Connect using the dgmgrl utility and OS authentication:***

```
$ dgmgrl
```

```
DGMGRL> connect /
```

Security Considerations in Multitenant Environment

Creating a common user

- 1. Connect to the root container as a common user who has create user privilege granted commonly (for example, c##ernesto or system user):***

```
SQL> connect c##ernesto@cdb1
```

- 2. Create a common user (for example, c##maja):***

```
c##ernesto@CDB1> create user c##maja identified by oracle1  
container=all;
```

Creating a local user

1. *Connect to PDB (for example, pdb1) as a common user or local user who has create userprivilege in that PDB (for example, c##ernesto or system user):*

```
SQL> connect c##ernesto@pdb1
```

2. *Create a local user (for example, steve):*

```
c##ernesto@PDB1> create user steve identified by pa3t5brii  
container=current;
```

Creating a common role

1. *Connect to the root container as a common user who has create role privilege granted commonly (for example, c##ernesto or system user):*

```
SQL> connect c##ernesto@cdb1
```

2. *Create a common role (for example, c##role1):*

```
SQL> create role c##role1 container=all;
```

Creating a local role

1. *Connect to PDB (for example, pdb1) as a common or local user who has create role privilege in that PDB (for example, c##maja):*

```
SQL> connect c##maja@pdb1
```

2. Create a local role (for example, local_role1):

```
c##maja@PDB1> create role local_role1 container=current;
```

Granting privileges and roles commonly

1. You should connect to the root container as a common user who can grant these privileges and roles (for example, c##maja or system user):

```
SQL> connect c##maja@cdb1
```

2. Grant a privilege (for example, create session) to a common user (for example, c##john) commonly:

```
c##maja@CDB1> grant create session to c##john container=all;
```

3. Grant a privilege (for example, select any table) to a common role (for example, c##role1) commonly:

```
c##maja@CDB1> grant select any table to c##role1 container=all;
```

4. Grant a common role (for example, c##role1) to a common role (for example, c##role2) commonly:

```
c##maja@CDB1> grant c##role1 to c##role2 container=all;
```

5. Grant a common role (for example, c##role2) to a common user (for example, c##john) commonly:

```
c##maja@CDB1> grant c##role2 to c##john container=all;
```

Granting privileges and roles locally

- 1. You should connect to the container (root or pluggable database) in which you want to grant the privilege as a common or local user who can grant that privilege (for example, c##maja):*

```
SQL> connect c##maja@pdb1
```

- 2. Grant a privilege (for example, create synonym) to a common user (for example, c##john) locally:*

```
c##maja@PDB1> grant create synonym to c##john container=current;
```

- 3. Grant a privilege (for example, create view) to a local user (for example, steve) locally:*

```
c##maja@PDB1> grant create view to steve container=current;
```

- 4. Grant a privilege (for example, create table) to a common role (for example, c##role1) locally:*

```
c##maja@PDB1> grant create table to c##role1 container=current;
```

- 5. Grant a privilege (for example, create procedure) to a local role (for example, local_role1) locally:*

```
c##maja@PDB1> grant create procedure to local_role1
```

```
container=current;
```

- 6. Grant a common role (for example, c##role2) to another common role (for example, c##role3) locally:*

```
c##maja@PDB1> grant c##role2 to c##role3 container=current;
```

7. Grant a common role (for example, c##role3) to a local role (for example, local_role1) locally:

```
c##maja@PDB1> grant c##role3 to local_role1 container=current;
```

8. Grant a local role (for example, local_role1) to a common role (for example, c##role4) locally:

```
c##maja@PDB1> grant local_role1 to c##role4 container=current;
```

9. Grant a common role (for example, c##role4) to a common user (for example, c##john) locally:

```
c##maja@PDB1> grant c##role4 to c##john container=current;
```

Effects of plugging/unplugging operations on users, roles, and privileges

1. Connect to the root container of cdb1 as user sys:

```
SQL> connect sys@cdb1 as sysdba
```

2. Unplug pdb1 by creating an XML metadata file:

```
SQL> alter pluggable database pdb1 unplug into  
'/u02/oradata/pdb1.xml';
```

3. Drop pdb1 and keep the datafiles:

```
SQL> drop pluggable database pdb1 keep datafiles;
```

4. Connect to the root container of cdb2 as user sys:

```
SQL> connect sys@cdb2 as sysdba
```

5. Create (plug) pdb1 to cdb2 by using the previously created metadata file:

```
SQL> create pluggable database pdb1 using '/u02/oradata/pdb1.xml'
```

```
    nocopy;
```

PL/SQL Security

Creating and using definer's rights procedures

1. Connect to the database as a user with the DBA role (for example, ernesto)

```
SQL> connect ernesto
```

2. Create two users (procowner and procuser) and grant them appropriate privileges:

```
SQL> create user procowner identified by oracle1;
```

```
SQL> create user procuser identified by oracle2;
```

```
SQL> grant create session, create procedure to procowner;
```

```
SQL> grant create session to procuser;
```

3. Create a table called ernesto.tbl and grant users privileges on this table:

```
SQL> create table ernesto.tbl(a number, b varchar2(40));  
SQL> insert into ernesto.tbl values(1, 'old_value');  
SQL> commit;  
SQL> grant select on ernesto.tbl to procuser;  
SQL> grant update on ernesto.tbl to procowner;
```

4. Connect as a user, procowner, create a procedure to update table ernesto.tbl, and grant execute on this procedure to user procuser:

```
SQL> connect procowner/oracle1  
CREATE OR REPLACE PROCEDURE UpdateTbl (x IN number,  
y IN varchar2)  
AUTHID DEFINER  
AS  
BEGIN  
    UPDATE ERNESTO.TBL  
    SET b = y  
    WHERE a = x;  
END;  
/  
SQL> grant execute on UpdateTbl to procuser;
```

5. Connect as user procuser and try to directly update table ernesto.tbl:

```
SQL> connect procuser/oracle2
```

```
SQL> UPDATE ERNESTO.TBL SET B = 'value1' WHERE A = 1;
```

```
UPDATE ERNESTO.TBL SET B = 'value1' WHERE A = 1
```

```
*
```

ERROR at line 1:

ORA-01031: insufficient privileges

6. When the previous step fails, update table by using the UpdateTbl procedure:

```
SQL> EXEC procowner.UpdateTbl(1, 'new_value');
```

PL/SQL procedure successfully completed.

7. Check whether the table is updated:

```
SQL> select * from ernesto.tbl;
```

A	B
1	new_value

Creating and using invoker's right procedures

1. Connect to the database as a user with the DBA role (for example, ernesto):

```
SQL> connect ernesto
```

2. Create two users (procuser1, procuser2) and grant them privileges:

```
SQL> create user procuser1 identified by oracle1;  
SQL> create user procuser2 identified by oracle2;  
SQL> grant create session to procuser1;  
SQL> grant create session to procuser2;
```

3. Create the table table1 and grant select and update privileges on that table to procuser1 and only select privilege to procuser2:

```
SQL> create table table1(a number, b varchar2(30));  
SQL> insert into ernesto.table1 values(1, 'old_value');  
SQL> commit;  
SQL> grant select on ernesto.table1 to procuser1;  
SQL> grant update on ernesto.table1 to procuser1;  
SQL> grant select on ernesto.table1 to procuser2;
```

4. Create an invoker's rights procedure to update table1:

```
CREATE OR REPLACE PROCEDURE UpdateTable1 (x IN number,  
                                         y IN varchar2)  
AUTHID CURRENT_USER  
AS  
BEGIN  
    UPDATE ERNESTO.TABLE1  
    SET b = y  
    WHERE a = x;
```

```
END;
```

```
/
```

5. Grant execute on that procedure to procuser1 and procuser2:

```
SQL> grant execute on ernesto.UpdateTable1 to procuser1;
```

```
SQL> grant execute on ernesto.UpdateTable1 to procuser2;
```

6. Connect as user procuser1 and execute the procedure UpdateTable1:

```
SQL> connect procuser1
```

```
SQL> EXEC ernesto.UpdateTable1(1, 'new_value');
```

```
PL/SQL procedure successfully completed.
```

```
SQL> commit;
```

7. Check whether the table is updated:

```
SQL> select * from ernesto.table1;
```

A	B
1	new_value

1	new_value
---	-----------

8. Connect as the user procuser2 and try to execute the procedure UpdateTable1:

```
SQL> connect procuser2
```

```
SQL> EXEC ernesto.UpdateTable1(1, 'newer_value');
```

```
BEGIN ernesto.UpdateTable1(1, 'new_value'); END;
```

*

ERROR at line 1:

ORA-01031: insufficient privileges

ORA-06512: at "ERNESTO.UPDATETABLE1", line 5

ORA-06512: at line 1

Using code-based access control

1. Connect to the database as a user with a DBA role (for example, ernesto), create proc_user, and grant him the create session privilege:

SQL> create user proc_user identified by oracle1;

SQL> grant create session to proc_user;

2. Create table tbl1 and insert test data:

SQL> create table tbl1(a number, b varchar2(30));

SQL> insert into tbl1 values (1, 'old_value');

SQL> commit;

3. Create the invoker's rights procedure UpdateTbl1 and grant execute on that procedure to proc_user:

CREATE OR REPLACE PROCEDURE UpdateTbl1 (x IN number,

y IN varchar2)

AUTHID CURRENT_USER

```
AS  
BEGIN  
    UPDATE ERNESTO.TBL1  
        SET b = y  
    WHERE a = x;  
END;  
/
```

SQL> grant execute on ernesto.UpdateTbl1 to proc_user;

4. Create the role proc_role and grant update on tbl1 to proc_role:

```
SQL> create role proc_role;  
SQL> grant update on ernesto.tbl1 to proc_role;
```

5. Grant proc_role to the procedure UpdateTbl1:

```
SQL> grant proc_role to procedure ernesto.UpdateTbl1;
```

6. Connect as a user proc_user:

```
SQL> connect proc_user
```

7. Try to directly update the table:

```
SQL> update ernesto.tbl1 set b = 'value1' where a = 1;  
update ernesto.tbl1 set b = 'value1' where a = 1
```

*

ERROR at line 1:

ORA-00942: table or view does not exist

8. Execute the procedure UpdateTbl1:

```
SQL> execute ernesto.UpdateTbl1(1, 'new_value');
```

PL/SQL procedure successfully completed.

9. Connect as the user ernesto and verify whether the table is updated:

```
SQL> connect ernesto
```

```
SQL> select * from tbl1;
```

A	B
1	new_value

Restricting access to program units by using accessible by

1. Connect as a user who has the create procedure privilege (for example, ernesto):

```
SQL> connect ernesto
```

2. Create the protected_pkg package that is only accessible by public_pkg:

```
CREATE OR REPLACE PACKAGE protected_pkg
```

```
ACCESSIBLE BY (public_pkg)
```

```
IS  
PROCEDURE protected_proc;  
END;  
/  
CREATE OR REPLACE PACKAGE BODY protected_pkg  
IS  
PROCEDURE protected_proc  
IS  
BEGIN  
DBMS_OUTPUT.PUT_LINE ('This is a Protected Procedure  
that can only be accessed from Public Package');  
END;  
END;  
/
```

3. Create the public_pkg package:

```
CREATE OR REPLACE PACKAGE public_pkg  
IS  
PROCEDURE public_proc;  
END;  
/  
CREATE OR REPLACE PACKAGE BODY public_pkg  
IS
```

```
PROCEDURE public_proc
IS
BEGIN
    DBMS_OUTPUT.PUT_LINE ('This is Public Procedure from
    Public Package!');
    protected_pkg.protected_proc;
END;
/

```

4. Execute the `public_proc` procedure from `public_pkg`:

SQL> set serveroutput on

SQL> EXEC public_pkg.public_proc;

This is Public Procedure from Public Package!

This is a Protected Procedure that can only be accessed from
Public Package

PL/SQL procedure successfully completed.

5. Try to directly execute `protected_proc` from `protected_pkg` and observe the error:

SQL> EXEC protected_pkg.protected_proc;

BEGIN protected_pkg.protected_proc; END;

*

ERROR at line 1:

ORA-06550: line 1, column 7:

PLS-00904: insufficient privilege to access object

PROTECTED_PKG

ORA-06550: line 1, column 7:

PL/SQL: Statement ignored

6. Try to create another package that accesses protected_proc from protected_pkg:

CREATE OR REPLACE PACKAGE other_pkg

IS

PROCEDURE other_proc;

END;

/

CREATE OR REPLACE PACKAGE BODY other_pkg

IS

PROCEDURE other_proc

IS

BEGIN

DBMS_OUTPUT.PUT_LINE ('This is Other Procedure from

Other Package!');

protected_pkg.protected_proc;

END;

```
END;
```

```
/
```

Warning: Package Body created with compilation errors.

7. Find the compilation errors, as follows:

```
SQL> show errors
```

Errors for PACKAGE BODY OTHER_PKG:

LINE/COL ERROR

```
-----  
7/7      PL/SQL: Statement ignored
```

```
7/7      PLS-00904: insufficient privilege to access object
```

PROTECTED_PKG

Virtual Private Database

Creating different policy functions

1. Connect to the database as a user who has appropriate privileges (for example, user maja):

```
$ sqlplus maja
```

2. Create a policy function that satisfies this condition: The user susan can't access data in a table (for example, hr.emp_vpd_test) and other users can access entire data in the table.

Worksheet Query Builder

```
CREATE OR REPLACE FUNCTION no_access(
    schema_var IN VARCHAR2,
    table_var  IN VARCHAR2)
RETURN VARCHAR2
IS
    return_value VARCHAR2 (400);
BEGIN
    IF (SYS_CONTEXT('USERENV','SESSION_USER')) = 'SUSAN' THEN
        return_value := '1=2';
    ELSE
        return_value := '1=1';
    END IF;
    RETURN return_value;
END no_access;
/
```

Script Output X

| Task completed in 0.47 seconds

Function NO_ACCESS compiled

3. Create an application context that has the `emp_id` attribute and the value is `emp_id` (from the `hr.emp_vpd_test`) of the connected user or if the connected user is not employee.

Create an application context

Create a PL/SQL package

```

Worksheet | Query Builder
CREATE OR REPLACE PACKAGE hremp_ctx_pkg
IS
    PROCEDURE set_emp_id;
END;
/
CREATE OR REPLACE PACKAGE BODY hremp_ctx_pkg
IS
    PROCEDURE set_emp_id
    IS
        v_emp_id NUMBER;
    BEGIN
        SELECT emp_id
        INTO v_emp_id
        FROM hr.emp_vpd_test
        WHERE UPPER(email) = (SYS_CONTEXT('USERENV','SESSION_USER') || '@COMPANY.EXAMPLE.COM');
        DBMS_SESSION.SET_CONTEXT ('hremp_ctx','emp_id',v_emp_id);
    EXCEPTION
        WHEN no_data_found THEN
            DBMS_SESSION.SET_CONTEXT ('hremp_ctx','emp_id',0);
    END;
END;
/

```

Script Output x
Task completed in 0.115 seconds

Package HREMP_CTX_PKG compiled

Package body HREMP_CTX_PKG compiled

Create a logon trigger

```

SQL> CREATE OR REPLACE TRIGGER hremp_ctx_logon
  2  AFTER LOGON ON DATABASE
  3  BEGIN
  4      hremp_ctx_pkg.set_emp_id();
  5  END;
  6  /

```

Trigger created.

4. Create a policy function (for example, emp_access) that satisfies this condition: a "regular" employee can access only his or her data in a table (for example, hr.emp_vpd_test) and manager users can access his or her data in the table and data for employees he or she directly manages.

```

Worksheet | Query Builder
CREATE OR REPLACE FUNCTION emp_access(
    schema_var IN VARCHAR2,
    table_var IN VARCHAR2)
RETURN VARCHAR2
IS
    return_value VARCHAR2 (400);
BEGIN
    return_value:= '(emp_id = SYS_CONTEXT(''hremp_ctx'', ''emp_id'')) OR (mgr_id = SYS_CONTEXT(''hremp_ctx'', ''emp_id''))';
    RETURN return_value;
END emp_access;
/

```

Script Output x
Task completed in 0.123 seconds

Function EMP_ACCESS compiled

The emp_access policy function

5. Create a role (for example, HREMP_TEST).

6. Create a policy function that satisfies this condition: Only users who have the HREMP_TEST role can view data in a table (for example, hr.emp_vpd_test).

The screenshot shows the Oracle SQL Developer interface. The top window is titled "Worksheet" and contains the following PL/SQL code:

```
CREATE OR REPLACE FUNCTION role_access(
    schema_var IN VARCHAR2,
    table_var  IN VARCHAR2)
RETURN VARCHAR2
IS
    return_value VARCHAR2 (400);
BEGIN
    IF (SYS_CONTEXT('SYS_SESSION_ROLES','HREMP_TEST')) = 'TRUE' THEN
        return_value:= '1=1';
    ELSE
        return_value := '1=2';
    END IF;
    RETURN return_value;
END role_access;
/
```

The condition `IF (SYS_CONTEXT('SYS_SESSION_ROLES','HREMP_TEST')) = 'TRUE'` is highlighted in yellow. The bottom window is titled "Script Output" and displays the results of the compilation:

Function ROLE_ACCESS compiled | Task completed in 0.014 seconds

The role_access policy function

Creating Oracle Virtual Private Database row-level policies

1. Connect to the database as a user who has appropriate privileges (for example, the user maja):

\$ sqlplus maja

2. Create a VPD policy (for example, test_pol1) that protects the hr.emp_vpd_test table in the following way: it restricts SELECT operation based on a policy function (for example, no_access).

```
SQL> exec DBMS_RLS.ADD_POLICY('HR', 'EMP_VPD_TEST', 'TEST_POL1', 'MAJA', 'NO_ACCESS', 'SELECT')
```

```
PL/SQL procedure successfully completed.
```

3. To test VPD policy created in the previous step, connect as the user susan to the database (keep in mind that she has the SELECT ANY TABLE privilege) and try to access data in the table hr.emp_vpd_test.

```
SQL> connect susan
Enter password:
Connected.
SQL> SELECT * FROM HR.EMP_VPD_TEST;
no rows selected
```

Susan can't access data

4. Connect to the database as a user who can create a VPD policy (for example, user maja). Create a VPD policy (for example, test_pol2) that additionally protects the hr.emp_vpd_test table in the following way: it restricts the SELECT and DELETE operations based on a policy function (for example, emp_access).

```
SQL> connect maja
```

```
SQL> exec DBMS_RLS.ADD_POLICY('HR', 'EMP_VPD_TEST', 'TEST_POL2', 'MAJA', 'EMP_ACCESS', 'SELECT,DELETE')
```

```
PL/SQL procedure successfully completed.
```

The VPD policy TEST_POL2

5. Connect to the database as the user joel and execute the following query:

```
SELECT * FROM HR.EMP_VPD_TEST;
```

The result will show 3 rows, because joel can view his data and data for his direct employees (policy function emp_access).

```

SQL> connect joel
Enter password:
Connected.
SQL> select * from hr.emp_vpd_test;

        EMP_ID FIRST_NAME                      LAST_NAME
----- EMAIL                               SALARY   COMM_PCT    MGR_ID
----- 1 Maja                            Veselica      11000     .05       3
maja@company.example.com
----- 2 ernesto                          Pavlovic      11500     .02       3
ernesto@company.example.com
----- 3 Joel                            Adams        15000     .04
joel@company.example.com

```

Joel can view his data and data for his direct employees

6. Connect to the database as the user emma and execute the following query:

SELECT * FROM HR.EMP_VPD_TEST;

The result will show only 1 row, because emma is a "regular" employee, so she can view only her own data (policy function emp_access).

```

SQL> connect emma
Enter password:
Connected.
SQL> select * from hr.emp_vpd_test;

        EMP_ID FIRST_NAME                      LAST_NAME
----- EMAIL                               SALARY   COMM_PCT    MGR_ID
----- 4 Emma                            Cole        8000      .1       5
emma@company.example.com

```

Emma can only view her own data

Creating column-level policies

1. Connect to the database as a user who has appropriate privileges (for example, the user *maja*):

```
$ sqlplus maja
```

2. Create a VPD policy (for example, *test_col*) that protects the *hr.emp_vpd_test* table in the following way: it defines that *salary* and *comm_pct* are sensitive columns and a user can access them only if he or she has the *HREMP_TEST* role (the *role_access* policy function).

The screenshot shows the Oracle SQL Developer interface. The top tab bar has 'Worksheet' selected. The main area contains a PL/SQL block:

```
BEGIN
  DBMS_RLS.ADD_POLICY(object_schema => 'HR',
  object_name => 'EMP_VPD_TEST',
  policy_name => 'TEST_COL',
  function_schema => 'MAJA',
  policy_function => 'ROLE_ACCESS',
  statement_types => 'SELECT',
  sec_relevant_cols => 'SALARY,COMM_PCT',
  sec_relevant_cols_opt => DBMS_RLS.ALL_ROWS);
END;
/
```

Below the code, the 'Script Output' tab is visible, showing the results of the execution:

```
Script Output X
Task completed in 0.15 seconds
PL/SQL procedure successfully completed.
```

3. Grant the role *HREMP_TEST* to user *ernesto*:

```
SQL> grant HREMP_TEST to ernesto;
```

4. Connect to the database as the user *ernesto* and view data in the table *hr.emp_vpd_test*.

```
SQL> grant HREMP_TEST to ernesto
Grant succeeded.

SQL> connect ernesto
Enter password:
Connected.
SQL> select * from hr.emp_vpd_test;

  EMP_ID FIRST_NAME          LAST_NAME
-----  -----
EMAIL           SALARY    COMM_PCT   MGR_ID
-----  -----
2 ernesto          Pavlovic
ernesto@company.example.com      11500       .02        3
```

5. Connect to the database as the user `maja` and disable the VPD policy `TEST_POL2`.

```
SQL> connect maja
Enter password:
Connected.
SQL> exec DBMS_RLS.ENABLE_POLICY('HR','EMP_VPD_TEST','TEST_POL2',FALSE);
PL/SQL procedure successfully completed.
```

6. Repeat step 4.

```

SQL> connect ernesto
Enter password:
Connected.
SQL> select * from hr.emp_vpd_test;

        EMP_ID FIRST_NAME                      LAST_NAME
-----  -----
EMAIL                               SALARY   COMM_PCT    MGR_ID
-----  -----
1 Maja                            Veselica
maja@company.example.com          11000      .05         3
2 ernesto                          Pavlovic
ernest01@company.example.com     11500      .02         3
3 Joel                            Adams
joel@company.example.com         15000      .04         0

        EMP_ID FIRST_NAME                      LAST_NAME
-----  -----
EMAIL                               SALARY   COMM_PCT    MGR_ID
-----  -----
4 Emma                            Cole
emma@company.example.com         8000       .1          5
5 Susan                            Smith
susan@company.example.com        16000       0          0

```

SQL> █

7. Connect to the database as the user joel and execute the same statement as in the previous step.

```

SQL> connect joel
Enter password:
Connected.
SQL> select * from hr.emp_vpd_test;

EMP_ID FIRST_NAME          LAST_NAME
-----  -----
EMAIL
-----  -----
1 Maja Veselica
maja@company.example.com      3
2 ernesto Pavlovic
ernesto@company.example.com   3
3 Joel Adams
joel@company.example.com

EMP_ID FIRST_NAME          LAST_NAME
-----  -----
EMAIL
-----  -----
4 Emma Cole
emma@company.example.com     5
5 Susan Smith
susan@company.example.com

```

SQL> ■

Creating a driving context

1. *Connect to the database as a user who has appropriate privileges (for example, the user majा):*

\$ sqlplus majа

2. *Create a driving context (for example, driver_ctx):*

SQL> CREATE CONTEXT driver_ctx using driver_ctx_pkg;

3. Set the driving context:

```
SQL> CREATE OR REPLACE PACKAGE driver_ctx_pkg IS
```

```
    PROCEDURE set_driver (p_group varchar2);
```

```
END;
```

```
/
```

```
SQL> CREATE OR REPLACE PACKAGE BODY driver_ctx_pkg IS
```

```
    PROCEDURE set_driver (p_group varchar2)
```

```
    IS
```

```
    BEGIN
```

```
        DBMS_SESSION.SET_CONTEXT('driver_ctx','ACTIVE',p_group);
```

```
    END;
```

```
END;
```

```
/
```

Creating policy groups

1. Connect to the database as a user who has appropriate privileges (for example, the user maja):

```
$ sqlplus maja
```

2. Create the first policy group (for example, pol_grp_A):

```
SQL> BEGIN
```

```
DBMS_RLS.CREATE_POLICY_GROUP(
```

```
object_schema => 'HR',
object_name => 'EMP_VPD_TEST',
policy_group => 'pol_grp_A');

END;

/
```

3. Create the second policy group (for example, pol_grp_B):

```
SQL> BEGIN

DBMS_RLS.CREATE_POLICY_GROUP(
object_schema => 'HR',
object_name => 'EMP_VPD_TEST',
policy_group => 'pol_grp_B');

END;

/
```

Setting context as a driving context

1. Connect to the database as a user who has appropriate privileges (for example, the user maja):

```
$ sqlplus maja
```

2. Make an existing application context a driving context.

```

SQL> connect maja
Enter password:
Connected.
SQL> BEGIN
 2  DBMS_RLS.ADD_POLICY_CONTEXT('HR','EMP_VPD_TEST','DRIVER_CTX','ACTIVE');
 3 END;
 4 /
PL/SQL procedure successfully completed.

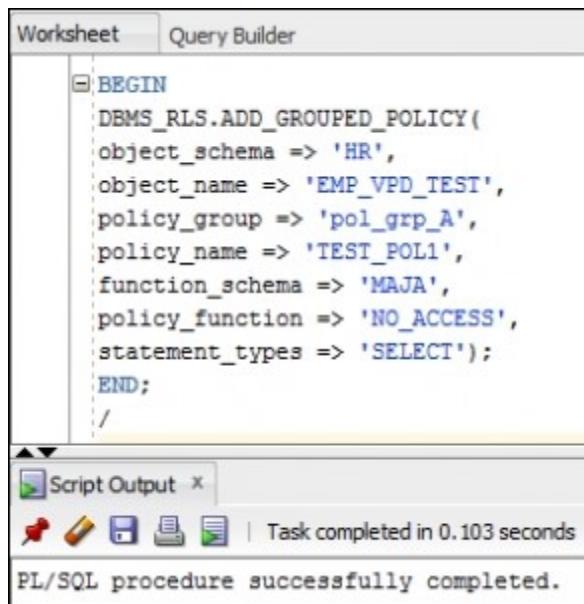
```

Adding policy to a group

1. Connect to the database as a user who has appropriate privileges (for example, the user maja):

\$ sqlplus maja

2. Add **TEST_POL1** to policy group **pol_grp_A**.



The screenshot shows the Oracle SQL Developer interface with the 'Worksheet' tab selected. The code area contains the following PL/SQL block:

```

BEGIN
  DBMS_RLS.ADD_GROUPED_POLICY(
    object_schema => 'HR',
    object_name => 'EMP_VPD_TEST',
    policy_group => 'pol_grp_A',
    policy_name => 'TEST_POL1',
    function_schema => 'MAJA',
    policy_function => 'NO_ACCESS',
    statement_types => 'SELECT');
END;
/

```

The 'Script Output' window at the bottom shows the message "Task completed in 0.103 seconds" and "PL/SQL procedure successfully completed."

3. Add **TEST_COL** to policy group **pol_grp_A**.

Worksheet Query Builder

```
BEGIN
DBMS_RLS.ADD_GROUPED_POLICY(
object_schema => 'HR',
object_name => 'EMP_VPD_TEST',
policy_group => 'pol_grp_A',
policy_name => 'TEST_COL',
function_schema => 'MAJA',
policy_function => 'ROLE_ACCESS',
statement_types => 'SELECT',
sec_relevant_cols => 'SALARY,COMM_PCT',
sec_relevant_cols_opt => DBMS_RLS.ALL_ROWS);
END;
/

```

Script Output x

Task completed in 0.008 seconds

PL/SQL procedure successfully completed.

4. Add TEST_POL2 to policy group pol_grp_B.

Worksheet Query Builder

```
BEGIN
DBMS_RLS.ADD_GROUPED_POLICY(
object_schema => 'HR',
object_name => 'EMP_VPD_TEST',
policy_group => 'pol_grp_B',
policy_name => 'TEST_POL2',
function_schema => 'MAJA',
policy_function => 'EMP_ACCESS',
statement_types => 'SELECT, DELETE');
END;
/

```

Script Output x

Task completed in 0.008 seconds

PL/SQL procedure successfully completed.

5. Create a logon trigger.

Worksheet | Query Builder

```

CREATE OR REPLACE TRIGGER driver_ctx_logon
    AFTER LOGON ON DATABASE
BEGIN
    IF (SYS_CONTEXT('USERENV','CLIENT_PROGRAM_NAME')) = 'SQL Developer'
        THEN DRIVER_CTX_PKG.set_driver('pol_grp_A');
        else DRIVER_CTX_PKG.SET_DRIVER('pol_grp_B');
        end if;
    END;
/

```

Script Output X | Query Result X

Task completed in 0.209 seconds

Trigger DRIVER_CTX_LOGON compiled

6. Connect to the database as the user joel using SQL*Plus and execute the SELECT statement, as shown:

```

SQL> select sys_context('driver_ctx', 'ACTIVE') from dual;
SYS_CONTEXT('DRIVER_CTX', 'ACTIVE')
-----
pol_grp_B

```

7. View data in the table hr.emp_vpd_test.

```

SQL> connect joel
Enter password:
Connected.
SQL> select * from hr.emp_vpd_test;

      EMP_ID FIRST_NAME          LAST_NAME
----- EMAIL           SALARY   COMM_PCT   MGR_ID
----- 1 Maja          Veselica
maja@company.example.com       11000     .05        3
2 Ernesto         Pavlovic
ernesto@company.example.com    11500     .02        3
3 Joel            Adams
joel@company.example.com      15000     .04

```

8. Connect to the database as the user susan using SQL*Plus and view data in the table hr.emp_vpd_test:

SQL> connect susan

```
SQL> connect joel
Enter password:
Connected.
SQL> select * from hr.emp_vpd_test;

  EMP_ID FIRST_NAME          LAST_NAME
----- EMAIL           SALARY   COMM_PCT   MGR_ID
----- 1 Maja           Veselica
maja@company.example.com      11000     .05        3
2 Ernesto          Pavlovic
ernesto@company.example.com    11500     .02        3
3 Joel             Adams
joel@company.example.com      15000     .04        4
```

9. Connect as the user emma using SQL Developer and view data in the table hr.emp_vpd_test.

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a tab labeled "Worksheet". Below it, the "Query Builder" tab is active. A SQL query is entered in the worksheet:

```
select * from hr.emp_vpd_test;
```

In the bottom-right corner, there's a tab labeled "Query Result". It displays the results of the executed query. The results are presented in a grid format:

	EMP_ID	FIRST_NAME	LAST_NAME	EMAIL	SALARY	COMM_PCT	MGR_ID
1	1 Maja	Veselica	maja@company.example.com	(null)	(null)	(null)	3
2	2 Ernesto	Pavlovic	ernesto@company.example.com	(null)	(null)	(null)	3
3	3 Joel	Adams	joel@company.example.com	(null)	(null)	(null)	4
4	4 Emma	Cole	emma@company.example.com	(null)	(null)	(null)	5
5	5 Susan	Smith	susan@company.example.com	(null)	(null)	(null)	(null)

Below the grid, a status message indicates: "All Rows Fetched: 5 in 0.022 seconds".

Exempting users from VPD policies

1. Connect to the database as SYS user:

\$ sqlplus / as sysdba

2. Grant the EXEMPT ACCESS POLICY privilege to an existing user (for example, susan):

SQL> grant EXEMPT ACCESS POLICY to susan;

3. Connect to the database as the user susan and verify that now she can access data in the hr.emp_vpd_test table.

```
SQL> connect susan
Enter password:
Connected.
SQL> select * from hr.emp_vpd_test;

      EMP_ID FIRST_NAME          LAST_NAME
----- EMAIL           SALARY   COMM_PCT    MGR_ID
----- 1 Maja          Veselica
maja@company.example.com     11000      .05        3
                               2 Ernesto         Pavlovic
ernesto@company.example.com  11500      .02        3
                               3 Joel            Adams
joel@company.example.com    15000      .04
                               4 Emma            Cole
emma@company.example.com    8000       .1         5
                               5 Susan           Smith
susan@company.example.com   16000      0
```

Data Redaction

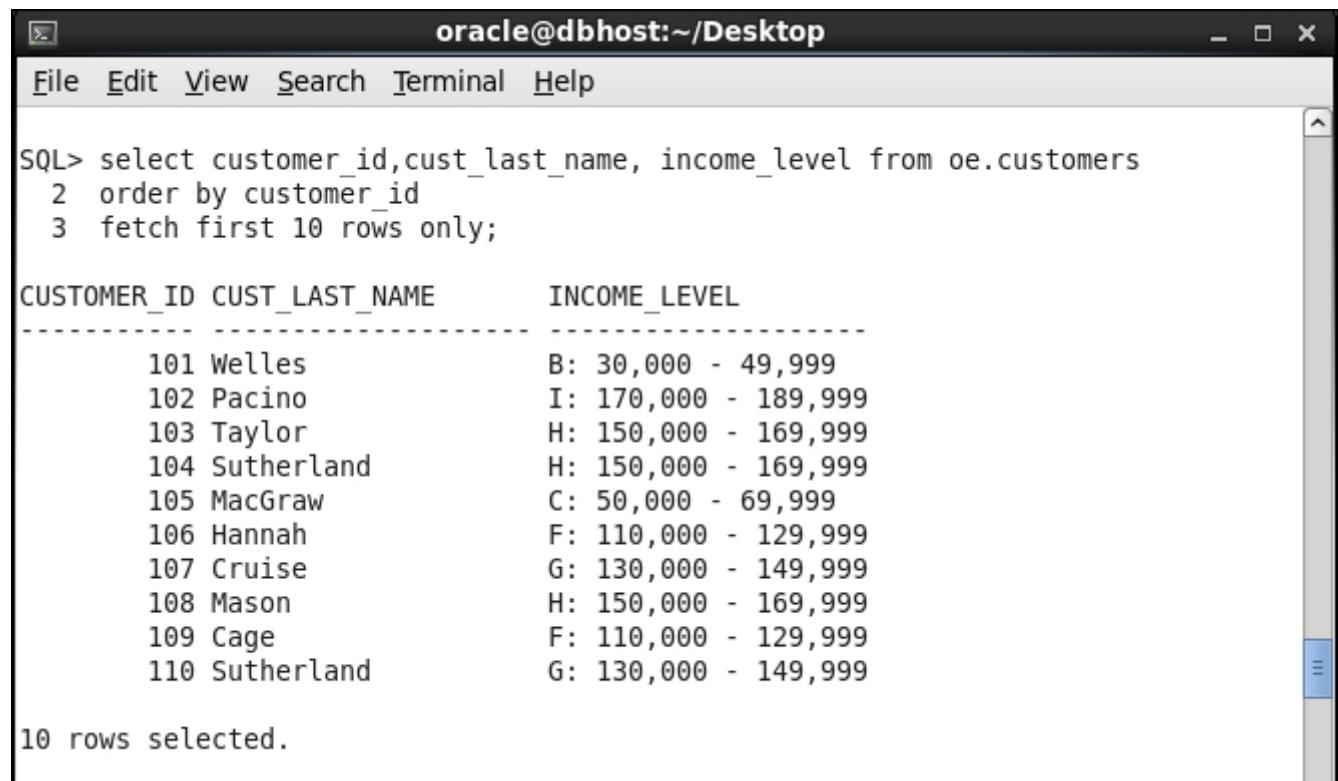
Creating a redaction policy when using full redaction

- 1. Connect to the database as a user who has the SELECT privilege on the OE.CUSTOMERS table or the SELECT ANY TABLE privilege (for example, the oe user):*

```
$ sqlplus oe
```

- 2. Verify that the user (for example, the user oe) can view data by executing the following query:*

```
select customer_id, cust_last_name, income_level from      oe.customers  
order by customer_id fetch first 10 rows    only;
```



The screenshot shows a terminal window titled "oracle@dbhost:~/Desktop". The menu bar includes File, Edit, View, Search, Terminal, and Help. The main area displays the following SQL query and its results:

```
SQL> select customer_id,cust_last_name, income_level from oe.customers  
  2 order by customer_id  
  3 fetch first 10 rows only;  
  
CUSTOMER_ID CUST_LAST_NAME      INCOME_LEVEL  
-----  
101 Welles          B: 30,000 - 49,999  
102 Pacino          I: 170,000 - 189,999  
103 Taylor          H: 150,000 - 169,999  
104 Sutherland      H: 150,000 - 169,999  
105 MacGraw          C: 50,000 - 69,999  
106 Hannah           F: 110,000 - 129,999  
107 Cruise           G: 130,000 - 149,999  
108 Mason            H: 150,000 - 169,999  
109 Cage              F: 110,000 - 129,999  
110 Sutherland      G: 130,000 - 149,999  
  
10 rows selected.
```

Data in the clear text format (before redaction) in the OE.CUSTOMERS table

3. Connect to the database as a user who can create the user secmgr (who will be responsible for managing redaction policies) and grant him appropriate privileges (for example, SYS):

```
SQL> create user secmgr identified by oracle;  
SQL> grant create session to secmgr;  
SQL> grant execute on dbms_redact to secmgr;
```

4. Connect to the database as the secmgr user:

```
SQL> connect secmgr/oracle
```

5. Create the redaction policy CUST_POL in such a manner that data in the column income_level (the table oe.customers) is redacted using full redaction:

```
SQL> begin 2 dbms_redact.add_policy 3 (object_schema =>  
'OE', 4 object_name => 'CUSTOMERS', 5 policy_name =>  
'CUST_POL', 6 column_name =>  
'INCOME_LEVEL', 7 function_type => DBMS_REDACT.FULL, 8  
expression => '1=1'); 9 end; 10 /  
PL/SQL procedure successfully completed.
```

6. Connect to the database as the same user as in step 1 (for example, oe) and execute the same query as in step 2.

```
oracle@dbhost:~/Desktop
File Edit View Search Terminal Help
SQL> connect oe
Enter password:
Connected.
SQL> select customer_id,cust_last_name, income_level from oe.customers
  2  order by customer_id
  3  fetch first 10 rows only;
CUSTOMER_ID CUST_LAST_NAME      INCOME_LEVEL
----- -----
101 Welles
102 Pacino
103 Taylor
104 Sutherland
105 MacGraw
106 Hannah
107 Cruise
108 Mason
109 Cage
110 Sutherland
10 rows selected.

SQL>
```

After applying the redaction policy

Creating a redaction policy when using partial redaction

1. Log in to database as a user who has a DBA role (for instance, ernesto):

\$ sqlplus ernesto/oracle

2. Create a test table and insert some data in it:

SQL> create table tbl (a number);

SQL> insert into tbl values (123456);

SQL> insert into tbl values (234567);

SQL> insert into tbl values (345678);

SQL> commit;

3. Create role (that is going to be used in redaction policy) and user usr1 as the first test user:

SQL> create role myrole;

SQL> create user usr1 identified by oracle1;

SQL> grant create session to usr1;

4. Grant the select privilege and role to usr1:

SQL> grant select on ernesto.tbl to usr1;

SQL> grant myrole to usr1;

5. Create the second test user and grant him create session and select privilege, but don't grant him the role myrole:

SQL> create user usr2 identified by oracle2;

SQL> grant create session to usr2;

SQL> grant select on ernesto.tbl to usr2;

6. Create redaction policy to redact column a of the type Number using partial redaction (first four digits will be redacted and won't be seen at all). This redaction policy will be applied only to users that don't have role myrole and don't have the EXEMPT REDACTION POLICY privilege:

```
SQL> BEGIN
  2 DBMS_REDACT.ADD_POLICY(
  3 object_schema    => 'ernesto',
  4 object_name      => 'tbl',
  5 column_name      => 'a',
  6 column_description => 'Sensitive column A',
```

```
7 policy_name      => 'a_tbl_partial',
8 policy_description => 'Redact column A of tbl',
9 function_type     => DBMS_REDACT.PARTIAL,
10 function_parameters => '0,1,4',
11 expression        => 'SYS_CONTEXT(
"SYS_SESSION_ROLES",
"MYROLE") =
"FALSE"); 12 END; 13 /
```

7. Connect to database as the user usr1 and select from the table tbl in the schema ernesto:

SQL> connect usr1/oracle1

SQL> select a from ernesto.tbl;
A ----- 123456 234567 345678

8. Now, connect to database as the user usr2 and again select from the table tbl in the schema ernesto:

SQL> connect usr2/oracle2

usr2@ORA12CR1> select a from ernesto.tbl;
A ----- 56 67 78

9. Log in to database as a user who has a DBA role (for instance, ernesto):

\$ sqlplus ernesto/oracle

10. Create the test table to store credit cards data and insert some data in it:

SQL> create table customers (name varchar2(20 CHAR), credit_card varchar2(20 CHAR));

SQL> insert into customers values ('tom', '3455647456589132');

SQL> insert into customers values ('steve', '3734982321225691');

```
SQL> insert into customers values ('john',      '3472586894975806');  
SQL> commit;
```

11. Grant select privilege on table customers in the schema ernesto to usr1:

```
SQL> grant select on ernesto.customers to usr1;
```

12. Create a redaction policy to redact column credit_card of type Varchar2 using partial redaction (first 12 values will be redacted with #sign). This redaction policy will be applied to all users, except those who have the EXEMPT REDACTION POLICY privilege (see the Exempting users from data redaction policies recipe):

```
SQL> BEGIN  
 2 DBMS_REDACT.ADD_POLICY(  
 3 object_schema      => 'ernesto',  
 4 object_name         => 'customers',  
 5 column_name         => 'credit_card',  
 6 column_description  => 'Credit Card numbers',  
 7 policy_name         => 'CCN_POLICY',  
 8 policy_description   => 'Redact column  
credit_card of table          customers', 9 function_type  
=> DBMS_REDACT.PARTIAL, 10 function_parameters  =>  
'VVVVVVVVVVVVVVVVVV,  
VVVVVVVVVVVVVVVVVV, #, 1,           12', 11 expression  
=> '1=1'); 12 END; 13 /
```

13. Connect to database as the user usr1 and select from the table customers in the schema ernesto:

```
SQL> connect usr1/oracle1
```

```
SQL> select * from ernesto.customers;
```

NAME	CREDIT_CARD
------	-------------

tom #####9132 steve #####5691 john
#####5806

Creating a redaction policy when using random redaction

- 1. Connect to the database as a user who has the SELECT privilege on the HR.EMPLOYEES table or the SELECT ANY TABLE privilege (for example, hr user):*

```
$ sqlplus hr
```

- 2. Verify that the user (for example, hr user) can view data by executing the following query:*

```
select employee_id, salary, commission_pct from    hr.employees where  
commission_pct IS NOT NULL order by    employee_id fetch first 10  
rows only;
```

```

oracle@dbhost:~/Desktop
File Edit View Search Terminal Help
SQL> select employee_id, salary, commission_pct from hr.employees
  2 where commission_pct IS NOT NULL
  3 order by employee_id
  4 fetch first 10 rows only;

EMPLOYEE_ID      SALARY  COMMISSION_PCT
-----          -----
        145       14000          .4
        146       13500          .3
        147       12000          .3
        148       11000          .3
        149       10500          .2
        150       10000          .3
        151       9500           .25
        152       9000           .25
        153       8000           .2
        154       7500           .2

10 rows selected.

SQL>

```

Data in the clear text format in the HR.EMPLOYEES table

3. Connect to the database as the secmgr user:

SQL> connect secmgr/oracle

4. Create the redaction policy *EMP_POL* in such a way that data in column salary (the table hr.employees) is redacted using random redaction only when user in step 1 (for example, hr) tries to view it. If you don't use the hr user, modify line 8 to reflect that change:

SQL> begin
 2 dbms_redact.add_policy
 3 (object_schema => 'HR',
 4 object_name => 'EMPLOYEES',
 5 policy_name => 'EMP_POL',
 6 column_name => 'SALARY',
 7 function_type => DBMS_REDACT.RANDOM,

```
8 expression => 'SYS_CONTEXT("USERENV",
"SESSION_USER") = "HR"');
9 end;
10 /
```

PL/SQL procedure successfully completed.

5. Connect to the database as the same user as in step 1 (for example, hr) and execute the same query, as in step 2, twice.

```
oracle@dbhost:~/Desktop
File Edit View Search Terminal Help
SQL> connect hr
Enter password:
Connected.
SQL> select employee_id, salary, commission_pct from hr.employees
  2  where commission_pct IS NOT NULL
  3  order by employee_id
  4  fetch first 10 rows only;

EMPLOYEE_ID      SALARY COMMISSION_PCT
-----  -----
145          4756          .4
146         12122          .3
147         10270          .3
148           412          .3
149          7371          .2
150          2197          .3
151          2478          .25
152          4422          .25
153          4453          .2
154          1731          .2

10 rows selected.

SQL> select employee_id, salary, commission_pct from hr.employees
  2  where commission_pct IS NOT NULL
  3  order by employee_id
  4  fetch first 10 rows only;

EMPLOYEE_ID      SALARY COMMISSION_PCT
-----  -----
145          1729          .4
146            54          .3
147         2311          .3
148         9038          .3
149         3040          .2
150         9835          .3
151         7076          .25
152         6991          .25
153         2474          .2
154         1050          .2

10 rows selected.
```

After applying redaction policy

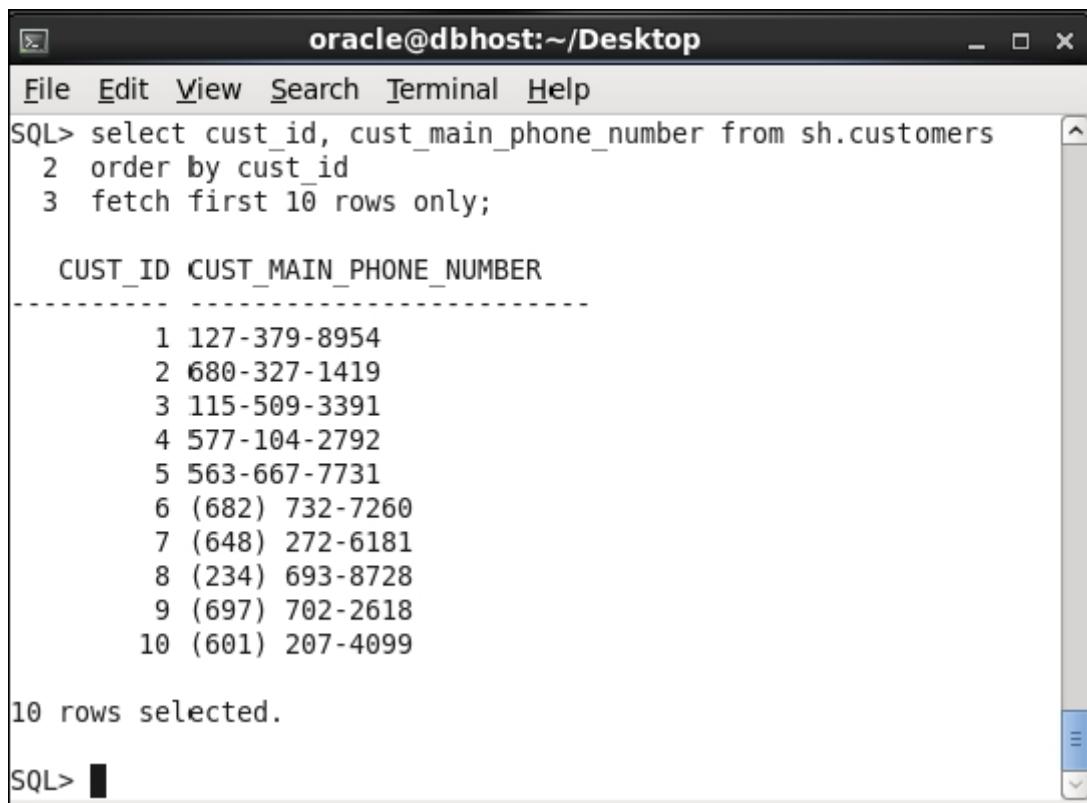
Creating a redaction policy when using regular expression redaction

1. Connect to the database as a user who has the SELECT privilege on the SH.CUSTOMERS table or the SELECT ANY TABLE privilege (for example, the sh user):

```
$ sqlplus sh
```

2. Verify that the user (for example, the user sh) can view data by executing the following query:

```
select cust_id, cust_main_phone_number from sh.customers      order by
cust_id fetch first 10 rows only;
```



The screenshot shows a terminal window titled "oracle@dbhost:~/Desktop". The window contains the following SQL command and its output:

```
File Edit View Search Terminal Help
SQL> select cust_id, cust_main_phone_number from sh.customers
  2 order by cust_id
  3 fetch first 10 rows only;

 CUST_ID CUST_MAIN_PHONE_NUMBER
-----
 1 127-379-8954
 2 680-327-1419
 3 115-509-3391
 4 577-104-2792
 5 563-667-7731
 6 (682) 732-7260
 7 (648) 272-6181
 8 (234) 693-8728
 9 (697) 702-2618
10 (601) 207-4099

10 rows selected.

SQL> ■
```

The output displays 10 rows of data from the SH.CUSTOMERS table, showing the CUST_ID and CUST_MAIN_PHONE_NUMBER for each customer.

Data in the clear text format (before redaction) in the SH.CUSTOMERS table

3. Connect to the database as the secmgr user:

```
SQL> connect secmgr/oracle
```

4. Create the redaction policy *SHORT_POL* in such a manner that data in the column *cust_main_phone_number* (the table *sh.customers*) is redacted using regular expression redaction:

```
SQL> begin
  2 dbms_redact.add_policy
  3 (object_schema => 'SH',
  4 object_name => 'CUSTOMERS',
  5 policy_name => 'SHORT_POL',
  6 column_name => 'CUST_MAIN_PHONE_NUMBER',
  7 function_type => DBMS_REDACT.REGEXP,
  8 expression => '1=1',
  9 regexp_pattern => DBMS_REDACT.RE_PATTERN_US_PHONE,
 10 regexp_replace_string => DBMS_REDACT.
RE_REDACT_US_PHONE_L7,
 11 regexp_position => DBMS_REDACT.RE_BEGINNING,
 12 regexp_occurrence => DBMS_REDACT.RE_FIRST);
 13 end;
 14 /
```

PL/SQL procedure successfully completed.

5. Connect to the database as the same user as in step 1 (for example, *sh*) and execute the same query as in step 2.

```
oracle@dbhost:~/Desktop
File Edit View Search Terminal Help
SQL> col cust_main_phone_number format A22
SQL> select cust_id, cust_main_phone_number from sh.customers
2 order by cust_id
3 fetch first 10 rows only;

CUST_ID CUST_MAIN_PHONE_NUMBER
-----
1 127-XXX-XXXX
2 680-XXX-XXXX
3 115-XXX-XXXX
4 577-XXX-XXXX
5 563-XXX-XXXX
6 T
7 T
8 T
9 T
10 T

10 rows selected.

SQL>
```

After applying the redaction policy

Using Oracle Enterprise Manager Cloud Control 12c to manage redaction policies

- 1. Log in to Oracle Enterprise Manager Cloud Control at <https://hostname:port/em>.*
- 2. Go to a Database home page (if it is a container database, you should go to a home page of PDB that contains sample schemas).*
- 3. On menu, select Security | Data Redaction.*

ORACLE Enterprise Manager Cloud Control 12c

Enterprise Targets Favorites History

dbm / CUST2

Oracle Database Performance Availability Security Schema Administration

Summary

Status

Up Time 1 days, 13 hrs
Version 12.1.0.2.0
Available Space N/A

Diagnostics

Incidents 0 0 0 0

Compliance Summary (Brief)

View Trends Compliance Standard

No data to display

Home Reports

Users Roles Profiles Audit Settings

Enterprise Data Governance Application Data Models

Configuration Compliance

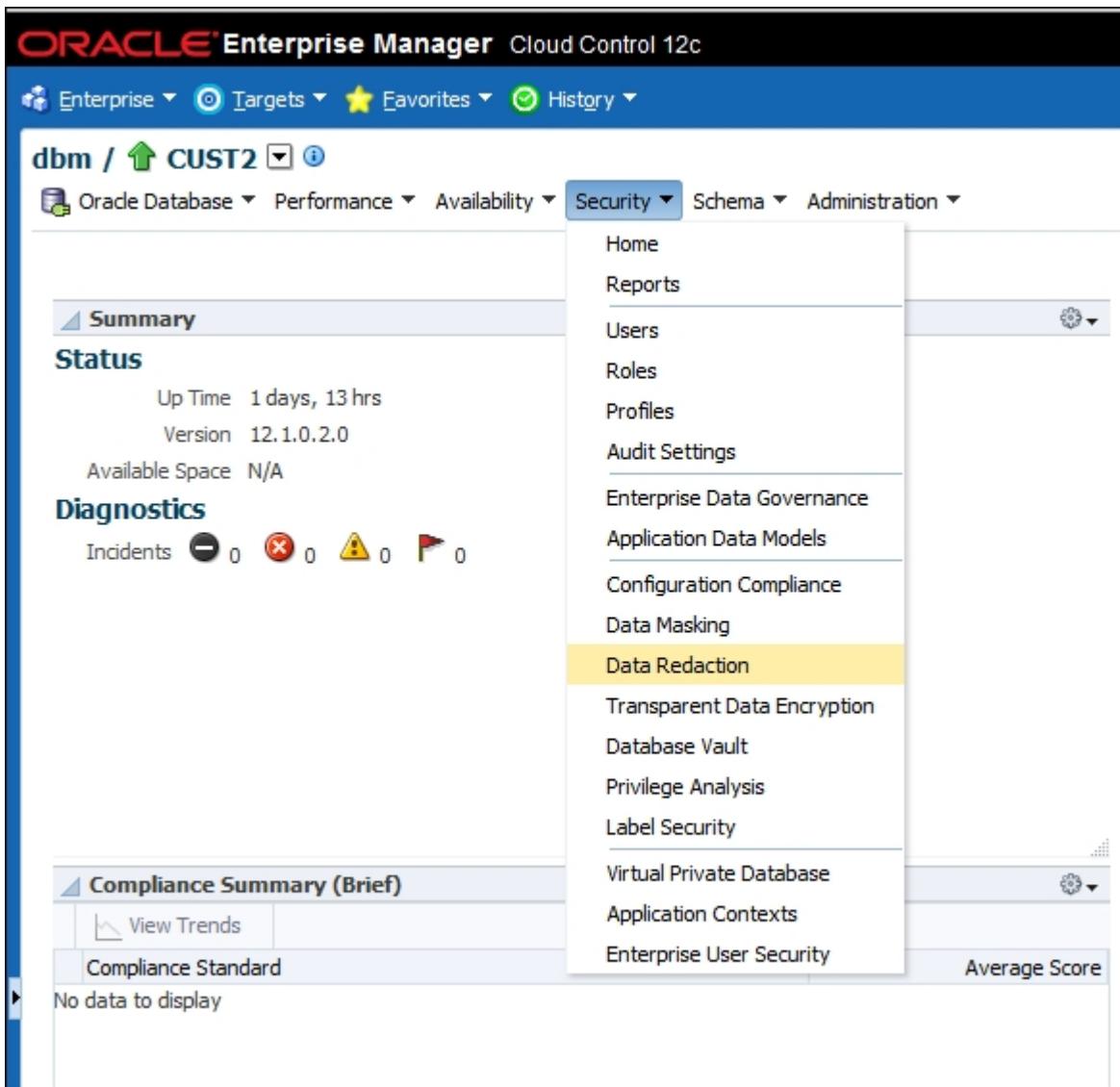
Data Masking

Data Redaction

Transparent Data Encryption Database Vault Privilege Analysis Label Security

Virtual Private Database Application Contexts Enterprise User Security

Average Score

The screenshot shows the Oracle Enterprise Manager interface for Cloud Control 12c. The main navigation bar includes links for Enterprise, Targets, Favorites, and History. Below the bar, the current database is identified as dbm / CUST2. A secondary navigation bar shows Oracle Database, Performance, Availability, Security (which is currently selected), Schema, and Administration. On the left, there are three main sections: Status (with uptime, version, and space info), Diagnostics (showing zero incidents), and Compliance Summary (Brief) (which is empty). A vertical sidebar on the right contains a navigation menu. The 'Data Redaction' option is highlighted with a yellow background, indicating it is the active selection.

Select Data Redaction.

4. On the Data Redaction screen, select Create

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. The top navigation bar includes links for Enterprise, Targets, Favorites, and History. Below the navigation is a breadcrumb trail: dbm / CUST2. The main menu has options for Oracle Database, Performance, Availability, Security, Schema, and Administration. The current section is "Data Redaction". A sub-section titled "Search Data Redaction Policies" contains three search fields: Schema (%), Table/View (%), and Policy Name (%), followed by a "Go" button. Below this is a table titled "Data Redaction Policies" with columns: Schema, Table/View, Policy Name, Enabled, and Redacted Columns. A "Create" button is highlighted in yellow at the top left of the table area.

Creating a redaction policy

5. Set Schema as HR and the table as EMPLOYEES. Enter SAL_POLICY as a policy name. Click on the Add button, to add column that is going to be redacted.

ORACLE Enterprise Manager Cloud Control 12c

Enterprise Targets Favorites History

dbm / CUST2

Oracle Database Performance Availability Security Schema Administration

Create Data Redaction Policy: SAL_POLICY

* Schema: HR

* Table/View: EMPLOYEES

* Policy Name: SAL_POLICY

1=1

* Policy Expression

Object Columns

Add	Modify	Remove	
Column	Column Datatype	Redaction Function	Function Attributes

The addition of a column

6. Select the SALARY column and specify RANDOM as a Redaction Function. Click on OK. On the next screen, click on OK at the top-right corner.

Add

* Column	SALARY	<input type="button" value=""/>
* Column Datatype	NUMBER	
Redaction Template	Custom	<input type="button" value=""/>
* Redaction Function	RANDOM	<input type="button" value=""/>
Random Redaction. The redacted data presented to the querying user appears as randomly-generated values each time it is displayed, depending on the data type of the column.		
		<input type="button" value="OK"/> <input type="button" value="Cancel"/>

Choose random redaction type

7. To edit SAL_POLICY, select it and click on Edit (you can search for policies by specifying schema, table, or policy name)

ORACLE Enterprise Manager Cloud Control 12c

Enterprise Targets Favorites History

dbm / **CUST2**

Oracle Database Performance Availability Security Schema Administration

Data Redaction

Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications

Search Data Redaction Policies

Schema %
Table/View %
Policy Name %
<input type="button" value="Go"/>

Data Redaction Policies

<input type="button" value="Create"/>	<input type="button" value="Edit"/>	<input type="button" value="View"/>	<input type="button" value="Enable"/>	<input type="button" value="Disable"/>	<input type="button" value="Delete"/>
Schema	Table/View	Policy Name	Enabled	Redacted Columns	
HR	EMPLOYEES	SAL_POLICY	<input checked="" type="checkbox"/>	1	

Alter policy

8. Select the SALARY column and click on Modify.

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. The top navigation bar includes links for Enterprise, Targets, Favorites, and History. Below the navigation is a breadcrumb path: dbm / CUST2. The main menu has options for Oracle Database, Performance, Availability, Security, Schema, and Administration. The current page is titled "Edit Data Redaction Policy: SAL_POLICY".
Form fields on the left:

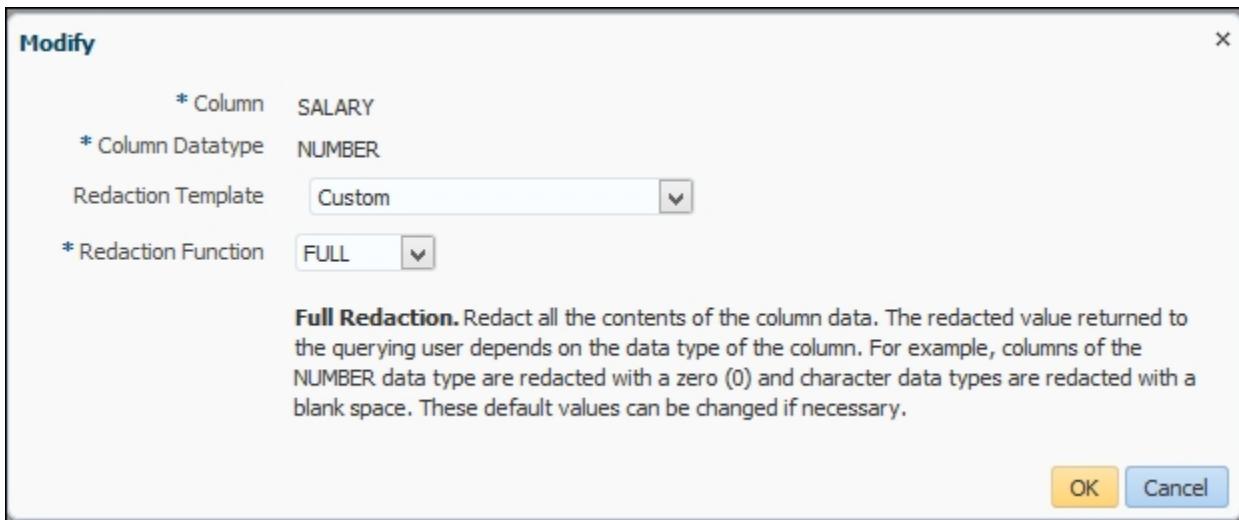
- * Schema: HR
- * Table/View: EMPLOYEES
- * Policy Name: SAL_POLICY
- * Policy Expression: 1=1

A large text area labeled "Policy Expression" contains the value "1=1".
Below the policy expression is a section titled "Object Columns". It contains a table with the following data:

Column	Column Datatype	Redaction Function	Function Attributes
SALARY	NUMBER	RANDOM	

Modifying a column

9. Change Redaction Function from RANDOM to FULL. Click on OK



Changing redaction type for salary column

10. Click on Add in order to add one more column to the redaction policy.

Create Data Redaction Policy: SAL_POLICY

* Schema HR
 * Table/View EMPLOYEES
 * Policy Name SAL_POLICY

1=1

* Policy Expression

Object Columns

Add	Modify	Remove	
Column	Column Datatype	Redaction Function	Function Attributes
SALARY	NUMBER	FULL	

Adding a column to the redaction policy

11. Select the EMAIL column, and as Redaction Template. select Email Address. You can see that this pattern uses Regular expression type of Data Redaction. You can also change any of the parameters. Click on OK.

Add

* Column	EMAIL
* Column Datatype	VARCHAR2
Redaction Template	Email Address
* Redaction Function	REGEX
Regular Expression Based Redaction. Specifies a regular expression that represents the column data that will be redacted.	
Function Attributes	
* Pattern	([A-Z0-9._%+-]+)@([A-Z0-9.-]+\.)
Specifies the regular expression pattern to be searched. Example: '\d\d\d\d\d\d' for number like '012345678'	
* Replace String	xxxx@\2
Example: Use 'XXXXXX\3' (replace string) to redact '012345678' (actual value) which matches '(\d\d\d\d) (\d\d\d\d) (\d\d\d)' (regexp pattern) to 'XXXXXX678' (redacted results). Note that the '\3' in the replace string preserves the actual data in the third set of parentheses in the pattern.	
* Position	1
Specifies the starting position of the string search. The default is 1, meaning it begins the search from the first character of column data.	
* Occurrence	1
Specifies how to perform the search and replace operation. Zero means it replaces all occurrences. Positive integer 'n' would replace nth occurrence of the string.	
Match Parameter	Ignore case
Specifies the matching parameters for the REGEX redaction function.	
OK Cancel	

Defining redaction type for email column

12. On next page you can change Policy Expression. Click OK on the top right corner.

The screenshot shows the Oracle Enterprise Manager interface for Cloud Control 12c. The title bar reads "ORACLE Enterprise Manager Cloud Control 12c". The navigation bar includes links for Enterprise, Targets, Favorites, and History. Below the navigation is a breadcrumb trail: dbm / CUST2. The main menu has links for Oracle Database, Performance, Availability, Security, Schema, and Administration. The current page is titled "Edit Data Redaction Policy: SAL_POLICY". The policy details are as follows:

* Schema	HR
* Table/View	EMPLOYEES
* Policy Name	SAL_POLICY

A red box highlights the "Policy Expression" section, which contains the value "1=1". To the right of this section is a small edit icon. Below this is a table titled "Object Columns" with the following data:

Column	Column Datatype	Redaction Function	Function Attributes
SALARY	NUMBER	FULL	
EMAIL	VARCHAR2	REGEX	([A-Z0-9._%+-]+)@([A-Z0-9.-]+\.[A-Z]{2,4}),xxxx@ 2,1,1,i

You can change the policy expression

13. To disable SAL_POLICY, select it and click on Disable.

ORACLE Enterprise Manager Cloud Control 12c

Enterprise Targets Favorites History

dbm / CUST2

Oracle Database Performance Availability Security Schema Administration

Data Redaction

Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications

Search Data Redaction Policies

Schema %
Table/View %
Policy Name %
Go

Data Redaction Policies

	Create	Edit	View	Enable	Disable	Delete
Schema						
Table/View						
Policy Name						
Enabled						
Redacted Columns						
HR						
EMPLOYEES						
SAL_POLICY						
Enabled						
2						

Disabling the sal_policy redaction policy

14. To enable SAL_POLICY, select it and click on Enable.

ORACLE Enterprise Manager Cloud Control 12c

Enterprise Targets Favorites History

dbm / **CUST2**

Oracle Database Performance Availability Security Schema Administration

Confirmation
Policy SAL_POLICY is disabled.

Data Redaction
Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications

Search Data Redaction Policies

Schema %
Table/View %
Policy Name %
Go

Data Redaction Policies

Schema	Table/View	Policy Name	Enabled	Redacted Columns
HR	EMPLOYEES	SAL_POLICY	Enabled	2

Enabling the sal_policy redaction policy

15. To delete SAL_POLICY, select it and click on Delete.

ORACLE Enterprise Manager Cloud Control 12c

Enterprise Targets Favorites History

dbm / CUST2

Oracle Database Performance Availability Security Schema Administration

Data Redaction

Oracle Data Redaction provides an easy way to quickly redact sensitive information that is displayed in applications

Search Data Redaction Policies

Schema %
Table/View %
Policy Name %

Go

Data Redaction Policies

	Create	Edit	View	Enable	Disable	Delete
Schema						
Table/View						
Policy Name						
Enabled						
Redacted Columns						
HR						
EMPLOYEES						
SAL_POLICY						
Enabled						
Redacted Columns						2

Deleting the sal_policy redaction policy

16. You should see the Confirmation message.

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. The top navigation bar includes links for Enterprise, Targets, Favorites, and History. Below the navigation is a breadcrumb trail: dbm / CUST2. The main menu has options for Oracle Database, Performance, Availability, Security, Schema, and Administration. A yellow confirmation box displays the message: "Policy SAL_POLICY has been deleted successfully." Below this, the "Data Redaction" section is visible, featuring a search form with fields for Schema (%), Table/View (%), and Policy Name (%), and a "Go" button. A table titled "Data Redaction Policies" is shown, with columns for Schema, Table/View, Policy Name, Enabled, and Redacted Columns. The table currently contains no data.

The SAL_POLICY policy has been successfully deleted

Changing the function parameters for a specified column

1. Connect to the database as the secmgr user and alter the policy EMP_POL:

```
$ sqlplus secmgr
SQL> BEGIN
 2 DBMS_REDACT.ALTER_POLICY(
 3 object_schema => 'ernesto',
 4 object_name   => 'tbl',
 5 policy_name   => 'a_tbl_partial',
```

```
6 action      => DBMS_REDACT.MODIFY_COLUMN,
7 column_name  => 'a',
8 function_type => DBMS_REDACT.PARTIAL,
9 function_parameters => '9,1,4');
10 END;
11 /
```

2. Connect as the user usr2 to the database and view data in column A in the ernesto.tbl table:

```
SQL> connect usr2/oracle2
Connected.
```

```
SQL> select a from ernesto.tbl;
A -----
 999956  999967  999978
```

Add a column to the redaction policy

1. Connect to the database as the secmgr user and alter the EMP_POL policy:

```
$ sqlplus secmgr
SQL> BEGIN
 2 DBMS_REDACT.ALTER_POLICY(
 3 object_schema => 'HR',
 4 object_name   => 'EMPLOYEES',
 5 policy_name   => 'EMP_POL',
 6 action        => DBMS_REDACT.ADD_COLUMN,
 7 column_name   => 'COMMISSION_PCT',
 8 function_type => DBMS_REDACT.FULL);
 9 END;
10 /
```

PL/SQL procedure successfully completed.

2. Connect the user hr to the database and execute the following query:

```
select employee_id, salary, commission_pct from hr.employees      where  
commission_pct IS NOT NULL order by employee_id fetch      first 10  
rows only;
```

The screenshot shows a terminal window titled "oracle@dbhost:~/Desktop". The menu bar includes File, Edit, View, Search, Terminal, and Help. The command line starts with "SQL> connect hr" followed by an "Enter password:" prompt. After connecting, the user executes a query:

```
SQL> select employee_id, salary, commission_pct from hr.employees  
2  where commission_pct IS NOT NULL  
3  order by employee_id  
4  fetch first 10 rows only;
```

The output displays a table with three columns: EMPLOYEE_ID, SALARY, and COMMISSION_PCT. The data is as follows:

EMPLOYEE_ID	SALARY	COMMISSION_PCT
145	12359	0
146	11913	0
147	3375	0
148	3489	0
149	1320	0
150	5932	0
151	7726	0
152	5400	0
153	835	0
154	6766	0

Below the table, the message "10 rows selected." is displayed. The SQL prompt "SQL>" is at the bottom left.

Two columns are redacted

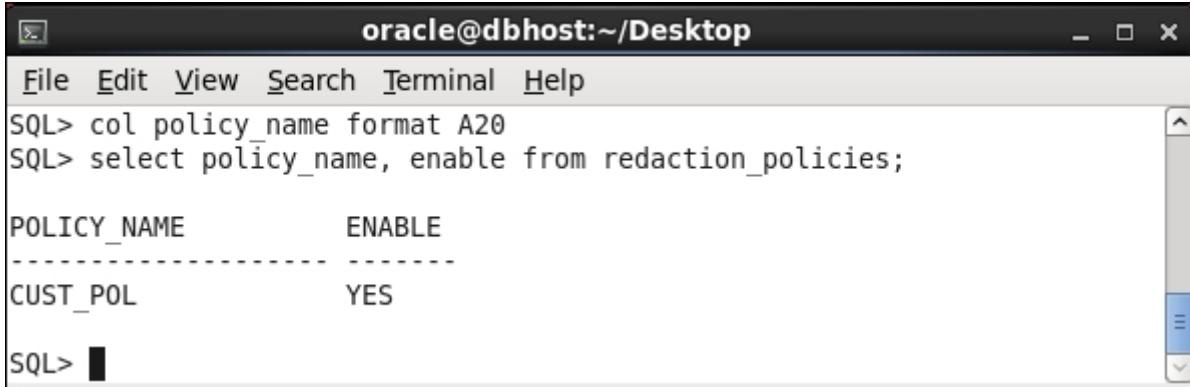
Enabling, disabling, and dropping redaction policy

1. Connect to the database as a user who has an execute privilege on dbms_redact package and select_catalog_role role (for example, secmgr user):

```
$ sqlplus secmgr
```

2. Find out which redaction policies exist in the database by querying the redaction_policies view:

```
SQL> col policy_name format A20
select policy_name, enable from redaction_policies;
```



The screenshot shows a terminal window titled "oracle@dbhost:~/Desktop". The window contains the following SQL*Plus session:

```
File Edit View Search Terminal Help
SQL> col policy_name format A20
SQL> select policy_name, enable from redaction_policies;

POLICY_NAME          ENABLE
-----
CUST_POL             YES

SQL> ■
```

Finding defined redaction policies

3. Connect to the database as the oe user and grant the SELECT privilege on OE.CUSTOMERS to the secmgr user. Connect to the database as the secmgr user. Verify that the secmgr user can't see original data in the column INCOME_LEVEL:

```
SQL> connect oe
```

```
SQL> grant select on oe.customers to secmgr;
```

```
SQL> connect secmgr
```

```
SQL> select customer_id, cust_last_name, income_level from oe.customers 2
order by customer_id 3 fetch first 10 rows only;
```

```

oracle@dbhost:~/Desktop
File Edit View Search Terminal Help
SQL> connect oe
Enter password:
Connected.
SQL> grant select on oe.customers to secmgr;
Grant succeeded.

SQL> connect secmgr
Enter password:
Connected.
SQL> select customer_id, cust_last_name, income_level from oe.customers
  2  order by customer_id
  3  fetch first 10 rows only;

CUSTOMER_ID CUST_LAST_NAME           INCOME_LEVEL
-----  -----
    101 Welles                         T
    102 Pacino                        T
    103 Taylor                         T
    104 Sutherland                     T
    105 MacGraw                        T
    106 Hannah                         T
    107 Cruise                         T
    108 Mason                          T
    109 Cage                           T
    110 Sutherland                     T

10 rows selected.

SQL> ■

```

Redacted data is displayed even to the user who created the policy

4. Disable the redaction policy CUST_POL (as the secmgr user):

```

SQL> begin
  2  dbms_redact.disable_policy
  3  (object_schema => 'OE',
  4  object_name => 'CUSTOMERS',
  5  policy_name => 'CUST_POL');
  6  end;
  7 /

```

PL/SQL procedure successfully completed.

5. Verify that now the secmgr user can view original data in the column INCOME_LEVEL and query the redaction_policies view by executing the following statements:

- `select customer_id, cust_last_name, income_level from oe.customers order by customer_id fetch first 10 rows only;`
- `col policy_name format A20`
- `select policy_name, enable from redaction_policies;`

```
oracle@dbhost:~/Desktop
File Edit View Search Terminal Help

SQL> select customer_id, cust_last_name, income_level from oe.customers
  2 order by customer_id
  3 fetch first 10 rows only;

CUSTOMER_ID CUST_LAST_NAME          INCOME_LEVEL
-----        -----
      101 Welles                  B: 30,000 - 49,999
      102 Pacino                 I: 170,000 - 189,999
      103 Taylor                  H: 150,000 - 169,999
      104 Sutherland              H: 150,000 - 169,999
      105 MacGraw                 C: 50,000 - 69,999
      106 Hannah                  F: 110,000 - 129,999
      107 Cruise                  G: 130,000 - 149,999
      108 Mason                   H: 150,000 - 169,999
      109 Cage                     F: 110,000 - 129,999
      110 Sutherland              G: 130,000 - 149,999

10 rows selected.

SQL> col policy_name format A20
SQL> select policy_name, enable from redaction_policies;

POLICY_NAME          ENABLE
-----                -----
CUST_POL             NO

SQL> █
```

secmgr can view unmasked data in the column income_level, because the cust_pol policy is disabled

- **Enable the redaction policy CUST_POL:**

```
SQL> begin
 2 dbms_redact.enable_policy
 3 (object_schema => 'OE',
 4 object_name => 'CUSTOMERS',
 5 policy_name => 'CUST_POL');
 6 end;
 7 /
```

PL/SQL procedure successfully completed.

6. Verify that redaction is working properly by executing the following statements:

- **select customer_id, cust_last_name, income_level from oe.customers order by customer_id fetch first 10 rows only;**
- **col policy_name format A20**
- **select policy_name, enable from redaction_policies;**

```

oracle@dbhost:~/Desktop
File Edit View Search Terminal Help

SQL> select customer_id, cust_last_name, income_level from oe.customers
  2 order by customer_id
  3 fetch first 10 rows only;

CUSTOMER_ID CUST_LAST_NAME           INCOME_LEVEL
-----  -----
101 Welles                         T
102 Pacino                          T
103 Taylor                          T
104 Sutherland                      T
105 MacGraw                         T
106 Hannah                          T
107 Cruise                          T
108 Mason                           T
109 Cage                            T
110 Sutherland                      T

10 rows selected.

SQL> col policy_name format A20
SQL> select policy_name, enable from redaction_policies;

POLICY_NAME          ENABLE
-----  -----
CUST_POL                         YES

SQL>

```

Redacted data is displayed to the secmgr user because the cust_pol redaction policy is enabled

- **Drop the redaction policy CUST_POL:**

```

SQL> begin
  2 dbms_redact.drop_policy
  3 (object_schema => 'OE',
  4 object_name => 'CUSTOMERS',
  5 policy_name => 'CUST_POL');
  6 end;
  7 /

```

PL/SQL procedure successfully completed.

7. Verify that the redaction policy CUST_POL doesn't exist in the database by executing the following statements:

- `select customer_id, cust_last_name, income_level from oe.customers order by customer_id fetch first 10 rows only;`
- `col policy_name format A20`
- `select policy_name, enable from redaction_policies;`

The screenshot shows the Oracle SQL*Plus interface with the title bar "oracle@dbhost:~/Desktop". The menu bar includes File, Edit, View, Search, Terminal, and Help. The SQL prompt is at the top. Below it, the first query is run:

```
SQL> select customer_id, cust_last_name, income_level from oe.customers
  2 order by customer_id
  3 fetch first 10 rows only;
```

The results are displayed in a grid:

CUSTOMER_ID	CUST_LAST_NAME	INCOME_LEVEL
101	Welles	B: 30,000 - 49,999
102	Pacino	I: 170,000 - 189,999
103	Taylor	H: 150,000 - 169,999
104	Sutherland	H: 150,000 - 169,999
105	MacGraw	C: 50,000 - 69,999
106	Hannah	F: 110,000 - 129,999
107	Cruise	G: 130,000 - 149,999
108	Mason	H: 150,000 - 169,999
109	Cage	F: 110,000 - 129,999
110	Sutherland	G: 130,000 - 149,999

Below the grid, the message "10 rows selected." is shown. The second query is run:

```
SQL> col policy_name format A20
SQL> select policy_name, enable from redaction_policies;
```

The output shows "no rows selected" in a green box, indicating that no policies are found.

```
no rows selected
```

SQL> █

The redaction policy cust_pol doesn't exist anymore

Exempting users from data redaction policies

1. Connect to the database as a user who has a DBA role (for example, user ernesto):

```
$ sqlplus ernesto/oracle
```

2. Create a new user (for example, vipuser) and grant him the create session privilege and select privilege on table customers in schema ernesto:

```
SQL> create user vipuser identified by oracle;
```

```
SQL> grant create session to vipuser;
```

```
SQL> grant select on ernesto.customers to vipuser;
```

3. Connect as a newly created user and try to select from the ernesto.customers table:

```
SQL> connect vipuser/oracle
```

```
SQL> select * from ernesto.customers;
```

NAME	CREDIT_CARD
tom	#####9132 steve
	#####5691 john
	#####5806

4. Connect again as the user ernesto, and grant the EXEMPT REDACTION POLICY privilege to the vipuser user:

```
SQL> connect ernesto/oracle
```

```
SQL> grant exempt redaction policy to vipuser;
```

5. As the user vipuser, now try to select from the table ernesto.customers:

```
SQL> connect vipuser/oracle
```

```
SQL> select * from ernesto.customers;
```

NAME	CREDIT_CARD
tom	3455647456589132
steve	3734982321225691
3472586894975806	john

Transparent Sensitive Data Protection

Creating a sensitive type

- 1. Connect to the database (for example, pdb1) as a user who has appropriate privileges (for example, c##ernesto):*

```
$ sqlplus c##ernesto@pdb1
```

- 2. Create a sensitive type (for example, email_type):*

```
SQL> BEGIN
  2  DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (
  3    sensitive_type => '<your_type>',
  4    user_comment=> '<description>');
  5  END;
  6 /
```

```
SQL> BEGIN
  2  DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (
  3    sensitive_type => 'email_type',
  4    user_comment=>'Type for email redaction');
  5  END;
  6 /
PL/SQL procedure successfully completed.

SQL> █
```

Creating a sensitive type

Determining sensitive columns

1. Connect to the database (for example, `pdb1`) as a user who has appropriate privileges (for example, `c##ernesto` user):

```
$ sqlplus c##ernesto@pdb1
```

2. Associate a sensitive column (for example, schema `CHALLENGEERNESTO`, table `T1`, column `EMAIL_ADDRESS`) with sensitive type you created in the previous recipe (for example, `email_type`)

```
SQL> BEGIN
 2  DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN (
 3  schema_name => 'CHALLENGEERNESTO',
 4  table_name => 'T1',
 5  column_name => 'EMAIL_ADDRESS',
 6  sensitive_type => 'email_type');
 7  END;
 8 /
```

PL/SQL procedure successfully completed.

Adding sensitive column email_address to email_type sensitive type

3. Associate another sensitive column (for example, schema `HR`, table `EMPLOYEES`, column `EMAIL`) with the same sensitive data type (for example, `email_type`).

```
SQL> BEGIN
 2  DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN (
 3  schema_name => 'HR',
 4  table_name => 'EMPLOYEES',
 5  column_name => 'EMAIL',
 6  sensitive_type => 'email_type');
 7  END;
 8 /
```

PL/SQL procedure successfully completed.

SQL> █

Adding sensitive column email to sensitive type email_type

Creating transparent sensitive data protection policy

1. Connect to the database (for example, pdb1) as a user who has appropriate privileges (for example, c##ernesto user):

```
$ sqlplus c##ernesto@pdb1
```

2. Create TSDP policy using Data Redaction.

```
SQL> DECLARE
  2  redact_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
  3  policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
  4  BEGIN
  5    redact_feature_options('expression') :='1=1';
  6    redact_feature_options('function_type') :='DBMS_REDACT.REGEXP';
  7    redact_feature_options('regexp_pattern'):= '([A-Za-z0-9_.%+-]+)@[([A-Za-z0-9_-]+\.[A-Za-z]{2,4})]';
  8    redact_feature_options('regexp_replace_string'):= '\1@xxxx.com';
  9    policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'VARCHAR2';
10   DBMS_TSDP_PROTECT.ADD_POLICY ('redact_regexp_email',DBMS_TSDP_PROTECT.REDACT,
11   redact_feature_options, policy_conditions);
12 END;
13 /
```

PL/SQL procedure successfully completed.

```
SQL> █
```

TSDP policy using Oracle Data Redaction

Associating transparent sensitive data protection policy with sensitive type

1. Connect to the database as a user (for example, pdb1) who has appropriate privileges (for example, c##ernesto user):

```
$ sqlplus c##ernesto@pdb1
```

2. Associate TSDP policy with sensitive type:

```
SQL> BEGIN
  2  DBMS_TSDP_PROTECT.ASSOCIATE_POLICY (
  3  policy_name => 'redact_regexp_email',
  4  sensitive_type => 'email_type',
  5  associate => true);
  6 END;
  7 /
```

PL/SQL procedure successfully completed.

```
SQL> █
```

Enabling, disabling, and dropping policy

- 1. Connect to the database (for example, pdb1) as a user who has the SELECT privilege on the HR.EMPLOYEES table and the CHALLENGEERNESTO.T1 table or the SELECT ANY TABLE privilege (for example, maja).*

```
$ sqlplus maja@pdb1
```

- 2. View sensitive data by executing the following two queries:*

SELECT EMAIL FROM HR.EMPLOYEES FETCH FIRST 10 ROWS ONLY;

```
SQL> SELECT EMAIL FROM HR.EMPLOYEES FETCH FIRST 10 ROWS ONLY;
EMAIL
-----
ABANDA
ABULL
ACABRIO
AERRAZUR
AFRIPP
AHUNOLD
AHUTTON
AKHOO
AMCEWEN
AMALSH
10 rows selected.
SQL> █
```

Before enabling the policy

SELECT EMAIL_ADDRESS FROM CHALLENGEERNESTO.T1;

```
SQL> connect maja/oracle@pdb1
Connected.
SQL> select email_address from challengeernesto.t1;
EMAIL_ADDRESS
-----
ERNESTO.PAVLOVIC@CHALLENGEERNESTO.COM
MAJA.VESELICA@CHALLENGEERNESTO.COM
SQL> █
```

Before enabling the policy

3. Connect to the database (for example, pdb1) as a user who can manage TSDP policies (for example, c##ernesto). Enable the TSDP policy:

```
SQL> connect c##ERNESTO/oracle@pdb1
Connected.
SQL> BEGIN
 2  DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE (
 3  sensitive_type => 'email_type');
 4 END;
 5 /
PL/SQL procedure successfully completed.
```

```
SQL> █
```

4. Repeat step 2 as user maja.

```
SQL> select email from hr.employees fetch first 10 rows only;
EMAIL
-----
10 rows selected.
```

Sensitive data is protected

5.Result of the second query is shown:

```
SQL> connect maja/oracle@pdb1
Connected.
SQL> select email_address from challenge.ernesto.t1;
EMAIL_ADDRESS
-----
ERNESTO.PAVLOVIC@xxxx.com
MAJA.VESELICA@xxxx.com
SQL> █
```

After enabling the policy

6. Connect to the database (for example, pdb1) as a user who can manage TSDP policies (for example, c##ernesto). Disable the TSDP policy.

```
SQL> BEGIN
 2  DBMS_TSDP_PROTECT.DISABLE_PROTECTION_TYPE (
 3  sensitive_type => 'email_type');
 4 END;
 5 /
PL/SQL procedure successfully completed.

SQL> █
```

7. Repeat step 2 as user maja.

```
SQL> SELECT EMAIL FROM HR.EMPLOYEES FETCH FIRST 10 ROWS ONLY;
EMAIL
-----
ABANDA
ABULL
ACABRIO
AERRAZUR
AFRIPP
AHUNOLD
AHUTTON
AKHOO
AMCEWEN
AWALSH
10 rows selected.

SQL> █
```

After the policy was disabled

8. The result of the second query is shown:

```
SQL> connect maja/oracle@pdb1
Connected.
SQL> select email_address from challenge@ernesto.t1;

EMAIL_ADDRESS
-----
ERNESTO.PAVLOVIC@CHALLENGEZORAN.COM
MAJA.VESELICA@CHALLENGEZORAN.COM

SQL> █
```

After the policy was disabled

9. Connect to the database (for example, `pdb1`) as a user who can manage TSDP policies (for example, `c##ernesto`). Drop both sensitive columns.

```
SQL> BEGIN
 2  DBMS_TSDP_MANAGE.DROP_SENSITIVE_COLUMN (
 3  schema_name => 'CHALLENGE.ERNESTO',
 4  table_name => 'T1',
 5  column_name => 'EMAIL_ADDRESS');
 6 END;
 7 /
PL/SQL procedure successfully completed.

SQL> BEGIN
 2  DBMS_TSDP_MANAGE.DROP_SENSITIVE_COLUMN (
 3  schema_name => 'HR',
 4  table_name => 'EMPLOYEES',
 5  column_name => 'EMAIL');
 6 END;
 7 /
PL/SQL procedure successfully completed.
```

10. Drop the sensitive type.

```
SQL> BEGIN
 2  DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE (
 3  sensitive_type => 'email_type');
 4 END;
 5 /
PL/SQL procedure successfully completed.
```

11. Drop the TSDP policy.

```
SQL> BEGIN
 2  DBMS_TSDP_PROTECT.DROP_POLICY (
 3  policy_name => 'redact-regexp_email');
 4 END;
 5 /
PL/SQL procedure successfully completed.

SQL> █
```

Altering transparent sensitive data protection policy

1. Connect to the database (for example, pdb1) as a user who can manage TSDP policies (for example, c##ernesto):

```
$ sqlplus c##ernesto@pdb1
```

2. If the policy is enabled, disable it for all columns (for instructions how to disable the TSDP policy, see recipe Enabling, disabling, and dropping policy).

3. Connect to the database (for example, pdb1) as a user who can view sensitive data (for example, maja). Execute the following queries:

```
SELECT EMAIL FROM HR.EMPLOYEES FETCH FIRST 10 ROWS ONLY;
```

```
SQL> SELECT EMAIL FROM HR.EMPLOYEES FETCH FIRST 10 ROWS ONLY;
EMAIL
-----
ABANDA@example.com
ABULL@example.com
ACABRIO@example.com
AERRAZUR@example.com
AFRIPPP@example.com
AHUNOLD@example.com
AHUTTON@example.com
AKHOO@example.com
AMCEWEN@example.com
AWALSH@example.com
10 rows selected.

SQL> █
```

Before altering and enabling the policy

```
SELECT EMAIL_ADDRESS FROM CHALLENGEERNESTO.T1;
```

```

SQL> SELECT EMAIL_ADDRESS FROM CHALLENGE_ERNESTO.T1;
EMAIL_ADDRESS
-----
ERNESTO.PAVLOVIC@CHALLENGE_ERNESTO.COM
MAJA.VESELICA@CHALLENGEZORAN.COM

SQL> █

```

Before altering and enabling the policy

4. Connect to the database (for example, `pdb1`) as a user who can manage TSDP policies (for example, `c##ernesto`). Alter the TSDP policy and enable it.

```

SQL> connect c##ernesto/oracle@pdb1
Connected.
SQL> DECLARE
  2   redact_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
  3   policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
  4   BEGIN
  5     redact_feature_options ('expression') :='1=1';
  6     redact_feature_options ('function_type') :='DBMS_REDACT.REGEXP';
  7     redact_feature_options ('regexp_pattern'):=('[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}');
  8     redact_feature_options ('regexp_replace_string'):='\1@mydomain.com';
  9     redact_feature_options ('regexp_position'):=1;
 10    redact_feature_options ('regexp_occurrence'):=DBMS_REDACT.RE_FIRST';
 11    policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'VARCHAR2';
 12    DBMS_TSDP_PROTECT.ALTER_POLICY ('redact_regex_email',redact_feature_options, policy_conditions);
 13  END;
 14 /
PL/SQL procedure successfully completed.

SQL> BEGIN
  2   DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE (
  3   sensitive_type => 'email_type');
  4   END;
  5 /
PL/SQL procedure successfully completed.

SQL> █

```

Alter the TSDP policy

5. View sensitive data as the user `maja` (repeat step 3).

```

SQL> connect maja/oracle@pdb1
Connected.
SQL> SELECT EMAIL FROM HR.EMPLOYEES FETCH FIRST 10 ROWS ONLY;

EMAIL
-----
ABANDA@mydomain.com
ABULL@mydomain.com
ACABRIO@mydomain.com
AERRAZUR@mydomain.com
AFRIPPP@mydomain.com
AHUNOLD@mydomain.com
AHUTTON@mydomain.com
AKHOO@mydomain.com
AMCEWEN@mydomain.com
AWALSH@mydomain.com

10 rows selected.

SQL> SELECT EMAIL_ADDRESS FROM CHALLENGE.ERNESTO.T1;

EMAIL_ADDRESS
-----
ERNESTO.PAVLOVIC@mydomain.com
MAJA.VESELICA@mydomain.com

SQL> █

```

After altering TSDP policy

Privilege Analysis

Creating database analysis policy

1. Connect to the database as system or a user who has appropriate privilege:

\$ sqlplus system

2. Create a privilege analysis policy that captures all the used privileges in the database:

```

SQL> BEGIN
  SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE( name =>
    '<policy_name>', description => '<your_desc>', type =>
    DBMS_PRIVILEGE_CAPTURE.G_DATABASE); END; /

```

```
SQL> BEGIN
 2  SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
 3  name => 'ALL_PRIV_POL',
 4  description => 'All privileges',
 5  type => DBMS_PRIVILEGE_CAPTURE.G_DATABASE);
 6 END;
 7 /
```

```
PL/SQL procedure successfully completed.
```

Database (unconditional) analysis policy

Creating role analysis policy

- 1. Connect to the database as system or a user who has appropriate privileges:*

\$ sqlplus system

- 2. Create a privilege analysis policy that captures all the used privileges granted through roles DBA and P1_ROLE:*

SQL> BEGIN

```
SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(    name
=> '<policy_name>',    description => '<your_desc>',    type =>
DBMS_PRIVILEGE_CAPTURE.G_ROLE,    roles => role_name_list
(<'role1',...,'role10'>);    END;    /
```

```
SQL> BEGIN
 2  SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
 3  name => 'ROLE_PRIV_POL',
 4  description => 'Usage of privileges granted through listed roles',
 5  type => DBMS_PRIVILEGE_CAPTURE.G_ROLE,
 6  roles => role_name_list ('DBA','P1_ROLE'));
 7 END;
 8 /
```

```
PL/SQL procedure successfully completed.
```

The role analysis policy

Creating context analysis policy

- 1. Connect to the database as system or a user who has appropriate privileges:*

```
$ sqlplus system
```

- 2. Create a privilege analysis policy that captures all the used (and unused) privileges by Steve:*

```
SQL> BEGIN
  SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE( name =>
  '<policy_name>', description => '<your_desc>', type =>
  DBMS_PRIVILEGE_CAPTURE.G_CONTEXT, condition =>
  '<your_condition>');
END; /
```

```
SQL> BEGIN
  2  SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  3  name => 'CONT_PRIV_POL',
  4  description => 'Privileges used by Steve',
  5  type => DBMS_PRIVILEGE_CAPTURE.G_CONTEXT,
  6  condition => 'SYS_CONTEXT(''USERENV'', ''SESSION_USER'')='STEVE''');
  7 END;
  8 /
```

PL/SQL procedure successfully completed.

The context analysis policy

Creating combined analysis policy

- 1. Connect to the database as system or a user who has appropriate privileges:*

```
$ sqlplus system
```

2. Create a privilege analysis policy that captures the usage of privileges, when using SQL Developer, which are granted through the role P2_ROLE:

```
SQL> BEGIN
  SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(      name
=> '<policy_name>',      description => '<your_desc>',      type =>
DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT,      roles
=> role_name_list (<'role1',...,'role10'>),      condition =>
'<your_condition>');
      END;      /
```

```
SQL> BEGIN
  2  SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  3  name => 'COM_PRIV_POL',
  4  description => 'Usage of privileges when using SQL Developer that are granted
d through role P2_ROLE ',
  5  type => DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT,
  6  roles => role_name_list ('P2_ROLE'),
  7  condition => 'SYS_CONTEXT(''USERENV'', ''CLIENT_PROGRAM_NAME'')='''SQL Develop
er'''');
  8  END;
  9  /
```

PL/SQL procedure successfully completed.

The combined analysis policy

Starting and stopping privilege analysis

1. Connect to the database as system or a user who has appropriate privileges:

\$ sqlplus system

2. List all existing privilege analysis policies by querying DBA_PRIV_CAPTURES.

```

SQL> column name format A20
SQL> select name, type, enabled
  2  from DBA_PRIV_CAPTURES;

NAME          TYPE      E
-----
ROLE_PRIV_POL    ROLE      N
ALL_PRIV_POL     DATABASE  N
CONT_PRIV_POL    CONTEXT   N
COM_PRIV_POL     ROLE_AND_CONTEXT N
ORA$DEPENDENCY   DATABASE  N

```

Finding all defined policies

3. Enable a privilege analysis (for example, ALL_PRIV_POL, which you created in the first recipe in this chapter):

```

SQL> BEGIN
  2  SYS.DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE( name =>
  3    '<policy_name>');
  4  END; /

```

```

SQL> BEGIN
  2  SYS.DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE(
  3    name => 'ALL_PRIV_POL');
  4  END;
  5 /

```

PL/SQL procedure successfully completed.

Start capturing all privileges

4. Connect to the database as the user alan and view the first names of employees who have salary less than 1000:

```

SQL> connect alan
Enter password:
Connected.
SQL> select first_name from HR.EMPLOYEES
  2  WHERE SALARY < 1000;
no rows selected

```

the first test of select privilege

5. Find first names of employees who earn less than 3 000.

```
SQL> select first_name from HR.EMPLOYEES  
2 WHERE SALARY < 3000;
```

The second test of select privilege

6. Try to delete all employees whose first name is Karen.

```
SQL> DELETE FROM HR.EMPLOYEES  
2 WHERE FIRST_NAME = 'Karen';  
DELETE FROM HR.EMPLOYEES  
*  
ERROR at line 1:  
ORA-02292: integrity constraint (HR.EMP_MANAGER_FK) violated - child record  
found
```

The test of delete privilege: integrity constraint violation

7. Connect to the database as system or a user who has appropriate privileges. Stop collecting data about privileges:

```
SQL> connect system  
SQL> BEGIN  
 2   SYS.DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE(  
 3     name => '<policy_name>');  
 4   END; /
```

```
SQL> BEGIN  
2   SYS.DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE(  
3     name => 'ALL_PRIV_POL');  
4   END;  
5   /
```

```
PL/SQL procedure successfully completed.
```

Stop capturing

8. Generate the result:

```
SQL> BEGIN  
  SYS.DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT(  
=> '<policy_name>'); END;      /
```

```
SQL> BEGIN  
 2  SYS.DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT(  
 3  name => 'ALL_PRIV_POL');  
 4  END;  
 5  /  
  
PL/SQL procedure successfully completed.
```

Generating the report

Reporting on used system privileges

1. Connect to the database as system or a user who has appropriate privileges:

```
$ sqlplus system
```

2. View system privileges that the user ALAN used:

```
SQL> select username, sys_priv  
 2  from DBA_USED_SYSPRIVS  
 3  where username='ALAN';  
  
USERNAME    SYS_PRIV  
-----  
ALAN        CREATE SESSION
```

The used system privileges

3. View grant path for the used system privileges generated by ALL_PRIV_POL for the user ALAN:

```

SQL> column path format A20
SQL> select sys_priv, path
  2  from DBA_USED_SYSPRIVS_PATH
  3  where capture='ALL_PRIV_POL' and username='ALAN';

SYS_PRIV          PATH
-----
CREATE SESSION      GRANT_PATH('ALAN')

```

The Grant path

Reporting on used object privileges

1. Connect to the database as system or a user who has appropriate privileges:

\$ sqlplus system

2. View which object privileges the user Alan has used while database policy ALL_PRIV_POL has been active.

```

SQL> select username, object_owner, object_name, obj_priv
  2  from DBA_USED_OBJPRIVS
  3  where username='ALAN';

USERNAME  OBJECT_OWNER OBJECT_NAME          OBJ_PRIV
-----
ALAN      SYS          DUAL                SELECT
ALAN      SYS          DUAL                SELECT
ALAN      SYS          DBMS_APPLICATION_INF EXECUTE
0

ALAN      HR           EMPLOYEES           DELETE
ALAN      HR           EMPLOYEES           SELECT
ALAN      SYSTEM        PRODUCT_PRIVS     SELECT

6 rows selected.

```

The used object privileges

3. View grant path by querying DBA_USED_OBJPRIVS_PATH:

```
SQL> select username, object_owner, object_name, obj_priv, path
  2  from DBA_USED_OBJPRIVS_PATH
  3  where capture='ALL_PRIV_POL' and username='ALAN';

USERNAME OBJECT_OWNER OBJECT_NAME          OBJ_PRIV      PATH
-----  -----  -----  -----  -----
ALAN     HR        EMPLOYEES            DELETE        GRANT_PATH('ALAN')
ALAN     SYS       DUAL                SELECT         GRANT_PATH('PUBLIC')
ALAN     SYS       DUAL                SELECT         GRANT_PATH('PUBLIC')
ALAN     SYS       DBMS_APPLICATION_INF EXECUTE      GRANT_PATH('PUBLIC')
                           0
ALAN     SYSTEM    PRODUCT_PRIVS        SELECT         GRANT_PATH('PUBLIC')
ALAN     HR        EMPLOYEES            SELECT         GRANT_PATH('ALAN')

6 rows selected.
```

Object privileges grant path

Reporting on unused system privileges

1. Connect to the database as system or a user who has appropriate privileges:

\$ sqlplus system

2. View that the user Alan has used all system privileges that have been granted to him (there are no unused system privileges):

```
SQL> select username, sys_priv
  2  from DBA_UNUSED_SYSPRIVS
  3  where username='ALAN';

no rows selected
```

The unused system privileges for the user Alan during the database policy ALL_PRIV_POL capture interval

Reporting on unused object privilege

- 1. Connect to the database as system or a user who has appropriate privileges:*

\$ sqlplus system

- 2. View which object privileges the user Alan has used during the database policy capture interval:*

```
SQL> select username, object_owner, object_name, obj_priv
  2  from DBA_UNUSED_OBJPRIVS
  3  where username='ALAN';

USERNAME    OBJECT_OWNER OBJECT_NAME          OBJ_PRIV
-----      -----      -----
ALAN        HR           EMPLOYEES            UPDATE
ALAN        HR           EMPLOYEES            INSERT
```

The unused object privileges

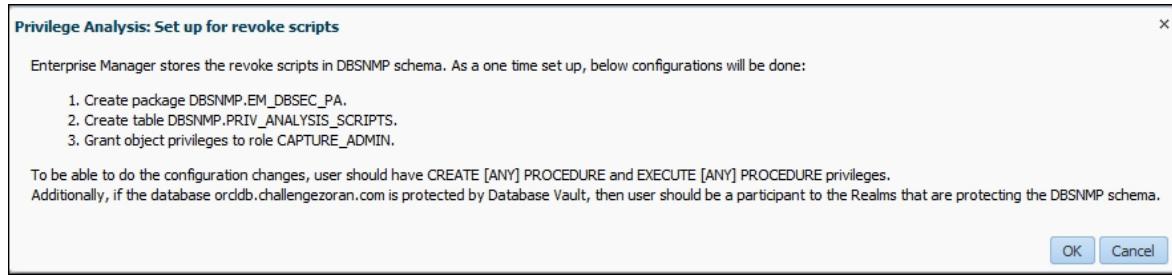
How to revoke unused privileges

- 1. Select policy, and from Actions drop-down menu, choose Revoke Scripts:*

	Active	Type	First Start Time	Last End Time
		Role	Aug 04, 2015 11:13 PM	
		Role and Context		
		Context		
		Database		
		Database	Aug 04, 2015 11:00 PM	Aug 05, 2015 06:30 AM

Create revoke scripts

2. You'll see a message about required privileges. Click on the OK button.



The info message

3. Select policy (Policy Name) and click on the Generate button:

Privilege Analysis: Revoke Scripts

This page lists down privilege revoke and regrant scripts generated and saved based on Privilege Analysis results.

Search

Policy Name ALL_PRIV_POL

Revoke Scripts

Policy Name	Script Name	Revoker Script	Regrant Script	Script Created By	Script Generated Time	Description
No data found						

Generating a script

4. Generate script to revoke all the unused object privileges from the user Alan. Fill out form as shown in the figure below and click on the Next button:

Script Details Select Grantee Unused System Privileges Unused Object Privileges Unused Roles Review

Generate Revoke Script: Script Details

Create a SQL script to revoke directly granted unused system/object privilege, and roles.

* policy Name ALL_PRIV_POL

* Script Name ALAN_OBJ_PRIV_REV

Description
Revoke all unused object privileges from Alan.

Grantee (user/role) All Customize

Unused System Privileges All None Customize

Unused Object Privileges All None Customize

Unused Roles All None Customize

Back Step 1 of 6 Next Done Cancel

Revoking the script configuration

5. Click on the **Select None** link and tick revoke checkbox for the user Alan:

Script Details Select Grantee Unused System Privileges Unused Object Privileges Unused Roles Review

Generate Revoke Script: Select Grantee

Following grantees will be included in the revoke script.

Select Users

Select All Select None

Revoke	Name
<input type="checkbox"/>	DVF
<input type="checkbox"/>	APEX_040200
<input type="checkbox"/>	MDSYS
<input type="checkbox"/>	OLAPSYS
<input type="checkbox"/>	CTXSYS
<input type="checkbox"/>	ORACLE_OCM
<input type="checkbox"/>	DVSYS
<input checked="" type="checkbox"/>	ALAN
<input type="checkbox"/>	SYSTEM
<input type="checkbox"/>	ORDPLUGINS
<input type="checkbox"/>	PM
<input type="checkbox"/>	ORDSYS
<input type="checkbox"/>	DBSNMP
<input type="checkbox"/>	GSMADMIN_INTERNAL
<input type="checkbox"/>	OE
<input type="checkbox"/>	IX
<input type="checkbox"/>	ORDDATA
<input type="checkbox"/>	XDB
<input type="checkbox"/>	LBACSYS
<input type="checkbox"/>	WMSYS

Rows Selected 1

Back Step 2 of 6 Next Cancel

Choose to revoke privilege only from the user Alan

Click on the **Next** button. Review your choices and click on the **Save** button:

Script Details Select Grantee Unused System Privileges Unused Object Privileges Unused Roles Review

Generate Revoke Script: Review

Back Step 6 of 6 Next Save Cancel

Policy Name	ALL_PRIV_POL
Script Name	ALAN_OBJ_PRIV_REV
Description	Revoke all unused object privileges from Alan.

Directly Granted Unused Object Privileges

Policy Name	Grantee Name	Grantee Type	Object Privileges	Object Owner	Object Name	Column Name	Object Type	Grant Option
ALL_PRIV_POL	ALAN	USER	INSERT	HR	EMPLOYEES		TABLE	
ALL_PRIV_POL	ALAN	USER	UPDATE	HR	EMPLOYEES		TABLE	

Review

You should receive confirmation similar to the one shown:

Confirmation
Script ALAN_OBJ_PRIV_REV has been generated successfully.

Privilege Analysis: Revoke Scripts

This page lists down privilege revoke and regrant scripts generated and saved based on Privilege Analysis results.

Search

Policy Name %

Revoke Scripts

Policy Name	Script Name	Revoke Script	Regrant Script	Script Created By	Script Generated Time	Description
ALL_PRIV_POL	ALAN_OBJ_PRIV_REV			SYSTEM	Aug 05, 2015 07:35 AM	Revoke all unused object privileges from Alan.

Download

The confirmation message

6. Click on the green arrow in the Revoke Script column to download the generated revoke script. Note that Regrant Script has also been generated.

7. View the generated revoke script- ALAN_OBJ_PRIV_REV_revokeScript.sql:

```
-- REVOKE SCRIPT GENERATED BY SYSTEM AT Aug 05, 2015 07:35 AM.  
-- SCRIPT FOR REVOKING DIRECTLY GRANTED UNUSED OBJECT  
PRIVILEGES :  
  
REVOKE INSERT ON HR.EMPLOYEES FROM ALAN;  
REVOKE UPDATE ON HR.EMPLOYEES FROM ALAN;
```

The generated revoke script

Dropping the analysis

- 1. Connect to the database as system or a user who has appropriate privileges:*

\$ sqlplus system

- 2. Drop a privilege analysis policy (for example, ALL_PRIV_POL,*

SQL> BEGIN
SYS.DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE(name =>
'<policy_name>'); END; /

```
SQL> BEGIN  
2  SYS.DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE(  
3  name => 'ALL_PRIV_POL');  
4  END;  
5  /
```

PL/SQL procedure successfully completed.

Drop policy

- 3. Verify that all the records about the used and unused privileges, which have been gathered according to the policy, are also dropped:*

```
SQL> SELECT username, sys_priv, obj_priv, object_owner,  
object_name  FROM DBA_USED_PRIVS  WHERE  
capture='<policy_name>';
```

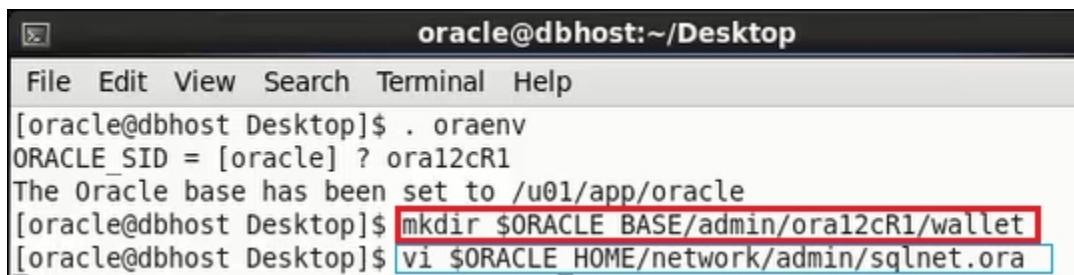
```
SQL> select username, sys_priv, obj_priv, object_owner, object_name  
  2  from DBA_USED_PRIVS  
  3  where capture='ALL_PRIV_POL';  
  
no rows selected
```

Records doesn't exist anymore

Transparent Data Encryption

Configuring keystore location in sqlnet.ora

1. *Create a directory, to hold a keystore, that is accessible to the owner of Oracle software (for example, \$ORACLE_BASE/admin/ora12cR1/wallet).*



The screenshot shows a terminal window titled "oracle@dbhost:~/Desktop". The window contains the following text:

```
[oracle@dbhost Desktop]$ . oraenv  
ORACLE_SID = [oracle] ? ora12cR1  
The Oracle base has been set to /u01/app/oracle  
[oracle@dbhost Desktop]$ mkdir $ORACLE_BASE/admin/ora12cR1/wallet  
[oracle@dbhost Desktop]$ vi $ORACLE_HOME/network/admin/sqlnet.ora
```

The command "mkdir \$ORACLE_BASE/admin/ora12cR1/wallet" is highlighted with a red rectangle, and the command "vi \$ORACLE_HOME/network/admin/sqlnet.ora" is highlighted with a blue rectangle.

Create a directory and edit sqlnet.ora

2. *Edit sqlnet.ora and add entry to specify the location of the keystore. This step is optional if you are using default location for the wallet, which is \$ORACLE_HOME/admin/<db_name>/wallet.*

Creating and opening the keystore

1. *Connect to the database as a user who can grant administer key management privilege (for example, SYS) and grant the privilege to an existing user (for example, maja).*

2. *To create a password-based software keystore, connect to the database as the user in the previous step (for example, `maja`) and execute the following statement (after you change parameters so that they are appropriate for your environment):*

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE  
'keystore_location' IDENTIFIED BY keystore_password;
```

```
SQL> grant administer key management to maja;  
Grant succeeded.  
  
SQL> connect maja  
Enter password:  
Connected.  
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/u01/app/oracle/admin/ora12cR1/w  
allet' identified by welcome1;  
  
keystore altered.
```

Creating a password-based software keystore

3. *Open the keystore you created in the previous step by executing the following statement:*

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN  
IDENTIFIED BY keystore_password;
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY welcome1;  
keystore altered.
```

Opening the password-based keystore

Setting master encryption key in software keystore

1. *Connect to the database as a user who has the `SYSKM` administrative or administer key management privilege (for example, `maja`):*

```
$ sqlplus maja
```

2. *Create a master key for the password-based keystore:*

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED  
BY      keystore_password  WITH BACKUP  USING 'desc_purpose';
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY welcome1 WITH BACKUP USING  
'transparent';  
  
keystore altered.
```

Column encryption - adding new encrypted column to table

1. *Connect to the database as a user who has administer key privilege or SYSKM privilege (for example, maja) and verify that the keystore is in the OPEN status. You should get the result similar to the next figure:*

```
$ sqlplus maja
```

```
SQL> SELECT WRL_PARAMETER, STATUS, WALLET_TYPE FROM V$ENCRYPTION_WALLET;  
  
WRL_PARAMETER                      STATUS        WALLET_TYPE  
-----  
/u01/app/oracle/admin/ora12cR1/wallet/          OPEN          PASSWORD
```

2. *Add a column (for example, bonus) to a table (for example, hr.employees), encrypted using the AES 256 algorithm.*

```
SQL> ALTER TABLE HR.EMPLOYEES ADD (BONUS NUMBER(10) ENCRYPT USING 'AES256');  
  
Table altered.
```

Adding the new encrypted column to the table

Column encryption - creating new table that has encrypted column(s)

1. *Connect to the database as a user who has administer key privilege or SYSKM privilege (for example, maja):*

```
$ sqlplus maja
```

2. *Create a new table (for example, table enc_cols in schema hr) that has, for example, the following structure:*

Column name	Column type	Encrypted
NAME	VARCHAR2 (50)	No
CREDIT_LIMIT	NUMBER (10)	Yes, AES192
SALARY	NUMBER (10)	Yes, AES192

```
SQL> CREATE TABLE HR.ENC_COLS (
 2  NAME VARCHAR2(50),
 3  CREDIT_LIMIT NUMBER(10) ENCRYPT,
 4  SALARY NUMBER(10) ENCRYPT);
```

Table created.

A syntax to create the table hr.enc_cols

3. *Connect to the database as a user who can insert and view data in the table (for example, hr user):*

SQL> connect hr

4. *Insert several arbitrary values into the table HR.ENC_COLS.*

```
SQL> INSERT INTO HR.ENC_COLS VALUES ('Debra',50000,20000);
1 row created.

SQL> INSERT INTO HR.ENC_COLS VALUES ('Sarah',48000,18500);
1 row created.

SQL> INSERT INTO HR.ENC_COLS VALUES ('Tim',45000,14800);
1 row created.

SQL> INSERT INTO HR.ENC_COLS VALUES ('Alex',49000,23000);
1 row created.
```

Test values

5. Verify that the user can view unencrypted values in all columns.

```
SQL> SET LINESIZE 300
SQL> COLUMN NAME FORMAT A10
SQL> select * from hr.enc_cols;

NAME      CREDIT_LIMIT      SALARY
-----  -----
Debra          50000        20000
Sarah          48000        18500
Tim            45000        14800
Alex           49000        23000
```

Encryption is transparent

6. Connect to the database as a user who can't view data in the table (for example, james) and try to view data in all columns:

```
SQL> connect james
SQL> select * from hr.enc_cols;
```

```
SQL> connect james
Enter password:
Connected.
SQL> select * from hr.enc_cols;
select * from hr.enc_cols
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

User who doesn't have "view" privilege(s) won't see encrypted values

Using salt and MAC

1. Connect to the database as a user who has administer key privilege or SYSKM privilege (for example, maja):

```
$ connect maja
```

2. *Encrypt two columns in an existing table (for example, sh.customers)*

```
SQL> ALTER TABLE SH.CUSTOMERS MODIFY (
  2 CUST_LAST_NAME ENCRYPT USING 'AES256',
  3 CUST_STREET_ADDRESS ENCRYPT USING 'AES256' NO SALT);

Table altered.
```

Using salt and MAC

Column encryption - encrypting existing column

1. *Connect to the database as a user who can read data from the OE.CUSTOMERS table (for example, the oe user):*

\$ sqlplus oe

2. *Select data from column you want to encrypt (for example, cust_email), just to verify that the user can view it.*

```
SQL> SELECT CUST_EMAIL FROM OE.CUSTOMERS
  2 WHERE CUST_EMAIL LIKE 'Am%';

CUST_EMAIL
-----
Amanda.Brown@THRASHER.EXAMPLE.COM
Amanda.Finney@STILT.EXAMPLE.COM
Amanda.Tanner@TEAL.EXAMPLE.COM
Amrishi.Palin@EIDER.EXAMPLE.COM
```

A test query

3. *Connect to the database as a user who has administer key privilege or SYSKM privilege (for example, maja):*

SQL> connect maja

4. *Encrypt the cust_email column in the oe.customers table using the default encryption algorithm (AES192) and no salt.*

```
SQL> ALTER TABLE OE.CUSTOMERS MODIFY (CUST_EMAIL ENCRYPT NO SALT);

Table altered.
```

Encrypting an existing column, which has an index

5. *Execute steps 1 and 2 again to verify that there is no change in the way user/application views data after TDE column encryption is applied.*

Auto-login keystore

1. *Connect to the database as a user who has administer key privilege or SYSKM privilege (for example, maja):*

```
$ sqlplus maja
```

2. *Create (local) an autologin keystore. In our case, keystore_location is /u01/app/oracle/admin/ora12cR1/wallet and keystore_password is welcome1:*

```
SQL> ADMINISTER KEY MANAGEMENT CREATE LOCAL  
AUTO_LOGIN KEYSTORE FROM KEYSTORE 'keystore_location'  
IDENTIFIED BY keystore_password; OR SQL> ADMINISTER KEY  
MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM  
KEYSTORE 'keystore_location' IDENTIFIED BY keystore_password;
```

Encrypting tablespace

1. *Connect to the database as a user who has a create tablespace privilege (for example, ernesto):*

```
$ sqlplus ernesto
```

2. *Create encrypted tablespace (for example, TEST_ENC) using AES192 encryption algorithm:*

```
SQL> CREATE TABLESPACE TEST_ENC DATAFILE  
'/u01/app/oracle/oradata/ORA12CR1/datafile/testenc01.dbf' SIZE 20M  
ENCRYPTION USING 'AES192' DEFAULT STORAGE (ENCRYPT);
```

```
SQL> CREATE TABLESPACE TEST_ENC  
  2  DATAFILE '/u01/app/oracle/oradata/ORA12CR1/datafile/testenc01.dbf'  SIZE 20M  
  3  ENCRYPTION USING 'AES192'  
  4  DEFAULT STORAGE (ENCRYPT);
```

```
Tablespace created.
```

Encrypting tablespace

Rekeying

1. *Connect to the database as a user who has administer key privilege or SYSKM privilege (for example, maja):*

```
$ sqlplus maja
```

2. *To rekey a table (for example, the oe.customer) using a different encryption algorithm (for example, AES128), execute the following statement:*

```
SQL> ALTER TABLE OE.CUSTOMERS REKEY USING 'AES128';
```

```
Table altered.
```

Rekeying a table key

3. *Change a master key by executing the following statement (in our example, keystore_password is welcome1):*

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED  
BY keystore_password WITH BACKUP;
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY welcome1  
  2  WITH BACKUP;
```

```
keystore altered.
```

Rekeying a master key

Backup and Recovery

1. *Connect to the RMAN as user who has the sysbackup privilege:*

```
$ rman target '"ernesto@orcl as sysbackup"'
```

2. *Configure encryption on a database level:*

```
RMAN> CONFIGURE ENCRYPTION FOR DATABASE ON;
```

3. *Backup a tablespace example in transparent mode:*

```
RMAN> BACKUP TABLESPACE EXAMPLE tag 'tran_mode';
```

4. *Enable dual mode encryption and backup tablespace example in dual mode:*

```
RMAN> SET ENCRYPTION ON IDENTIFIED BY
```

```
"password_1";RMAN> BACKUP TABLESPACE EXAMPLE tag  
'dual_mode';
```

5. *Enable password mode and backup tablespace example in password mode:*

```
RMAN> SET ENCRYPTION ON IDENTIFIED BY "password_2"  
ONLY; RMAN> BACKUP TABLESPACE EXAMPLE tag 'pass_mode';
```

Database Vault

Registering Database Vault

To register Database Vault with Oracle Database 12c when the database is already installed, perform the following steps:

1. *Connect to the root container as a user who has privileges to create users and grant create session and set container privileges (for example, c##maja):*

```
$ sqlplus c##maja
```

2. *Create two users (for example, c##dbv_owner and c##dbv_acctmgr) and grant them create session and set container privileges:*

```
SQL> create user c##dbv_owner identified by oraDVO123 CONTAINER =  
ALL; SQL> grant create session, set container to c##dbv_owner  
CONTAINER = ALL; SQL> create user c##dbv_acctmgr identified by  
oraDVA123 CONTAINER = ALL; SQL> grant create session, set container  
to c##dbv_acctmgr CONTAINER = ALL;
```

3. *Connect to the root as a SYS user:*

```
SQL> connect sys as sysdba
```

4. *Configure the Database Vault users:*

```
SQL> begin DVSYS.CONFIGURE_DV ( dvowner_uname =>  
'c##dbv_owner', dvacctmgr_uname => 'c##dbv_acctmgr'); end; /
```

5. *Execute the utlrp.sql script:*

```
SQL> @?/rdbms/admin/utlrp.sql
```

**6. *Connect to the root as the Database Vault Owner user that you just
configured (for example, the c##dbv_owner):***

```
SQL> connect c##dbv_owner/oraDVO123
```

7. *Enable Oracle Database Vault:*

```
SQL> exec DBMS_MACADM.ENABLE_DV
```

8. *Connect as a SYS user:*

```
SQL> CONNECT / AS SYSDBA
```

9. *Restart the database.*

For each PDB, perform step 3 through step 8 and then close and reopen the pluggable database (for example, PDB1).

```
SQL> alter pluggable database pdb1 close immediate;  
SQL> alter pluggable database pdb1 open;
```

Preventing users from exercising system privileges on schema objects

1. *Connect to a pluggable database (for example, pdb1) as a Database Vault account manager (for example, c##dbv_acctmgr):*

SQL> connect c##dbv_acctmgr@pdb1

2. *Create a new local user in the pluggable database (for example, usr1):*

SQL> create user usr1 identified by oracle;

3. *Connect to the pluggable database as a common user who has a DBA role in pdb1 (for example, c##ernesto):*

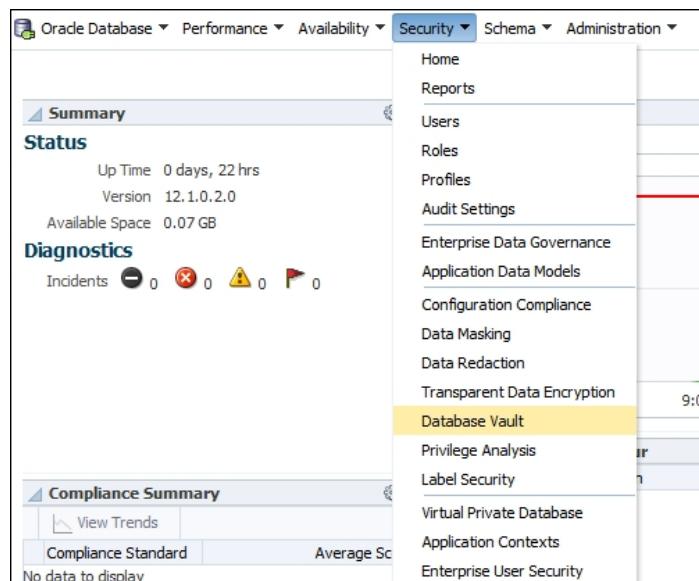
SQL> connect c##ernesto@pdb1

4. *Grant the select privilege on the table HR.EMPLOYEES and the create session privilege to the user usr1:*

SQL> grant select on hr.employees to usr1;

SQL> grant create session to usr1;

5. *Connect to the Enterprise Manager Cloud Control 12c (EM) as a privileged user (SYSMAN or some other privileged user, for example, ernesto). From Security drop-down menu, choose Database Vault*



Selecting Database Vault

6. *Log in to the pluggable database PDB1 as a user who is the Database Vault Owner*

Database Login

* Database	PDB1	<input type="button" value=""/>
* Username	c##dbv_owner	<input type="button" value=""/>
* Password	*****	<input type="button" value=""/>
Role	Normal	<input type="button" value=""/>
<input checked="" type="checkbox"/> Save As <input type="text" value="CDB1_DBV_OWNER"/>		
<input type="checkbox"/> Set As Preferred Credentials		
<input type="button" value="Login"/>		<input type="button" value="Cancel"/>

7. *On the next page, click on the Administration tab*

Oracle Database Vault

Home **Administration**

Logged in as C##DBV_OWNER

Page Refreshed Jun 13, 2015 9:43:48 PM CEST

General

Status Enabled	<input type="button" value="Disable"/>
Realms 7 0	
Command Rules 8 0	
Attempted Violations 4 (Last 24 Hours)	
Database Vault Policy Changes 0 (Last 24 Hours)	
Logged in as C##DBV_OWNER	<input type="button" value="Change Password"/>

Attempted Violations

Time Series View Data Last 24 hours

Top 5 Attempted Violations

Type Realms

HR_Realm(3)	75%
Oracle System Privilege and Role Management	25%
Realm(1)	

Top 5 Attempted Violators

Type Users

SYS(3)	75%
C##ZORAN(1)	25%

Database Vault Policy Propagation

Database Vault Policy Propagation
(Use this feature to securely propagate Database Vault Policies to multiple databases)

Database Vault Reports

Configuration Issues Reports
Enforcement Audit Reports
Configuration Changes Audit Reports

Alerts

Severity	Category	Name	Message	Alert Triggered
(No alerts)				

Switching to the Administration tab

8. *Create HR_Realm, as shown in the following figures. First, click on the Create button.*

The screenshot shows the Oracle Database Vault Administration interface. The top navigation bar has tabs for 'Home Page' and 'Administration'. Under 'Administration', the 'Database Vault Components' section is selected. On the left, a sidebar lists 'Realms', 'Command Rules', 'Rules', 'Rule Sets', 'Factors', 'Factor Types', 'Secure Application Roles', 'OLS Integration', and 'Database Vault Roles'. The main content area is titled 'Realms' and contains a brief description: 'Oracle Database Vault realms provide the ability to create protection zones around database objects that p...'. Below this is a 'Search' section with a 'Realm Name' input field and a 'Go' button. A message says 'The search returns all matches beginning with the string you enter. You can use the wildcard symbol (%) in...'. There is a toolbar with 'View', 'Create' (highlighted in yellow), 'Edit', 'Delete', and a checkbox for 'Show Oracle defined realms'. A sub-table below shows a single row: 'Realm Name' (HR_Realm) and 'Audit Options' (Audit on Failure). The message 'no data found' is displayed.

9. Name the realm (for example, *HR_Realm*) and leave default values for other parts of the form.

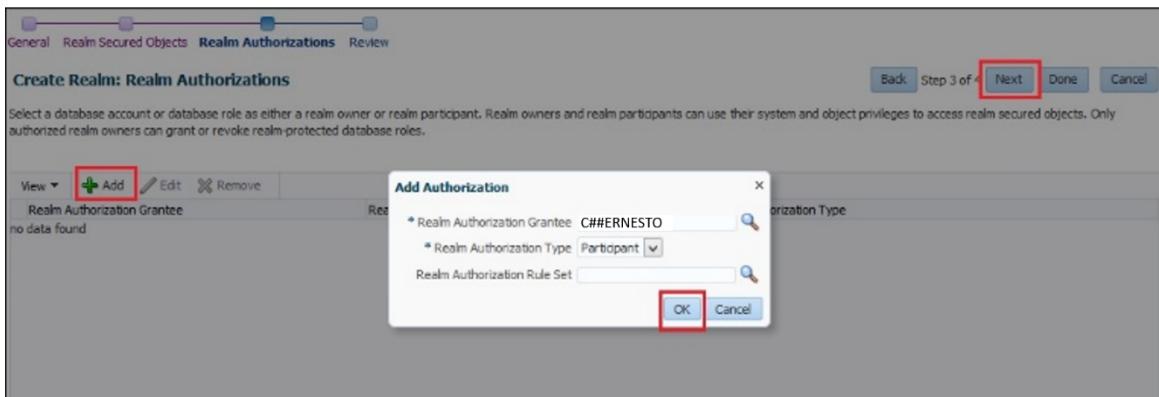
This screenshot shows the 'Create Realm: General' step of a four-step wizard. The top navigation bar includes 'General', 'Realm Secured Objects', 'Realm Authorizations', and 'Review'. The main area is titled 'Create Realm: General' and contains instructions: 'Define a Realm to control access to protected objects. If you mark a realm as mandatory, objects are protected from object owners accessing the data and other users exercising system or object privileges.' It has fields for 'Name' (HR_Realm), 'Description' (empty), 'Mandatory Realm' (unchecked), 'Status' (radio buttons for 'Enabled' and 'Disabled' with 'Enabled' selected), and 'Audit Options' (radio buttons for 'Audit Disabled', 'Audit on Success', 'Audit on Failure' (selected), and 'Audit on Success or Failure'). Navigation buttons at the top right include 'Back', 'Step 1 of 4', 'Next' (highlighted in yellow), 'Done', and 'Cancel'.

10. Securing all tables in HR schema.

This screenshot shows the 'Create Realm: Realm Secured Objects' step of a four-step wizard. The top navigation bar includes 'General', 'Realm Secured Objects' (highlighted in yellow), 'Realm Authorizations', and 'Review'. The main area is titled 'Create Realm: Realm Secured Objects' and instructs: 'Specify schema objects or database roles that should be protected by the realm. When specifying a role, please enter % in the Owner field.' It shows a table with one row: 'Owner' (HR) and 'Object Name' (%). A toolbar above the table includes 'View', 'Add' (with a red box around it), 'Edit', and 'Remove'. An 'Add Secured Object' dialog box is open in the foreground, containing fields: 'Owner' (HR), 'Object Type' (TABLE), and 'Object Name' (%). Navigation buttons at the top right include 'Back', 'Step 2 of 4', 'Next' (highlighted in yellow), 'Done', and 'Cancel'.

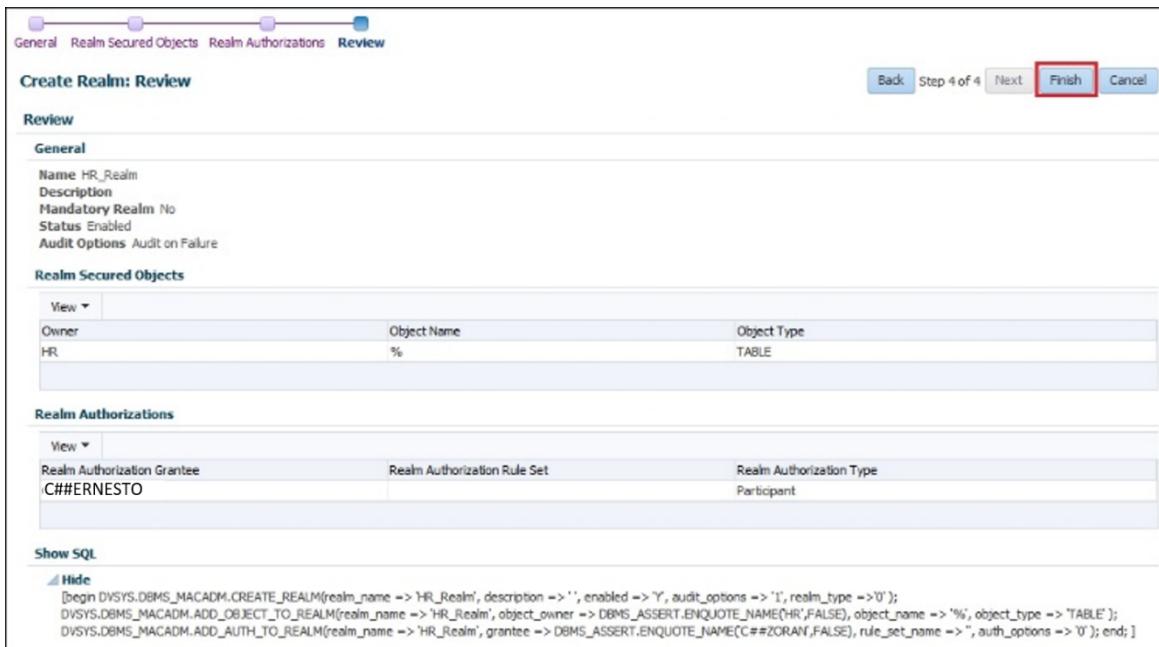
Adding secured objects

11. Add realm participant (for example, C##ERNESTO).



Adding authorized user(s)

12. After you make sure that you chose the options you wanted, click on the Finish button.



Reviewing and clicking on the Finish button

13. Verify that the usr1 and hr users can view data in the HR.EMPLOYEES table:

SQL> connect usr1@pdb1

SQL> select count(*) from hr.employees; COUNT(*) ----- 107

SQL> connect hr@pdb1

SQL> select count(*) from hr.employees; COUNT(*) ----- 107

14. To provide better security, edit the realm *HR_*Realm and select the checkbox *Mandatory Realm*.

The screenshot shows the Oracle Database Vault Administration interface. The left sidebar has a 'Realms' section with links to Command Rules, Rules, Rule Sets, Factors, Factor Types, Secure Application Roles, OLS Integration, and Database Vault Roles. The main area is titled 'Realms' and contains a brief description of what Oracle Database Vault realms do. Below is a search bar with 'Realm Name' and a 'Go' button. A table lists realms, with 'HR_Realm' selected. The table columns are Realm Name, Audit Options, Enabled, and Mandatory Realm. The 'Edit' button for HR_Realm is highlighted with a red box.

Realm Name	Audit Options	Enabled	Mandatory Realm
HR_Realm	Audit on Failure	✓	

*Editing HR_*Realm

The screenshot shows the 'Edit Realm : HR_Realm: General' step of a wizard. The top navigation bar includes 'General', 'Realm Secured Objects', 'Realm Authorizations', and 'Review'. The status bar says 'Step 1 of 4'. The 'Done' button is highlighted with a red box. The form fields include: 'Name' (HR_Realm), 'Description' (empty), 'Mandatory Realm' (checkbox checked, highlighted with a red box), 'Status' (radio buttons for 'Enabled' and 'Disabled' - 'Enabled' is selected), and 'Audit Options' (radio buttons for 'Audit Disabled', 'Audit on Success', 'Audit on Failure' - 'Audit on Failure' is selected). The 'Audit on Success or Failure' option is also available.

Mandatory Realm checkbox

15. Clicking on the Finish button.

The screenshot shows the 'Edit Realm' dialog box in Oracle Enterprise Manager Cloud Control 12c. It is on the 'Review' tab, which is the fourth step of a four-step process. The 'General' section shows a realm named 'HR_Realm' with 'Description' and 'Mandatory Realm' set to 'Yes'. The 'Status' is 'Enabled' and 'Audit Options' is 'Audit on Failure'. The 'Realm Secured Objects' section lists an object owned by 'HR' with a '%' object name and 'TABLE' type. The 'Realm Authorizations' section shows an authorization for 'C##ERNESTO' with a rule set and 'Participant' type. A 'Show SQL' button is present, showing the underlying PL/SQL code for the update. The 'Finish' button at the top right is highlighted with a red box.

Leaving other settings as they were and clicking on the Finish button

Securing roles

1. *Connect to the pluggable database PDB1 as a SYS user:*

SQL> connect sys@pdb1 as sysdba

2. *Create the role role1:*

SQL> create role role1;

3. *Grant the create session and select any table privileges to the role:*

SQL> grant create session, select any table to role1;

4. *Create realm ROLE1_Realm in Enterprise Manager Cloud Control 12c.*

Create Realm: General

Define a Realm to control access to protected objects. If you mark a realm as mandatory, objects are protected from object owners accessing the data and other users exercising system or object privileges.

* Name: ROLE1_Realm

Description:

Mandatory Realm

Status: Enabled Disabled

Audit Options: Audit Disabled Audit on Success Audit on Failure Audit on Success or Failure

Back Step 1 of 4 **Next** Done Cancel

Creating ROLE1_Realm

5. Add realm-secured objects.

Create Realm: Realm Secured Objects

Specify schema objects or database roles that should be protected by the realm. When specifying a role, please enter % in the Owner field.

View	Add	Edit	Remove
Owner	Object Name	Object Type	
SYS	ROLE1	ROLE	

Back Step 2 of 4 **Next** Done Cancel

Adding secured objects

6. Add realm authorizations and click on the Next button.

Create Realm: Realm Authorizations

Select a database account or database role as either a realm owner or realm participant. Realm owners and realm participants can use their system and object privileges to access realm secured objects. Only authorized realm owners can grant or revoke realm-protected database roles.

View	Add	Edit	Remove
Realm Authorization Grantee	Realm Authorization Rule Set	Realm Authorization Type	
C##ERNESTO		Owner	

Back Step 3 of 4 **Next** Done Cancel

Realm authorizations

7. Review and click on the Finish button

General Realm Secured Objects Realm Authorizations Review

Create Realm: Review

Review

General

Name	ROLE1_Realm
Description	
Mandatory Realm	No
Status	Enabled
Audit Options	Audit on Failure

Realm Secured Objects

View	Owner	Object Name	Object Type
	SYS	ROLE1	ROLE

Realm Authorizations

View	Realm Authorization Grantee	Realm Authorization Rule Set	Realm Authorization Type
	CH#ERNESTO		Owner

Show SQL

Hide

```
[begin DV$SYS.DBMS_MACADM.CREATE_REALM(realm_name => 'ROLE1_Realm', description => '', enabled => 'Y', audit_options => '1', realm_type =>'0');
DV$SYS.DBMS_MACADM.ADD_OBJECT_TO_REALM(realm_name => 'ROLE1_Realm', object_owner => DBMS_ASSERT.ENQUOTE_NAME('SYS',FALSE), object_name => 'ROLE1', object_type => 'ROLE');
DV$SYS.DBMS_MACADM.ADD_AUTH_TO_REALM(realm_name => 'ROLE1_Realm', grantee => DBMS_ASSERT.ENQUOTE_NAME('C##ZORAN',FALSE), rule_set_name => '', auth_options => '1');
end; ]
```

8. Connect to the pluggable database PDB1 as a SYS user:

SQL> connect sys@pdb1 as sysdba

9. Verify that SYS still can revoke/grant privileges from/to role role1, even though role1 is protected by the realm:

SQL> revoke select any table from role1;

SQL> grant drop any synonym to role1;

10. Edit the realm ROLE1_Realm and make it mandatory (select the Mandatory Realm checkbox).

General Realm Secured Objects Realm Authorizations Review

Edit Realm : ROLE1_Realm: General

Step 1 of 4 Back Next Done Cancel

Define a Realm to control access to protected objects. If you mark a realm as mandatory, objects are protected from object owners accessing the data and other users exercising system or object privileges.

* Name	ROLE1_Realm
Description	
Mandatory Realm	<input checked="" type="checkbox"/>
Status	<input checked="" type="radio"/> Enabled <input type="radio"/> Disabled
Audit Options	<input type="radio"/> Audit Disabled <input type="radio"/> Audit on Success <input checked="" type="radio"/> Audit on Failure <input type="radio"/> Audit on Success or Failure

Editing realm

11. Review and confirm the change of ROLE1_Realm.

The screenshot shows the 'Edit Realm : ROLE1_Realm: Review' page. At the top, there are tabs: General, Realm Secured Objects, Realm Authorizations, and Review. The 'Review' tab is selected. Below the tabs, there are sections for 'General', 'Realm Secured Objects', and 'Realm Authorizations'. The 'General' section shows details: Name: ROLE1_Realm, Description: , Mandatory Realm: Yes, Status: Enabled, Audit Options: Audit on Failure. The 'Realm Secured Objects' section shows a table with one row: Owner %, Object Name ROLE1, Object Type ROLE. The 'Realm Authorizations' section shows a table with one row: Realm Authorization Grantee C##ERNESTO, Realm Authorization Rule Set, Realm Authorization Type Owner. At the bottom, there is a 'Show SQL' link and a 'Hide' link. The 'Show SQL' link displays the following PL/SQL code:

```
[begin DVSYS.DBMS_MACADM.UPDATE_REALM(realm_name => 'ROLE1_Realm', description => '', enabled => 'Y', audit_options => 'I', realm_type =>'1'); end; ]
```

Preventing users from executing specific command on specific object

Create a command rule by following these steps:

The screenshot shows the 'Command Rules' page under the 'Administration' tab. On the left, there is a sidebar with links: Realms, Command Rules (highlighted with a red box), Rules, Rule Sets, Factors, Factor Types, Secure Application Roles, OLS Integration, and Database Vault Roles. The main area has a 'Search' section with fields for 'Rule Set Name' and 'Command'. Below it is a table header with columns: View, Create, Command, Object Owner, Object Name, Rule Set Name, Enabled, and Last Updated Date. A note states 'no data found'.

Creating a command rule

In the **Command** field, write UPDATE; in the **Applicable Object Owner** field, write OE; in the **Applicable Object Name** field, write ORDERS; and select **Disabled** for **Rule Set**.

Create Command Rule

This page allows you to create or edit a command rule that can be associated with an existing Database Vault rule set.

* Command: UPDATE

Status: Enabled
 Disabled

* Applicable Object Owner: OE

* Applicable Object Name: ORDERS

* Rule Set: Disabled

Show SQL Cancel OK

A Command rule to secure the UPDATE operation on OE.Orders

Creating a rule set

1. Go to Rule Sets component and then click on Create

Oracle Database Vault

Home Page Administration

Database Vault Components

Realms

Command Rules

Rules

Rule Sets

Factors

Factor Types

Secure Application Roles

OLIS Integration

Database Vault Roles

Rule Sets

A Rule Set is a collection of one or more rules that you can associate with a Realm Authorization, Command Rule, Factor Assignment, or Secure Application Role. The Rule Set evaluates to true or false based on the evaluation of each rule it contains and the evaluation type (All True or Any True). A Rule Set can be static so that it is evaluated only once during a user session.

Search

Rule Set Name: Go

The search returns all matches beginning with the string you enter. You can use the wildcard symbol (%) in the search string.

View	Create	Edit	Delete	<input type="checkbox"/> Show Oracle defined Rule Sets		
Rule Set Name	Static Rule Set	Error Handling	Audit Options	Evaluation Options	Enabled	Last Updated Date
no data found						

2. As a name, enter Working Hours and click on Next. For Evaluation Options, choose All True.

General Associate with Rules Error Handling and Audit Options Review

Create Rule Set: General

Step 1 of 4 Back Next Done Cancel

Enter the general information required to create a Rule Set.

* Rule Set Name: Working Hours

Description:

Static Rule Set

Status: Enabled
 Disabled

Evaluation Options: All True
 Any True

Our rule set "Working Hours"

- 3. Add two rules (*Is Working Day* and *Is Working Hour*) by clicking on *Create Rule* before adding each of them. Enter the details in *Rule Name* and *Rule Expression* as shown in the next figure. After you added both rules, click on *Next*.**

Rule Name	Rule Expression
Is Working Day	to_char(sysdate,'d') between '2' and '6'
Is Working Hour	to_char(sysdate,'hh24') between '09' and '17'

Create two rules

- 4. Leave all options on defaults and click on *Next*.**

Error Handling	<input checked="" type="radio"/> Show Error Message <input type="radio"/> Do Not Show Error Message
Fail Code	<input type="text"/>
Fail Message	<input type="text"/>
Custom Event Handler Options	<input checked="" type="radio"/> Handler Disabled <input type="radio"/> Execute on Failure <input type="radio"/> Execute on Success <input type="radio"/> Execute on Success or Failure
Custom Event Handler Logic	<input type="text"/>
Audit Options	<input type="radio"/> Audit Disabled <input type="radio"/> Audit on Success <input checked="" type="radio"/> Audit on Failure <input type="radio"/> Audit on Success or Failure

Error handling and audit options

- 5. Click on *Finish*.**

Create Rule Set: Review

Review

This review screen shows the data and options that are selected. If everything is correct, click "Finish" to create the Rule Set. Use the "Back" button if you want to change any data or option.

General

Rule Set Name	Working Hours	Static Rule Set No	Evaluation Options	All True
Description	Status Y			

Rules Associated

Name	Expression
Is Working Day	to_char (sysdate,'d') between '2' and '6'
Is Working Hour	to_char (sysdate, hh24) between '09' and '17'

Error Handling and Audit Options

Error Handling	Show Error Message	Fail Message	Custom Event Handler Logic
Fail Code	Custom Event Handler	Handler Disabled Options	Audit Options
Audit on Failure			

Show SQL

Hide

```
[begin DECLARE x VARCHAR2(40);static_option BOOLEAN := FALSE; BEGIN x:='N'; IF x = 'Y' THEN static_option := TRUE; ELSE static_option := FALSE; END IF;
DVSYS.DBMS_MACADM.CREATE_RULE_SET(rule_set_name =>'Working Hours',description =>'',enabled =>'Y',eval_options => 1,audit_options => 1,fail_options => 1,fail_message => '',
fail_code => '',handler_options => 0,handler =>"',is_static => static_option);END;DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET(rule_set_name =>'Working Hours',rule_name =>'Is Working Day',rule_order =>'1',enabled =>'Y');DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET(rule_set_name =>'Working Hours',rule_name =>'Is Working Hour',rule_order =>'1',enabled =>'Y');
end; ]
```

Finish

Creating a secure application role

1. *Create rule set with name Can Access Customer Data and with rule DVF.F\$MACHINE = <your host name> (for example, name it: Is Local Machine). In our case, hostname is host01.challengeernesto.com.*

Create Rule Set: Associate with Rules

Add existing rules to the Rule Set or create new rules for the Rule Set.

Create Rule

Rule Name	Rule Expression
Is Local Machine	DVF.F\$MACHINE = 'host01.challengeernesto.com'

Is a Local Machine rule

2. *In the Database Vault Components panel, click on the Secure Application Roles link and then click on the Create button.*

Create a secure application role

3. ***Define secure application role settings. In our case, we secure the role cust_role and condition for enablement is defined by the Can Access Customer Data rule set.***

Define secure application role

Using Database Vault to implement that administrators cannot view data

1. ***Connect to the pluggable database PDB1 as the user c##dbv_acctmgr:***

SQL> connect c##dbv_acctmgr@pdb1 SQL> create user orders_dba identified by oracle1; SQL> create user orders_user identified by oracle2;

2. ***Connect to the pluggable database PDB1 as a SYS user and execute the following statements:***

SQL> connect sys@pdb1 as sysdba SQL> grant dba to orders_dba;
SQL> grant create session to orders_user; SQL> grant select on oe.orders to orders_user; SQL> grant update on oe.orders to orders_user; SQL> create role ord_usr_role; SQL> grant ord_usr_role to orders_user;

3. ***Create a realm that protects all objects in OE schema and authorize user orders_dba as owner .***

Create Realm: Review

Review

General

Name	OE_Realm
Description	
Mandatory Realm No	
Status	Enabled
Audit Options	Audit on Failure

Realm Secured Objects

View	Owner	Object Name	Object Type
	OE	%	%

Realm Authorizations

View	Realm Authorization Grantee	Realm Authorization Rule Set	Realm Authorization Type
	ORDERS_DBA		Owner

Create realm OE_Realm

4. *Create realm that protects the ORDUSRROLE role and authorize the user c##ernesto as owner*

General Realm Secured Objects Realm Authorizations Review

Create Realm: Review

Review

General

Name	Orders_Role_Realm
Description	
Mandatory Realm No	
Status	Enabled
Audit Options	Audit on Failure

Realm Secured Objects

View	Owner	Object Name	Object Type
	%	ORDUSRROLE	ROLE

Realm Authorizations

View	Realm Authorization Grantee	Realm Authorization Rule Set	Realm Authorization Type
	C##ERNESTO		Owner

Show SQL

```

 Hide
[begin DV$SYS.DBMS_MACADM.CREATE_REALM(realname => 'Orders_Role_Realm', description => '', enabled => 'Y', audit_options => '1', realm_type =>'0'); DV$SYS.DBMS_MACADM.ADD_OBJECT_TO_REALM(realname => 'Orders_Role_Realm', object_owner => DBMS_ASSERT.ENQUOTE_NAME('%',FALSE), object_name => 'ORDUSRROLE', object_type => 'ROLE'); DV$SYS.DBMS_MACADM.ADD_AUTH_TO_REALM(realname => 'Orders_Role_Realm', grantee => DBMS_ASSERT.ENQUOTE_NAME('C##ERNESTO',FALSE), rule_set_name => '', auth_options => '1'); end;]

```

Create realm Orders_Role_Realm

5. *Create rule set (exp. role check) that has one rule with name Has ORDUSRROLE and expression DBMS_MACUTL.USER_HAS_ROLE_VARCHAR('ORDUSRROLE') = 'Y'*

General Associate with Rules Error Handling and Audit Options Review

Create Rule Set: Review

Review

This review screen shows the data and options that are selected. If everything is correct, click "Finish" to create the Rule Set. Use the "Back" button if you want to change any data or option.

General

Rule Set Name	Role check	Static Rule Set	No	Evaluation Options	All True
Description	Status Y				

Rules Associated

View ▾

Name	Expression
Has ORD_USR_ROLE	DBMS_MACUTL.USER_HAS_ROLE_VARCHAR('ORDUSR_ROLE') = 'Y'

Error Handling and Audit Options

Error Handling	Show Error Message	Fail Message	Custom Event Handler Logic
Fail Code	Custom Event Handler	Handler Disabled	Audit Options
Options Audit on Failure			

Show SQL

Hide

```
[begin DECLARE x VARCHAR2(40);static_option BOOLEAN := FALSE; BEGIN x:='N'; IF x = 'Y' THEN static_option := TRUE; ELSE static_option := FALSE; END IF; DBMS_DBMS_MACADM.CREATE_RULE_SET(rule_set_name =>'Role check', description =>' ', enabled =>'Y', eval_options => 1, audit_options => 1, fail_options => 1, fail_message => "", fail_code => "", handler_options => 0, handler => "", is_static => static_option);END;
DBMS_DBMS_MACADM.ADD_RULE_TO_RULE_SET(rule_set_name =>'Role check', rule_name =>'Has ORDUSR_ROLE', rule_order =>'1', enabled =>'Y');end; ]
```

Create Rule Set Role check

6. Create a command rule for Select on all objects in OE schema, and as rule set, select one that you created in previous step (exp role check)

Create Command Rule

This page allows you to create or edit a command rule that can be associated with an existing Database Vault rule set.

Show SQL Cancel OK

* Command:

Status: Enabled Disabled

* Applicable Object Owner:

* Applicable Object Name:

* Rule Set:

Create a command rule

Running Oracle Database Vault reports

Let's connect as user system and violate some restrictions. First, we are going to select from hr schema, which is going to violate HR realm, and second, we are going to update sal in the scott.emp table, which is going to violate the command rule (we are updating it outside of working hours).

1.
SQL> connect system@pdb1

2.

SQL> select count(*) from hr.employees;

3.

SQL> update scott.emp set sal = sal*1.20 where empno = 7839;

Let's see reports for these violations:

1. *Go to Database Vault home page*
2. *Click on Enforcement Audit Reports*

The screenshot shows the Oracle Database Vault Home page. On the left, there are sections for General (Status Enabled, Realms, Command Rules, Attempted Violations 10, Database Vault Policy Changes 3), Database Vault Policy Propagation, and Database Vault Reports (Configuration Issues Reports, Enforcement Audit Reports, Configuration Changes Audit Reports). A red box highlights the 'Enforcement Audit Reports' link. On the right, there are two pie charts: 'Attempted Violations' (33%, 33%, 33%) and 'Top 5 Attempted Violators' (40%, 20%, 20%, 10%, 10%). Below the charts is a table for Alerts (Severity, Category, Name, Message, Alert Triggered).

3. *Click on Realm Audit Report. Observe the line marked in red (violation from step 2 is audited).*

The screenshot shows the Oracle Database Vault Reports page with the 'Realm Audit Report' selected. The search criteria include Match (All), Timestamp, Account, User Host, Realm Name, Rule Set, and Command. The results table lists audit records with columns: Timestamp, Account, Return Code, User Host, Instance Number, Realm Name, Rule Set, Command, and Violation. A red box highlights the first row in the table, which corresponds to the violation from step 2.

Timestamp	Account	Return Code	User Host	Instance Number	Realm Name	Rule Set	Command	Violation
2015-06-15 01:31:51.0	SYSTEM	1031	host01.challenger...	0	HR_Realm		SELECT COUNT(*) FROM HR.EMPLOYEES	Realm Violation Audit
2015-06-15 05:59:01.0	ORDERS_DBAA	47401	host01.challenger...	0	OE_Realm		GRANT SELECT ON OE.ORDERS TO OR...	Realm Violation Audit
2015-06-15 05:59:01.0	CHERNESTO	47410	host01.challenger...	0	Orders_Role_Realm		GRANT ORDUSR_ROLE TO ORDERS_U...	Realm Violation Audit
2015-06-14 17:26:54.0	ERNESTO	47410	host01.challenger...	0	Oracle System Privilege and Role Mana...		GRANT CREATE SESSION TO USR2	Realm Violation Audit

4. *Next, click on Command Rule Audit Report. Observe the line marked in red (violation from step 3 is audited).*

The Command Rule Audit Report shows audit records generated by command rule processing operations. When you configure a command rule, you can set it to audit the rule set processing results.

Timestamp	User Host	Account	Instance	Rule No.	Command Rule	Rule Set	Command	Violation
2015-06-15 01:35:35.0	1031	host01.challengeren...	SYSTEM	0	UPDATE	Working Hours	UPDATE SCOTT.EMP SET SAL = SAL*1.20 WHERE...	Command Failure Audit
2015-06-15 02:39:46.0	1031	host01.challengeren...	ORDERS_DBA	0	SELECT	Role check:	SELECT COUNT(*) FROM OE.ORDERS	Command Failure Audit
2015-06-14 17:26:00.0	1031	host01.challengeren...	ERNESTO	0	CREATE USER	Can Maintain Accounts/Profiles	CREATE USER USR2 IDENTIFIED BY "	Command Failure Audit
2015-06-14 15:15:40.0	1031	host01.challengeren...	SYS	0	UPDATE	Working Hours	UPDATE SCOTT.EMP SET SAL = SAL + 300 WHERE...	Command Failure Audit
2015-06-14 14:22:50.0	1031	host01.challengeren...	SYS	0	UPDATE	Working Hours	UPDATE SCOTT.EMP SET SAL = SAL + 300 WHERE...	Command Failure Audit

Disabling Database Vault

1. Go to Database Vault home page of your database or pluggable database and click on Disable:

General

- Status: Enabled
- Disable
- Realms: 20 (2g)
- Command Rules: 12 (Last 24 hours)
- Attempted Violations: 12 (Last 24 hours)
- Database Vault Policy Changes: 3 (Last 24 hours)
- Logged in as: C##DBV_OWNER Change Password

Database Vault Policy Propagation

Database Vault Policy Propagation (Use this feature to securely propagate Database Vault Policies to multiple databases)

Database Vault Reports

- Configuration Issues Reports
- Enforcement Audit Reports
- Configuration Changes Audit Reports

Alerts

Severity	Category	Name	Message	Alert Triggered
(No alerts)				

2. Click on continue in a small pop-up window

Disable Database Vault

Confirmation

Are you sure you want to disable Database Vault? Note: This operation requires database restart.

Cancel Continue

Or

Connect to the database as Database Vault owner and disable it through command line:

SQL> EXEC DBMS_MACADM.DISABLE_DV;

3. *Connect to your database or pluggable database and restart it:*

SQL> connect / as sysdba SQL> alter pluggable database pdb1 close immediate; SQL> alter pluggable database pdb1 open;

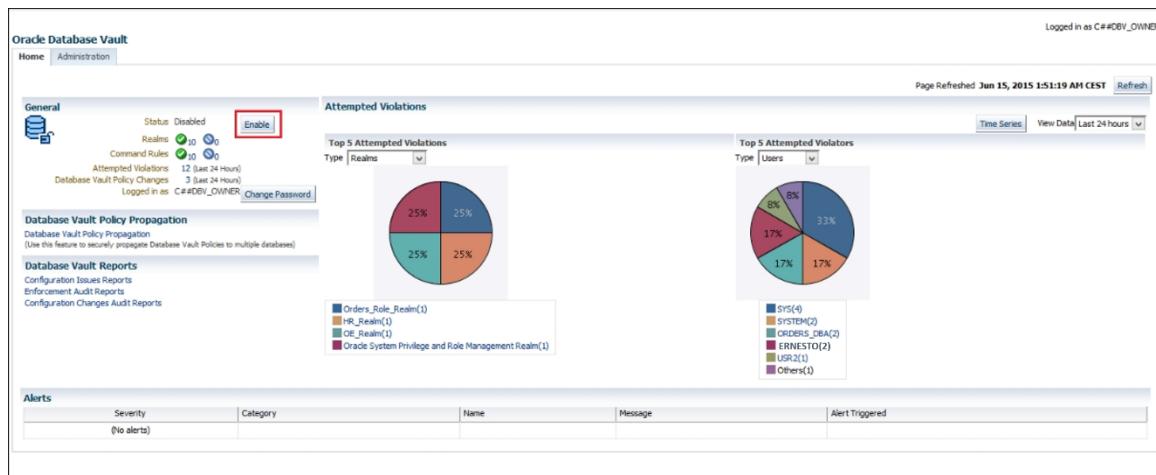
4. *Confirm that Database Vault is disabled:*

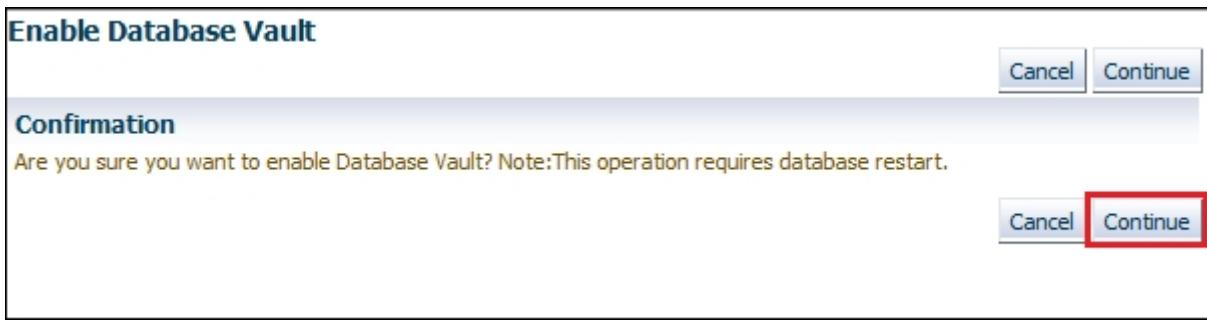
SQL> connect c##dbv_owner@pdb1

SQL> SELECT PARAMETER, VALUE FROM V\$OPTION WHERE
PARAMETER = 'Oracle Database Vault'; PARAMETER
VALUE ----- ----- Oracle Database Vault
FALSE

Re-enabling Database Vault

1. *Go to Database Vault home page of your database or pluggable database and click on Enable, then click on continue in a small pop-up window.*





Or

Connect to the database as Database Vault owner and enable it through command line:

```
SQL> EXEC DBMS_MACADM.ENABLE_DV;
```

2. *Connect to your database or pluggable database and restart it:*

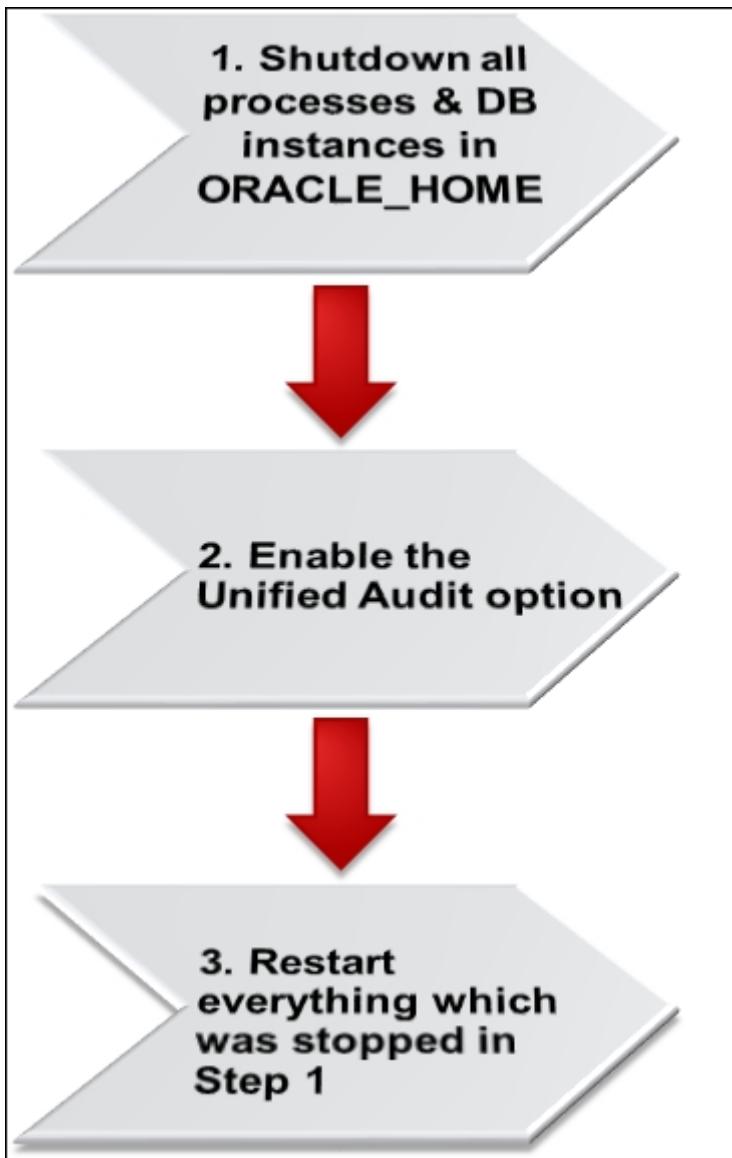
```
SQL> connect / as sysdba  
SQL> alter pluggable database pdb1 close immediate;  
SQL> alter pluggable database pdb1 open;
```

3. *Confirm that Database Vault is enabled:*

```
SQL> connect c##dbv_owner@pdb1  
SQL> SELECT PARAMETER, VALUE FROM V$OPTION WHERE  
PARAMETER = 'Oracle Database Vault';  
PARAMETER  
VALUE ----- Oracle Database Vault  
TRUE
```

Enabling Unified Auditing mode

The process of enabling unified auditing is depicted in the next figure:



- 1. In our case, there is only one database instance. Connect to the instance as sysoper and shut it down. Also, stop the listener:*

```
$ sqlplus / as sysoper
SQL> shutdown immediate SQL> exit $ lsnrctl stop
```

- 2. Relink Oracle binaries with the uniaud_on option:*

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk uniaud_on ioracle
```

- 3. Start the listener and the database instance:*

```
$ lsnrctl start $ sqlplus / as sysoper SQL> startup
```

To verify that unified auditing is enabled, issue the following SQL statement:

```
SQL> SELECT PARAMETER, VALUE  
  2  from v$option  
  3 where PARAMETER = 'Unified Auditing';
```

You should see that value for Unified Auditing parameter is true:

PARAMETER	VALUE
Unified Auditing	TRUE

Configuring whether loss of audit data is acceptable

1. *Connect to the database as user who has the audit_admin role (for example, jack):*

```
SQL> connect jack
```

2. *If you want audit records to be immediately written to the unified audit trail set immediate-write mode:*

```
SQL> EXEC DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_PROPERTY  
(DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,DBMS_AUDIT_MGM  
T.AUDIT_TRAIL_WRITE_MODE,  
DBMS_AUDIT_MGMT.AUDIT_TRAIL_IMMEDIATE_WRITE);
```

3. *Check that the mode is set to immediate-write:*

```
SQL> select * from dba_audit_mgmt_config_params  
where parameter_name='AUDIT WRITE MODE';
```

You should see that the value for the AUDIT WRITE MODE parameter is IMMEDIATE WRITE MODE:

PARAMETER_NAME	PARAMETER_VALUE	AUDIT_TRAIL
AUDIT WRITE MODE	IMMEDIATE WRITE MODE	UNIFIED AUDIT TRAIL

If you want audit records to be queued in memory and at later time persisted, then set the **queued-write mode**. Instead of step 2, execute:

```
SQL> EXEC  
DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_PROPERTY  
(DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,DBMS_AUDIT_M  
GMT.AUDIT_TRAIL_WRITE_MODE,  
DBMS_AUDIT_MGMT.AUDIT_TRAIL_QUEUED_WRITE);
```

Which roles do you need to have to be able to create audit policies and to view audit data?

1. *Connect to the database as a user who has the dba role (for example, maja):*

```
$ sqlplus maja
```

2. *Create the user jack and grant him the create session privilege and the audit_admin role.*

```
SQL> create user jack identified by pQ3s7a4w2;  
SQL> grant create session, audit_admin to jack;
```

3. *Create the user jill and grant her the create session privilege and the audit_viewer role.*

```
SQL> create user jill identified by t1m5_R2f3;  
SQL> grant create session, audit_viewer to jill;
```

Auditing RMAN operations

1. *Connect to the target database as a user who has the SYSBACKUP privilege (for example, tom).*

```
$ rman target ""tom@ora12cR1 AS SYSBACKUP""
```

2. *Backup the EXAMPLE tablespace and view information about backups:*

```
RMAN> backup tablespace EXAMPLE;  
RMAN> list backup;  
RMAN> exit
```

3. *Connect to the database as a user who has the DBA role (for example, maja):*

```
$ sqlplus maja
```

4. *Find the location of datafile for EXAMPLE tablespace:*

```
SQL> select file_name from dba_data_files where  
tablespace_name='EXAMPLE';  
FILE_NAME
```

```
/u01/app/oracle/oradata/ORA12CR1/datafile/  
o1_mf_example_9z79vpcj_.dbf
```

5. *Remove the EXAMPLE tablespace datafile:*

```
SQL> !rm /u01/app/oracle/oradata/ORA12CR1/datafile/  
o1_mf_example_9z79vpcj_.dbf
```

6. *Put the EXAMPLE tablespace offline:*

```
SQL> alter tablespace example offline immediate;  
SQL> exit
```

7. *Restore the EXAMPLE tablespace datafile:*

```
$ rman target ""tom@ora12cR1 AS SYSBACKUP""  
RMAN> restore tablespace EXAMPLE;
```

8. *Recover the EXAMPLE tablespace datafile:*

```
RMAN> recover tablespace EXAMPLE;  
RMAN> exit
```

9. *Put tablespace back online:*

```
$ sqlplus maja  
SQL> alter tablespace EXAMPLE online;
```

10. *To verify that RMAN operations were successfully audited, execute the following statements:*

```
SQL> connect jack
```

```
SQL> EXEC  
SYS.DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL;
```

```
SQL> select dbusername, rman_operation from unified_audit_trail where  
rman_operation is not null;
```

Auditing Data Pump operations

1. *Connect to the database as a user who has the audit_admin role (for example, jack):*

```
$ sqlplus jack
```

2. *Create an audit policy to audit Data Pump export operations:*

```
SQL> CREATE AUDIT POLICY DP_POLICY ACTIONS  
COMPONENT=datapump export;
```

3. *Enable the audit policy:*

```
SQL> AUDIT POLICY DP_POLICY;
```

4. *Export the table hr.departments:*

```
$ expdp maja@ora12cR1 dumpfile=test tables=hr.departments  
DIRECTORY=my_dir
```

5. *Verify that the export operation was successfully audited:*

```
SQL> connect jack  
SQL> select  
DP_TEXT_PARAMETERS1,DP_BOOLEAN_PARAMETERS1  
from unified_audit_trail  
where audit_type='Datapump' and dbusername='MAJA';
```

Auditing Database Vault operations

1. *Connect to the database as a user who has the audit_admin role (for example, jack):*

\$ connect jack

2. *Create the audit policy dbv_policy:*

```
SQL> CREATE AUDIT POLICY dbv_policy  
ACTIONS COMPONENT = DV Rule Set Failure on "Working  
Hours",realm  
violation on "HR Realm";
```

3. *Enable the audit policy dbv_policy:*

```
SQL> audit policy dbv_policy;
```

4. *Execute several statements that will cause generation of audit records:*

```
SQL> select * from oe.orders;
```

```
SQL> update hr.employees set salary=30000 where salary=24000;
```

Creating audit policies to audit privileges, actions and roles under specified conditions

1. *Connect to the database as a user who has the audit_admin role (for example, jack):*

\$ sqlplus jack

2. *Create audit policy my_policy1:*

```
SQL> CREATE AUDIT POLICY MY_POLICY1  
PRIVILEGES SELECT ANY TABLE  
ACTIONS CREATE TABLE, DROP TABLE;
```

3. *Create the audit policy role_con_policy:*

```
SQL> CREATE AUDIT POLICY ROLE_CON_POLICY  
ROLES HR_ROLE  
WHEN  
'SYS_CONTEXT("USERENV","HOST")="dbhost.orapassion.com"'  
EVALUATE PER SESSION;
```

4. *Create the audit policy hr_policy:*

```
SQL> CREATE AUDIT POLICY HR_POLICY  
ACTIONS SELECT,INSERT,UPDATE,DELETE ON  
HR.DEPARTMENTS;
```

5. *Create the audit policy oe_policy:*

```
SQL> CREATE AUDIT POLICY OE_POLICY  
ACTIONS ALL ON OE.ORDERS;
```

Enabling audit policy

1. *Connect to the database as a user who has audit_admin role (for example, jack)*

```
SQL> connect jack
```

2. *Enable audit policy oe_policy in such way that it applies only to user JOHN*

```
SQL> audit policy OE_POLICY BY JOHN;
```

3. *Enable audit policy hr_policy to capture only successful events.*

```
SQL> AUDIT POLICY HR_POLICY WHENEVER SUCCESSFUL;
```

4. *Enable policy my_policy1 to audit unsuccessful events for all users except maja and ernesto.*

```
SQL> audit policy my_policy1 EXCEPT MAJA,  
ERNESTO WHENEVER NOT SUCCESSFUL;
```

5. *Enable audit policy role_con_policy using default options.*

```
SQL> audit policy role_con_policy;
```

Finding information about audit policies and audited data

1. *Connect to the database as a user who has the audit_admin role (for example, jack):*

SQL> connect jack

2. *Find which unified audit policies are defined (exist in the database):*

SQL> select distinct policy_name

from audit_unified_policies;

SQL> desc audit_unified_policies

3. *View which unified audit policies are enabled:*

SQL> select * from audit_unified_enabled_policies;

4. *Connect to the database as the user john:*

SQL> connect john

5. *Execute several statements on the tables HR.EMPLOYEES, HR.DEPARTMENTS, and OE.ORDERS:*

SQL> create table t(a number(10));

SQL> select count(*) from oe.orders;

SQL> select first_name from hr.employees;

SQL> drop table t;

SQL> connect sys / as sysdba

SQL> create table hr.my_table(b varchar2(10));

SQL> connect john

SQL> drop table hr.my_table;

6. *Connect to the database as a user who has the audit_viewer role (for example, jill):*

SQL> connect jill

7. *View audit records:*

SQL> set linesize 250

SQL> col event_timestamp format a30

SQL> col action_name format a20

SQL> col unified_audit_policies format a20

SQL> col sql_text format a80

```
SQL> select event_timestamp, action_name,unified_audit_policies,  
sql_text from  
unified_audit_trail where DBUSERNAME = 'SYS' and ACTION_NAME  
NOT IN ('LOGON','LOGOFF') ORDER BY EVENT_TIMESTAMP  
DESC; SQL> select event_timestamp, action_name, unified_audit_policies,  
sql_text from unified_audit_trail where DBUSERNAME = 'JONH' and  
ACTION_NAME NOT IN ('LOGON','LOGOFF') ORDER BY  
EVENT_TIMESTAMP DESC;
```

Auditing application contexts

1. *Connect to the database as a user who has the audit_admin role (for example, jack):*

```
$ sqlplus jack
```

2. *Configure application context auditing:*

```
SQL> AUDIT CONTEXT NAMESPACE USERENV  
ATTRIBUTES SESSION_USER, SERVICE_NAME;
```

Audit succeeded.

```
SQL> AUDIT CONTEXT NAMESPACE USERENV ATTRIBUTES  
HOST BY jill; Audit succeeded.
```

3. *View for which application contexts audit data is going to be captured:*

```
SQL> set linesize 180 SQL> column namespace format A30 SQL>  
column attribute format A30 SQL> column user_name format A30  
SQL> select * from audit_unified_contexts;
```

4. *Connect user jill as follows:*

```
SQL> connect jill
```

5. *View audit records:*

```
SQL> SELECT APPLICATION_CONTEXTS FROM  
UNIFIED_AUDIT_TRAIL  
WHERE APPLICATION_CONTEXTS IS NOT NULL;
```

Purging audit trail

1. *Connect to the database as a user who has the audit_admin role (for example, jack):*

```
$ sqlplus jack
```

2. *View number of audit records in the unified audit trail before the cleanup:*

```
SQL> select count (*) from unified_audit_trail;
```

To perform the manual cleanup, execute:

```
SQL> exec DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(  
AUDIT_TRAIL_TYPE =>  
DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED)
```

To create a purge job:

```
SQL> exec DBMS_AUDIT_MGMT.CREATE_PURGE_JOB  
(AUDIT_TRAIL_TYPE =>  
DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,  
AUDIT_TRAIL_PURGE_INTERVAL => 24,  
AUDIT_TRAIL_PURGE_NAME => 'My_Job',  
USE_LAST_ARCH_TIMESTAMP => TRUE)
```

3. *View number of audit records in the unified audit trail after the cleanup:*

```
SQL> select count (*) from unified_audit_trail;
```

Disabling and dropping audit policies

1. *Connect to the database as a user who has the audit_admin role (for example, jack):*

```
$ sqlplus jack
```

2. *Verify that the policy is enabled:*

```
SQL> SELECT POLICY_NAME, ENABLED_OPT, USER_NAME,  
SUCCESS, FAILURE  
FROM AUDIT_UNIFIED_ENABLED_POLICIES;
```

3. *Disable the policy oe_policy:*

```
SQL> NOAUDIT policy oe_policy BY JOHN;
```

4. *Verify that oe_policy is disabled:*

```
SQL> select * from AUDIT_UNIFIED_ENABLED_POLICIES;
```

5. *Drop the policy oe_policy:*

```
SQL> drop audit policy oe_policy;
```

Additional Topics

Exporting data using Oracle Data Pump in Oracle Database Vault environment

1. *Connect to the database as a user who has the DV_OWNER or DV_ADMIN role (for example, dbv_owner):*

```
$ sqlplus dbv_owner
```

2. *Verify that the user piter has the DATAPUMP_EXP_FULL_DATABASE role:*

```
SQL> SELECT GRANTED_ROLE FROM DBA_ROLE_PRIVS  
WHERE  
GRANTED_ROLE LIKE '%FULL%' AND GRANTEE='PITER';
```

GRANTED_ROLE
DATAPUMP_EXP_FULL_DATABASE

Prerequisite role

3. *Authorize the user piter to perform Data Pump operations on HR schema (execute the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure):*

```
SQL> EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER
('PITER', 'HR');
```

PL/SQL procedure successfully completed.

4. *Query the DVSYS.DBA_DV_DATAPUMP_AUTH view to confirm that the user piter is authorized to perform export and import operations only on HR schema:*

```
SQL> column grantee format A10
SQL> column schema format A15
SQL> column object format A15
SQL> SELECT * FROM DVSYS.DBA_DV_DATAPUMP_AUTH
WHERE GRANTEE =      'PITER';
```

GRANTEE	SCHEMA	OBJECT
PITER	HR	%

Authorized for all database object in schema HR

5. *Export the HR.EMPLOYEES and HR.DEPARTMENTS tables:*

```
$ expdp piter DIRECTORY=dp_dir DUMPFILE= exptables.dmp
TABLES=    hr.employees, hr.departments
```

```

Connected to: Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Producti
on
With the Partitioning, Oracle Label Security, OLAP, Advanced Analytics,
Oracle Database Vault, Real Application Testing and Unified Auditing options
Starting "PITER"."SYS_EXPORT_TABLE_01": piter/******** DIRECTORY=dp_dir DUMPFILE=expab
les.dmp TABLES=hr.employees hr.departments
ORA-39327: Oracle Database Vault data is being stored unencrypted in dump file set.
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 128 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/GRANT/OWNER_GRANT/OBJECT_GRANT
Processing object type TABLE_EXPORT/TABLE/COMMENT
Processing object type TABLE_EXPORT/TABLE/INDEX/INDEX
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type TABLE_EXPORT/TABLE/TRIGGER
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
. . exported "HR"."DEPARTMENTS"                      7.125 KB      27 rows
. . exported "HR"."EMPLOYEES"                        17.08 KB     107 rows
Master table "PITER"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
*****

```

The warning message

6. Export HR schema in an unencrypted format:

```
$ expdp piter DIRECTORY=dp_dir DUMPFILE=expsh.dmp
SCHEMAS=hr    ENCRYPTION=NONE
```

You'll receive the same message as in the previous step (ORA-39327), even though you explicitly stated that you don't want to encrypt export. At the end of the job, you'll see that it completed with one error meaning that one warning:

```

Dump file set for PITER.SYS_EXPORT_SCHEMA_01 is:
  /u01/app/oracle/oradata/dp_exp/expsh.dmp
Job "PITER"."SYS_EXPORT_SCHEMA_01" completed with 1 error(s)

```

Successful export with warning

Creating factors in Oracle Database Vault

- 1. Log in to EM12c as a SYSMAN or some other privileged user. Select your database. Then, from Security drop-down menu, choose Database Vault.***



2. *Log in as the dbv_owner user*

A screenshot of a 'Database Login' dialog box. The dialog has a title bar 'Database Login'. Inside, there are fields for 'Username' (containing 'dbv_owner'), 'Password' (containing several redacted dots), and 'Role' (set to 'Normal'). There is also a checkbox for 'Save As' which is unchecked. At the bottom are two buttons: 'Login' and 'Cancel'.

3. *Choose the Administration tab and click on the Factors link*

Oracle Database Vault

Home Page **Administration**

Database Vault Components

Realms

- Command Rules
- Rules
- Rule Sets
- Factors**
- Factor Types
- Secure Application Roles
- OLS Integration
- Database Vault Roles

4. Click on the Create button to create your first custom factor

Oracle Database Vault

Home Page **Administration**

Database Vault Components

Factors

Realms
Command Rules
Rules
Rule Sets
Factors
Factor Types
Secure Application Roles
OLS Integration
Database Vault Roles

Database Vault factor is a context that you define and use in rules that are attached to a rule set which, in turn, can be attached to Realm Authorizations, Command Rules, and Database Vault Secure Application Roles. After you define the factor, the value can be checked using the function DVF.DF\$(factor_name).

Search

Factor Name Go

The search returns all matches beginning with the string you enter. You can use the wildcard symbol (%) in the search string.

View	Create	View	Edit	Delete	Show Oracle defined factors
Factor Name	Factor Type	Evaluation Options	Identified By	Audit Options	Fail Options
no data found					

Columns Hidden 1

5. The name of the factor will be Day, the description will be The name of day, and Factor Type will be Time. After you enter that information, click on the button Next.

Create Factor: General

Enter the general information required to create a factor.

* Name

Description

* Factor Type

6. Enter these configuration details for the factor and click on the button Next. It will appear as shown in this figure :

Create Factor: Configurations

Enter the configuration details for the factor.

* Factor Identification

* Evaluation

* Factor Labeling

Retrieval Method

Validation Method

7. For Audit Options, choose Never. Leave other default values and click the button Next.

8. You won't create new identities at this moment, so just click on the button Next. After you finish reviewing the configuration, click on the Finish button . You should receive a confirmation message and see the newly created factor Day.

Factor Name	Factor Type	Evaluation Options	Identified By	Audit Options	Fail Options	Last Updated Date
Day	Time	By Access	By Method	Never	Show Error Message	
Columns Hidden 1						

9. Click on Link Day (in the column Factor Name). You will see that factor Day will get value SUNDAY. Click on the OK button.

View Factor

Evaluation Results
Evaluated Value SUNDAY
General
Name Day
Description The name of day
Factor Type Time
Configurations
Factor Identification By Method
Evaluation By Access
Factor Labeling By Self
Retrieval Method TO_CHAR(sysdate,'DAY')
Validation Method
Options
Assignment Rule Set
Error Options Show Error Message
Audit Options Never

Now create the new factor NonWorkingDay (**Factor Type:** Time) which will, for the beginning, be based only on the factor Day and test it. After you create the factor Holiday, you'll edit the factor NonWorkingDay in such a way that it is based on both factors (Day and Holiday).

10. *Repeat steps 4 and 5.*

11. *Enter these configuration details for the factor and click on the button Next:*

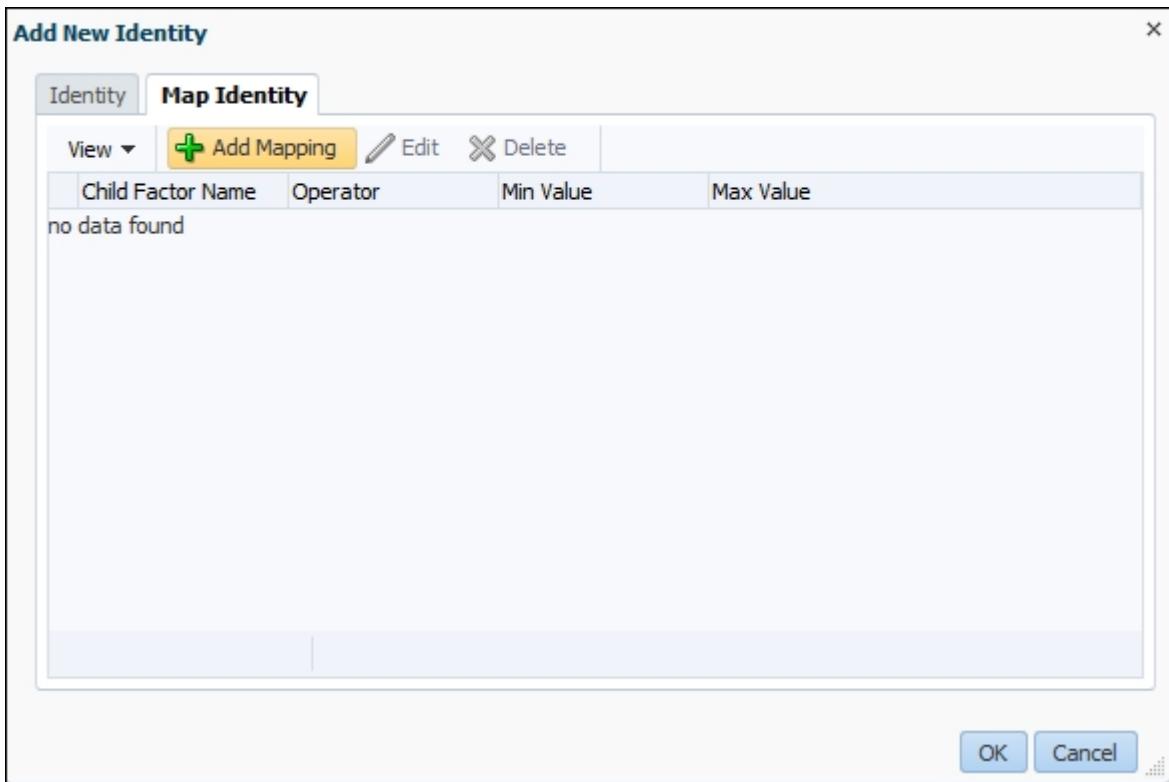
Factor Identification:	By Factors
Evaluation:	By Access
Factor Labeling:	By Self

12. *Leave the default values and click on the Next button.*

13. *Click on the green plus button - Add New Identity*

14. On the tab Identity, enter Value as TRUE and select Untrusted for Trust Level.

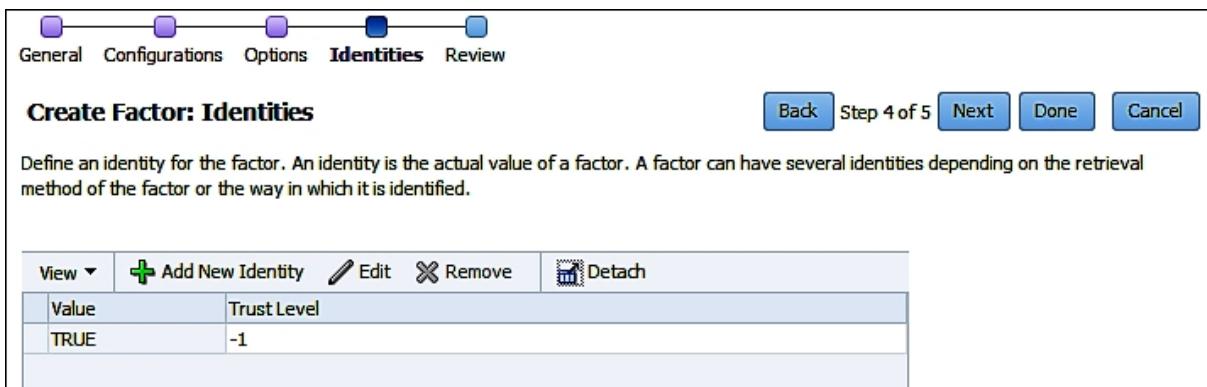
15. Click on the Map Identity tab . Click on the green plus button - Add Mapping



16. Select the following values and click on the OK button:

Child Factor Name	Day
Operator	Like
Min Value	S%

17. You should see that identity is added



18. Add the new identity FALSE. Repeat steps from 13 to 16 with appropriate values (for example, the value FALSE, Trust Level as Somewhat trusted; instead of the Like operator, choose Not Like).

19. Click on the Next button. Review the configuration and click on the Finish button. You should see confirmation message.
20. Click on the link NonWorkingDay (in the column Factor Name). You will see that the factor NonWorkingDay will get value TRUE. Click on the OK button.

View Factor

Evaluation Results						
Evaluated Value TRUE						
General						
Name NonWorkingDay						
Description						
Factor Type Time						
Configurations						
Factor Identification By Factors						
Evaluation By Access						
Factor Labeling By Self						
Retrieval Method						
Validation Method						
Options						
Assignment Rule Set						
Error Options Show Error Message						
Audit Options Always						
Identities						
<table border="1"> <thead> <tr> <th>Value</th> <th>Trust Level</th> </tr> </thead> <tbody> <tr> <td>TRUE</td> <td>-1</td> </tr> <tr> <td>FALSE</td> <td>5</td> </tr> </tbody> </table>	Value	Trust Level	TRUE	-1	FALSE	5
Value	Trust Level					
TRUE	-1					
FALSE	5					

21. On the Factors page, in the table select row in which Day factor is displayed and click on the Edit button (pencil icon). Click on the Next button.
22. Change Retrieval Method to `RTRIM(TO_CHAR(sysdate, 'DAY'))` and click on the Done button.
23. Create the new factor Holiday (Factor Type: Time).
24. Enter these configuration details for the factor and click on the Done button:

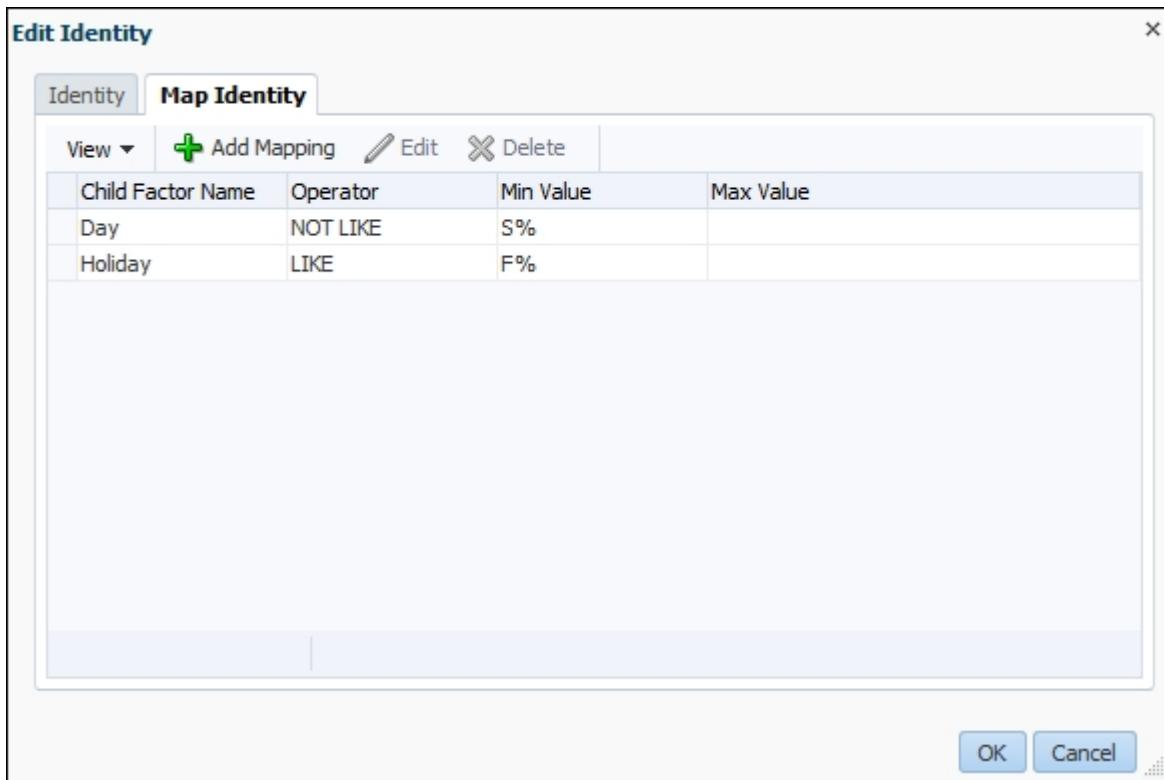
Factor Identification:	By Method
Evaluation:	By Access
Factor Labeling:	By Self
Retrieval Method:	PITER.GET_HOLIDAY

It will appear as shown in this figure :

25. Edit the factor *NonWorkingDay* so that it has three identities (*NO*, *WEEKEND*, and *COMPANY_HOLIDAY*) and click on **OK.**

26. Perform the following for the three identities:

26 a. Edit the FALSE identity (change value to **NO**, add mapping in the **Map Identity - Child Factor Name**: Holiday, **Operator**: Like, **Min Value**: F%). Click on the **OK** button.



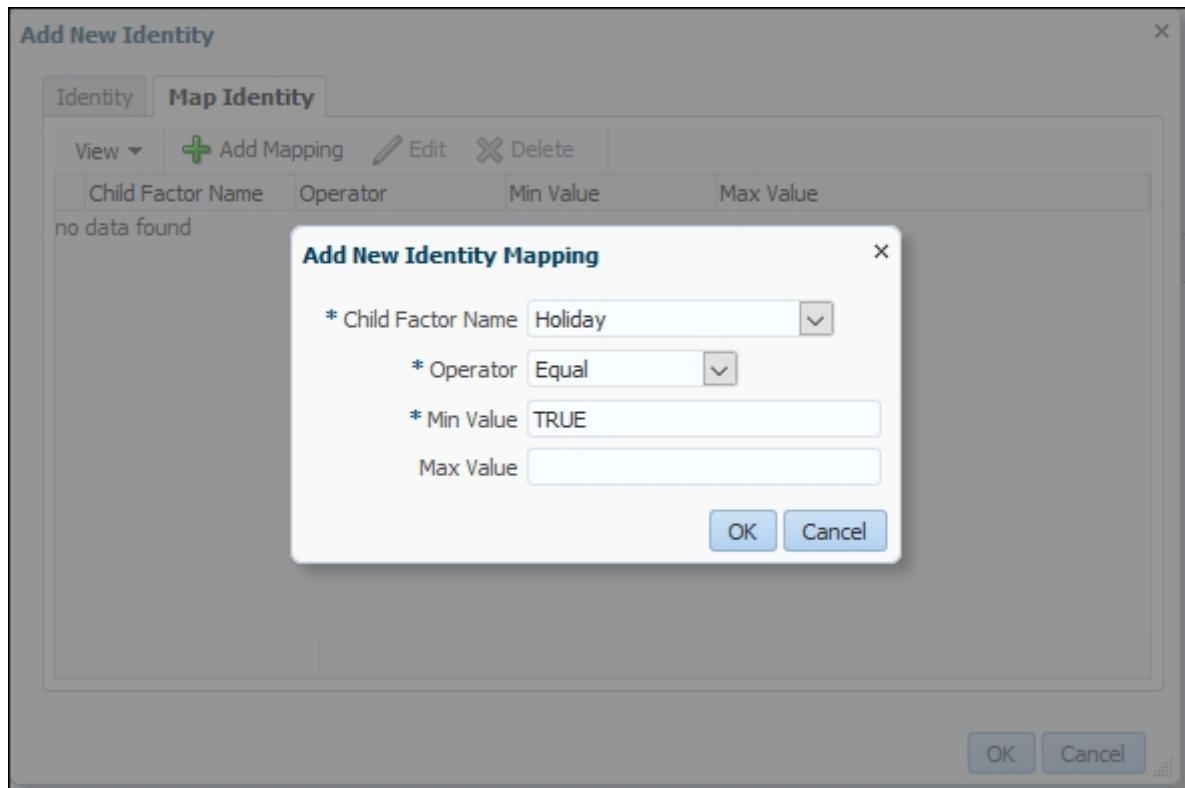
26 b. Edit the TRUE identity and click on the **OK** button (change value to WEEKEND, change mapping to have two rows):

Child Factor Name	Operator	Min Value	Max Value
Day	Equal	SATURDAY	
Day	Equal	SUNDAY	

26 c. Add the new COMPANY_HOLIDAY identity (**Trust Level: Untrusted**). On the **Map Identity** tab, click on **Add Mapping**. Set the following values and click on **OK**:

Child Factor Name	Operator	Min Value	Max Value
Holiday	Equal	TRUE	

This will appear as shown in this figure:



27. View evaluated value for factor Day (repeat step 9).

The screenshot shows the 'View Factor' dialog box. It has sections for 'Evaluation Results' (highlighted with a blue border) and 'General' (also highlighted with a blue border). In the 'General' section, 'Name Day' is listed. Below it, 'Description' is described as 'The name of day' and 'Factor Type' is 'Time'. The 'Configurations' section contains several configuration items: 'Factor Identification By Method', 'Evaluation By Access', 'Factor Labeling By Self', 'Retrieval Method RTRIM(TO_CHAR(sysdate,'DAY'))', and 'Validation Method'.

28. View evaluated value for factor Holiday. The result is shown next.

View Factor	
Evaluation Results	
Evaluated Value TRUE	
General	
Name	Holiday
Description	
Factor Type	Time
Configurations	
Factor Identification	By Method
Evaluation	By Access
Factor Labeling	By Self
Retrieval Method	PITER.GET_HOLIDAY
Validation Method	

29. *View an evaluated value for factor NonWorkingDay. The result is shown next:*

View Factor	
Evaluation Results	
Evaluated Value COMPANY_HOLIDAY	
General	
Name	NonWorkingDay
Description	
Factor Type	Time
Configurations	
Factor Identification	By Factors
Evaluation	By Access
Factor Labeling	By Self
Retrieval Method	
Validation Method	

Using TDE in a multitenant environment

1. *Enter the following text into your sqlnet.ora file located in a network/admin directory of your oracle home (for example, /u01/app/oracle/product/12.1.0/dbhome_1)*

ENCRYPTION_WALLET_LOCATION=
(SOURCE=

(METHOD=FILE)
(METHOD_DATA=
(DIRECTORY=/u01/app/oracle/admin/\$ORACLE_SID/wallet)))

2. *Change your environment to the first container database (for example, cdb1):*

```
[oracle@host01 ~]$ . oraenv  
ORACLE_SID = [oracle] ? cdb1
```

3. *Connect as a user with the DBA role (for example, system), create a new user (for example, c##tdedba) to manage key management administration, and grant him appropriate privileges:*

```
$ sqlplus system
```

```
SQL> create user c##tdedba identified by oracle123 container=all;
```

```
SQL> grant administer key management to c##tdedba container=all;
```

```
SQL> grant create session to c##tdedba container=all;
```

```
SQL> grant select any dictionary to c##tdedba container=all;
```

```
SQL> grant set container to c##tdedba container=all;
```

4. *Connect as a user c##tdedba and create a keystore:*

```
SQL> connect c##tdedba/oracle123  
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE  
'/u01/app/oracle/admin/cdb1/wallet' identified by oracle1;
```

5. *See information about the previously created keystore and open it:*

```
SQL> select wallet_type,wrl_type,status from v$encryption_wallet;
```

WALLET_TYPE	WRL_TYPE	STATUS
UNKNOWN	FILE	CLOSED

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN  
IDENTIFIED BY oracle1; SQL> select wallet_type,wrl_type,status  
from v$encryption_wallet;
```

```
WALLET_TYPE    WRL_TYPE    STATUS -----  
----- PASSWORD      FILE      OPEN_NO_MASTER_KEY
```

SQL> select con_id, tag, key_id from v\$encryption_keys;

no rows selected

6. *Create a new master key for root container:*

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY USING TAG  
'description:    root key' IDENTIFIED BY oracle1 WITH BACKUP;
```

SQL> select con_id, tag, key_id from v\$encryption_keys;

CON_ID	TAG	KEY_ID
0	description: root key	AQInxR2++E8yv2hvZVrc5aQAAAAAAAAAAAAAAAAAAAAAAA

SQL> select wallet_type,wrl_type,status from v\$encryption_wallet;

```
WALLET_TYPE    WRL_TYPE    STATUS -----  
----- PASSWORD      FILE      OPEN
```

7. *Connect to a pluggable database (for example, pdb11) inside the first container database and check availability of a keystore:*

SQL> alter session set container=pdb11;

SQL> select wallet_type,wrl_type,status from v\$encryption_wallet;

```
WALLET_TYPE    WRL_TYPE    STATUS -----  
----- UNKNOWN      FILE      CLOSED
```

8. *Open a keystore, check availability of a master key, and create one:*

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN  
IDENTIFIED BY oracle1;
```

SQL> select wallet_type,wrl_type,status from v\$encryption_wallet;

```
WALLET_TYPE    WRL_TYPE    STATUS -----  
-----
```

----- **PASSWORD FILE OPEN_NO_MASTER_KEY**

SQL> select con_id, tag, key_id from v\$encryption_keys; no rows selected

**SQL> ADMINISTER KEY MANAGEMENT SET KEY USING TAG
'description: pdb11 key' IDENTIFIED BY oracle1 WITH BACKUP;**

SQL> select con_id, tag, key_id from v\$encryption_keys;

CON_ID	TAG	KEY_ID
0	description: pdb11 key	AeC4mqH5WU+mvxjEMBNk7lcAAAAAAAAAAAAAAAAAAAAA

SQL> select wallet_type,wrl_type,status from v\$encryption_wallet;

WALLET_TYPE	WRL_TYPE	STATUS
PASSWORD	FILE	OPEN

9. *Change environment for the second container database (for example, cdb2):*

[oracle@host01 ~]\$. oraenv
ORACLE_SID = [cdb1] ? cdb2

10. *Connect as a user with the sysdba privileges, create a new user (for example, c##tdedba), and grant him appropriate privileges:*

\$ sqlplus / as sysdba

SQL> create user c##tdedba identified by oracle321 container=all;
SQL> grant syskm to c##tdedba container=all;

11. *Connect as a user c##tdedba (as syskm), create a keystore, and open it for all pluggable databases:*

**SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE
'/u01/app/oracle/admin/cdb2/wallet' identified by oracle2;**

SQL> select wallet_type,wrl_type,status from v\$encryption_wallet;

WALLET_TYPE	WRL_TYPE	STATUS
UNKNOWN	FILE	CLOSED

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN  
IDENTIFIED BY      oracle2 container=all;
```

```
SQL> select wallet_type,wrl_type,status from v$encryption_wallet;
```

```
WALLET_TYPE    WRL_TYPE    STATUS -----  
----- PASSWORD     FILE      OPEN_NO_MASTER_KEY
```

12. *Create new master keys for all pdbs:*

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY USING TAG  
'description: all pdbs' IDENTIFIED BY oracle2 WITH BACKUP  
container=all;
```

```
SQL> select con_id, tag, key_id from v$encryption_keys;
```

CON_ID	TAG	KEY_ID
0	description: all pdbs	AZ07ljU0QU8cv/wRuUgzoBEAAAAAAAAAAAAAAAAAAAAAAA
0	description: all pdbs	AbwHF8tkj0+ov9/HG430YiUAAAAAAAAAAAAAAAAAAAAAAA

13. *Connect to a pluggable database as a SYS user and check keystore and masterkey:*

```
SQL> connect / as sysdba
```

```
SQL> alter session set container=pdb21;
```

```
SQL> select wallet_type,wrl_type,status from v$encryption_wallet;
```

```
WALLET_TYPE    WRL_TYPE    STATUS -----  
----- PASSWORD     FILE      OPEN
```

```
SQL> select con_id, tag, key_id from v$encryption_keys;
```

CON_ID	TAG	KEY_ID
0	description: all pdbs	AZ07ljU0QU8cv/wRuUgzoBEAAAAAAAAAAAAAAA

14. *Change your environment to the first container database (for example, cdb1):*

```
[oracle@host01 ~]$ . oraenv ORACLE_SID = [cdb2] ? cdb1
```

15. Connect to the pluggable database as a user who has the DBA role (for example, c##ernesto), create a test table with one encrypted column, and insert some data:

```
$ sqlplus c##erneesto@pdb11
SQL> create table hr.enc_tbl(a int, b varchar2(20) encrypt);
SQL> insert into hr.enc_tbl values (1, 'value1');
SQL> insert into hr.enc_tbl values (2, 'value2');
SQL> commit;
SQL> select * from hr.enc_tbl;
```

A	B	-----	1	value1	2	value2
---	---	-------	---	--------	---	--------

16. Export a master key:

```
SQL> ADMINISTER KEY MANAGEMENT EXPORT KEYS WITH
SECRET "secret1"      to '/home/oracle/keys.exp' IDENTIFIED BY
oracle1;
```

17. Close the pluggable database pdb11 and unplug it:

```
SQL> alter pluggable database pdb11 close immediate;
SQL> alter pluggable database pdb11 unplug into
'/home/oracle/pdb11.xml';
SQL> drop pluggable database pdb11 keep datafiles;
```

18. Change your environment to the second container database (for example, cdb2):

```
[oracle@host01 ~]$ . oraenv ORACLE_SID = [cdb1] ? cdb2
```

19. Connect to the second container database (for example, cdb2) as a sys user and plug the previously unplugged database (pdb11):

```
$ sqlplus / as sysdba
SQL> create pluggable database pdb11 using '/home/oracle/pdb11.xml';
```

20. Open the pluggable database:

```
SQL> alter pluggable database pdb11 open;
```

Warning: PDB altered with errors.

```
SQL> show pdbs
  CON_ID CON_NAME      OPEN MODE  RESTRICTED -----
----- 2 PDB$SEED    READ ONLY NO
3 PDB21      READ WRITE NO      4 PDB11      READ WRITE
YES
```

21. Connect to pdb11, as a SYS user, open the keystore, and try to select from table with encrypted column:

```
SQL> alter session set container=pdb11;
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
IDENTIFIED BY oracle2;
SQL> select * from hr.enc_tbl;
select * from hr.enc_tbl
*
```

ERROR at line 1:

ORA-28362: master key not found

22. Import the master key for this pluggable database and restart it:

```
SQL> ADMINISTER KEY MANAGEMENT IMPORT ENCRYPTION
KEYS WITH SECRET "secret1" FROM '/home/oracle/keys.exp'
IDENTIFIED BY oracle2 WITH BACKUP; SQL> alter pluggable
database pdb11 close immediate;
```

```
SQL> alter pluggable database pdb11 open;
```

```
SQL> show pdbs
  CON_ID CON_NAME      OPEN MODE  RESTRICTED -----
----- 2 PDB$SEED    READ ONLY NO
3 PDB21      READ WRITE NO      4 PDB11      READ WRITE
NO
```

23. Connect to the pluggable database (pdb11), open the keystore, and select from table with encrypted column:

```
SQL> alter session set container=pdb11;
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
IDENTIFIED BY oracle2;
```

```
SQL> select * from hr.enc_tbl;
A      B ----- 1      value1 2      value2

```

Appendix – Application Contexts

Exploring and using built-in contexts

1. *Connect to the database as a user who has appropriate privileges (for example, user maja):*

\$ sqlplus maja

2. *Find the name of host machine from which the client has connected to the database.*

```
SQL> select sys_context('USERENV','HOST') from dual;
SYS_CONTEXT('USERENV','HOST')
-----
dbhost.orapassion.com
```

The name of the client host machine

3. *Find the name of the user who logged on to the database.*

```
SQL> select sys_context('USERENV','SESSION_USER') from dual;
SYS_CONTEXT('USERENV','SESSION_USER')
-----
MAJA
```

The name of the session user

4. *Find the name of the program used for the database session.*

```
SQL> SELECT sys_context('USERENV', 'CLIENT_PROGRAM_NAME') FROM dual;
SYS_CONTEXT('USERENV','CLIENT_PROGRAM_NAME')
-----
sqlplus@dbhost.orapassion.com (TNS V1-V3)
```

The name of the client program

5. Find unified audit session ID.

```
SQL> select sys_context ('USERENV','UNIFIED_AUDIT_SESSIONID') from dual;  
SYS_CONTEXT('USERENV','UNIFIED_AUDIT_SESSIONID')  
-----  
2303811715
```

A unified audit session ID

Creating an application context

1. *Connect to the database as a user who has appropriate privileges (for example, user maja).*

\$ sqlplus maja

2. *Create a local application context (for example, sh_client).*

Note

The PL/SQL package that will be used to set application context attributes doesn't have to exist at this time, but you have to specify its name.

**SQL> CREATE CONTEXT <context_name> USING
<PL/SQL_package_name>;**

```
SQL> CREATE CONTEXT sh_client USING sh_ctx_pkg;  
Context created.
```

Creating an application context

Setting application context attributes

1. *Connect to the database as a user who has appropriate privileges (for example, user maja):*

\$ sqlplus maja

2. *Create the PL/SQL package that will set the cust_id attribute with the value, which is equal to the value of the cust_id column when the following statement is evaluated: UPPER(cust_email) = (SYS_CONTEXT('USERENV', 'SESSION_USER') || '@COMPANY.EXAMPLE.COM'). In case session user is not a customer, set the value for cust_id attribute in the application context to 0.*

```
SQL> CREATE OR REPLACE PACKAGE sh_ctx_pkg IS
  2  PROCEDURE set_cust_id;
  3 END;
 4 /
Package created.

SQL> CREATE OR REPLACE PACKAGE BODY sh_ctx_pkg IS
  2  PROCEDURE set_cust_id
  3  IS
  4    v_cust_id NUMBER;
  5  BEGIN
  6    SELECT cust_id INTO v_cust_id FROM sh.customers
  7      WHERE UPPER(cust_email) = (SYS_CONTEXT('USERENV','SESSION_USER') || '@COMPANY.EXAMPLE.COM');
  8    DBMS_SESSION.SET_CONTEXT('sh_client','cust_id',v_cust_id);
  9  EXCEPTION
 10    WHEN no_data_found THEN
 11      DBMS_SESSION.SET_CONTEXT('sh_client','cust_id',0);
 12  END;
 13 END;
 14 /
Package body created.
```

Creating a PL/SQL package

3. *Create a logon trigger that calls the sh_ctx_pkg.set_cust_id procedure.*

```
SQL> CREATE OR REPLACE TRIGGER sh_ctx_logon
  2  AFTER LOGON ON DATABASE
  3  BEGIN
  4    sh_ctx_pkg.set_cust_id();
  5  END;
 6 /
Trigger created.
```

A logon trigger

Using an application context

1. *Connect to the database as a newly created user (for example, user sofia):*

\$ sqlplus sofia

2. *Verify that the user (for example, sofia) can access all data in the sh.customers table.*

```
SQL> SELECT COUNT(*) FROM SH.CUSTOMERS;  
  
COUNT(*)  
-----  
55501
```

The entire data in sh.customers

3. *Verify that when executing the following statement, he or she (for example, sofia) can view only his or her data.*

```
SQL> SELECT COUNT(*) FROM SH.CUSTOMERS  
  2 WHERE cust_id = sys_context('sh_client','cust_id');  
  
COUNT(*)  
-----  
1
```

Only data about newly created user