Oracle Database 12*c*: Security

Activity Guide – Volume II

D81599GC10

Edition 1.0

June 2014

D84092

Author

Dominique Jeunot

Technical Contributors and Reviewers

Jean-François Verrier, Donna Keesling, James Spiller, Gerlinde Frenzen, Pat Huey, Veerabhadra Rao Putrevu, Joel Goodman

This book was published using: Oracle Tutor

Table of Contents

Practices for Lesson 13: Implementing Oracle Label Security Policies

Chapter 13

Practice 13-1: Registering and Enabling Oracle Label Security

Overview

In this practice, you register and enable Oracle Label Security (OLS) in the `pdb1_1` pluggable database of `cdb1` by using manual procedures. Then you will register and enable OLS in `orcl` using DBCA.

Tasks

1. Connect to the `pdb1_1` pluggable database and check whether OLS is registered. If it is registered, check if it is enabled.
   a. Connect to the `pdb1_1` pluggable database as `SYSDBA`.

```
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

   b. Check whether OLS is registered.

```
$ sqlplus sys@pdb1_1 as sysdba

Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL> SELECT status FROM DBA_OLS_STATUS
     WHERE  name = 'OLS_CONFIGURE_STATUS';
  2
STATU
-----
FALSE

SQL>
```

   c. Register OLS.

```
SQL> EXEC LBACSYS.CONFIGURE_OLS

PL/SQL procedure successfully completed.

SQL> SELECT status FROM DBA_OLS_STATUS
     WHERE  name = 'OLS_CONFIGURE_STATUS';
  2
```

```
STATU
-----
TRUE

SQL>
```

d. Check whether OLS is enabled.

```
SQL> SELECT value FROM V$OPTION
     WHERE  parameter = 'Oracle Label Security';
  2
VALUE
-----
FALSE

SQL>
```

e. Enable OLS.

```
SQL> EXEC LBACSYS.OLS_ENFORCEMENT.ENABLE_OLS

PL/SQL procedure successfully completed.

SQL> SELECT value FROM V$OPTION
     WHERE  parameter = 'Oracle Label Security';
  2
VALUE
----------------------------------------------------------------
TRUE

SQL> EXIT
$
```

Notice that you can register and enable OLS at the PDB level.

2. Connect to the `pdb1_2` pluggable database and check whether OLS is registered.

a. Connect to the `pdb1_2` pluggable database as `SYSDBA`.

```
$ sqlplus sys@pdb1_2 as sysdba

Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
SQL>
```

b. Check whether OLS is registered.

```
SQL> SELECT status FROM DBA_OLS_STATUS
     WHERE  name = 'OLS_CONFIGURE_STATUS';
  2
STATU
-----
FALSE

SQL>
```

c. Check whether OLS is enabled.

```
SQL> SELECT value FROM V$OPTION
     WHERE  parameter = 'Oracle Label Security';
  2
VALUE
-----
FALSE

SQL>
```

3. What happens if you register and enable OLS at the root level? Does it cascade to all PDBs?

a. Connect to `root` as `SYSDBA`.

```
SQL> conn / as sysdba
Connected.
SQL> SELECT status FROM DBA_OLS_STATUS
     WHERE  name = 'OLS_CONFIGURE_STATUS';
  2
STATU
-----
FALSE

SQL> SELECT value FROM V$OPTION
     WHERE  parameter = 'Oracle Label Security';
  2
VALUE
----------------------------------------------------------------
FALSE

SQL>
```

b. Register OLS.

```
SQL> EXEC LBACSYS.CONFIGURE_OLS

PL/SQL procedure successfully completed.
```

```
SQL> SELECT status FROM DBA_OLS_STATUS
     WHERE  name = 'OLS_CONFIGURE_STATUS';
  2
STATU
-----
TRUE

SQL>
```

c. Enable OLS.

```
SQL> EXEC LBACSYS.OLS_ENFORCEMENT.ENABLE_OLS

PL/SQL procedure successfully completed.

SQL> SELECT value FROM V$OPTION
     WHERE  parameter = 'Oracle Label Security';
  2
VALUE
----------------------------------------------------------------
TRUE

SQL>
```

d. Check whether the operations cascaded to all PDBs.

```
SQL> CONNECT sys@pdb1_2 as sysdba
Enter password: ******
Connected.
SQL> SELECT status FROM DBA_OLS_STATUS
     WHERE  name = 'OLS_CONFIGURE_STATUS';
  2
STATU
-----
FALSE

SQL> SELECT value FROM V$OPTION
     WHERE  parameter = 'Oracle Label Security';
  2
VALUE
----------------------------------------------------------------
FALSE

SQL> EXIT
$
```

It did not cascade to all PDBs. Registration and enabling execute at the container level.

4. Connect to the `orcl` database and check whether OLS is registered. If it is registered, check whether it is enabled.

```
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

a. Check whether OLS is registered.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> SELECT status FROM DBA_OLS_STATUS
     WHERE  name = 'OLS_CONFIGURE_STATUS';
  2
STATU
-----
TRUE

SQL> EXIT
$
```

Notice that the banner displays “Oracle Label Security.” The option has already been enabled because in a previous practice, Database Vault was enabled to protect `HR` objects. The configuration of Database Vault automatically enables OLS. OLS is a required option for Oracle Database Vault.

b. If OLS is not enabled because you did not execute practice 3-7, use DBCA to register and enable OLS only.

1) Start `dbca` and perform the following steps.

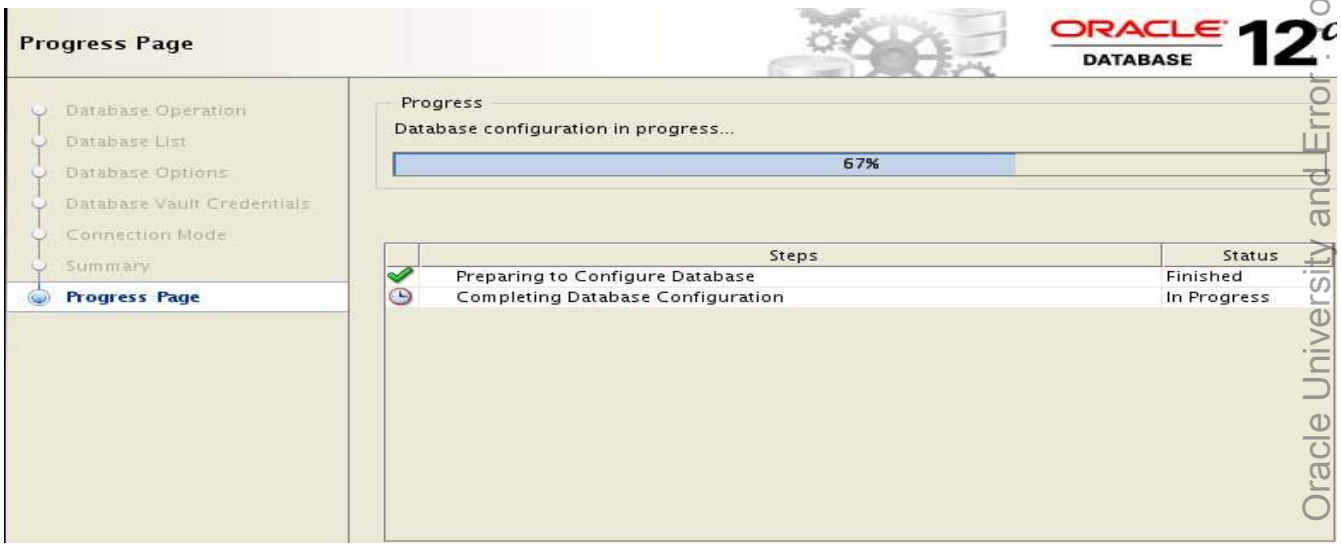
```
$ dbca
```

Step	Window/Page Description	Choices or Values
a.	Step 1: Database Operation	Select “Configure Database Options.” Click Next.
b.	Step 2: Database List	Select “**orcl**”. Click Next.
c.	Step 3: Database Options	Click Next.
d.	Step 4: Database Vault Credentials	Deselect “Configure Database Vault” if selected.

Step	Window/Page Description	Choices or Values
		Select “Configure Label Security” if not selected. Click Next.
e.	Step 5: Connection Mode	Click Next.
f.	Step 6: Summary	Click Finish.
g.	Step 7: Progress Page	On the Database Configuration Assistant page, click OK. Click Close.

The following screenshot shows step 6.

The following screenshot shows step 7.



c. Check whether OLS is registered and enabled.

```
$ sqlplus / as sysdba


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
```

```
SQL> SELECT value FROM V$OPTION
     WHERE  parameter = 'Oracle Label Security';
  2
VALUE
-----
TRUE

SQL> SELECT status FROM DBA_OLS_STATUS
     WHERE  name = 'OLS_CONFIGURE_STATUS';
  2
STATU
-----
TRUE

SQL> EXIT
$
```

Notice that the banner now displays “Oracle Label Security.”

Practice 13-2: Implementing Oracle Label Security

Overview

In this practice, you implement a simple label security system in a pluggable database.

Scenario

This practice uses the `HR.LOCATIONS` and `HR.JOB_HISTORY` sample schema tables in the `pdb1_1` pluggable database. Oracle Label Security assigns sensitivity labels to data rows in the `LOCATIONS` and `JOB_HISTORY` tables. The data has been analyzed and can be placed in three sensitivity levels. There are four groups: one for each region and a `GLOBAL` group. The three locations in the Asia region are assigned the `SENSITIVE::ASIA` sensitivity label. One location in the United States region is assigned the `HIGHLY_SENSITIVE::UNITED_STATES` sensitivity label. All remaining locations are assigned the `PUBLIC` sensitivity label. From this analysis, the components and labels are displayed in the following table:

Levels for the `FACILITY` policy

Short Name	Long Name	Numeric
P	PUBLIC	1000
S	SENSITIVE	2000
HS	HIGHLY_SENSITIVE	3000

Groups for the `FACILITY` policy

Short Name	Long Name	Numeric
US	United States	101
EU	Europe	102
ASIA	Asia	103
GLOBAL	Global	1000

Active data labels for `FACILITY`

Label	Tag Number
P	1000
S::US	2101
S::ASIA	2103
HS::US	3101
HS::ASIA	3103

Levels for the `PRIVACY` policy

Short Name	Long Name	Numeric

C	CONFIDENTIAL	1000
S	SENSITIVE	2000

Active data labels for PRIVACY

Label	Tag Number
C	101000
S	102000

Data rows in the JOB_HISTORY table with END_DATE greater than seven years are assigned the SENSITIVE sensitivity label. Data rows with END_DATE less than or equal to five years are assigned the CONFIDENTIAL sensitivity label.

The HR application owner is authorized to read and write all data rows in both the JOB_HISTORY and LOCATIONS tables.

The MYCO_MGR application user is authorized to view all data in the LOCATIONS table labeled SENSITIVE and below, and having the US, ASIA, or EUROPE groups. The MYCO_PLANNING application user is authorized to view all data in the LOCATIONS table labeled HIGHLY SENSITIVE and below, and having the GLOBAL group. Note that the ASIA, EUROPE, and US groups are created as subordinate to the GLOBAL group. MYCO_EMP is allowed access only to the data labeled PUBLIC.

Two Oracle Label Security policies are created:

- **FACILITY:** The designated security column is FACLAB.
- **PRIVACY:** The designated security column is PRIVLAB.

The security columns for both columns are marked HIDDEN at policy-creation time.

For this practice, you must log in as the oracle user. All scripts are found in the $HOME/labs/OLS directory. In this practice, it is assumed that the sessions are connected using the database environment variable.

1. Create three users: MYCO_EMP, MYCO_MGR, and MYCO_PLANNING. You also grant them access to the JOB_HISTORY and LOCATIONS tables in the HR schema. Open a terminal window. Set the database environment variables. Change the directory to /home/oracle/labs/OLS. In the SQL*Plus session, execute the create_OLS_users.sql script.

```
$ cd /home/oracle/labs/OLS
$ sqlplus system@pdb1_1

Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL>
SQL> @create_OLS_users.sql
```

```
SQL> ALTER USER hr IDENTIFIED BY oracle_4U ACCOUNT UNLOCK;

User altered.

SQL>
SQL> -- Cleanup from previous runs
SQL>
SQL> DROP USER myco_EMP;

User dropped.

SQL> DROP USER myco_MGR;

User dropped.

SQL> DROP USER myco_PLANNING;

User dropped.

SQL>
SQL> --
*****************************************************************
*******
SQL> -- Create Users MYCO_EMP
SQL> -- Create Users MYCO_MGR
SQL> -- Create Users MYCO_PLANNING
SQL> --
*****************************************************************
*******
SQL>
SQL> GRANT CREATE SESSION to MYCO_EMP IDENTIFIED BY oracle_4U;

Grant succeeded.

SQL> GRANT CREATE SESSION to MYCO_MGR IDENTIFIED BY oracle_4U;

Grant succeeded.

SQL> GRANT CREATE SESSION to MYCO_PLANNING IDENTIFIED BY
oracle_4U;

Grant succeeded.
```

```
SQL>
SQL>
SQL> --
*****************************************************************
*******
SQL> -- Connect as User HR and grant select on job_history to
SQL> -- MYCO_MGR, MYCO_EMP and MYCO_PLANNING
SQL> --
SQL> -- Grant select on locations to MYCO_EMP and MYCO_MGR.
SQL> -- Grant select, insert, update, delete on locations to
MYCO_PLANNING
SQL> --
SQL> -- Note - A database role could be used here in place of
direct grants
SQL> --
*****************************************************************
*******
SQL>
SQL> CONNECT HR/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
SQL> GRANT SELECT ON JOB_HISTORY TO MYCO_EMP;

Grant succeeded.

SQL> GRANT SELECT ON JOB_HISTORY TO MYCO_MGR;

Grant succeeded.

SQL> GRANT SELECT ON JOB_HISTORY TO MYCO_PLANNING;

Grant succeeded.

SQL>
SQL> GRANT SELECT ON LOCATIONS TO MYCO_EMP;

Grant succeeded.

SQL> GRANT SELECT ON LOCATIONS TO MYCO_MGR;

Grant succeeded.

SQL> GRANT SELECT, INSERT, UPDATE, DELETE ON LOCATIONS TO
MYCO_PLANNING;
```

```
Grant succeeded.

SQL>
```

2. At this point, a policy must be created to hold the label information. Only a user with proper privileges can create policies. The only user with those privileges is `LBACSYS`. The `LBACSYS` account is locked by the DBCA by default. You can unlock the `LBACSYS` account for these practices. If the `LBACSYS` account is locked, it can be unlocked with the following command:

```
ALTER USER lbacsys IDENTIFIED BY oracle_4U ACCOUNT UNLOCK;
```

```
SQL> connect system@pdb1_1
Enter password: ******
Connected.
SQL> ALTER USER lbacsys IDENTIFIED BY oracle_4U ACCOUNT UNLOCK;
ALTER USER lbacsys IDENTIFIED BY oracle_4U ACCOUNT UNLOCK
*
ERROR at line 1:
ORA-65066: The specified changes must apply to all containers

SQL>
```

The `LBACSYS` account is a common user. Any change must be applied to all containers. Connect to the `root` to be able to perform the operation.

```
SQL> connect system@cdb1
Enter password: ******
Connected.
SQL> ALTER USER lbacsys IDENTIFIED BY oracle_4U ACCOUNT UNLOCK
     CONTAINER=ALL;
  2
User altered.

SQL>
```

3. The first step in setting up OLS is to create policies. Create the `FACILITY` policy in `pdb1_1`. Then you create the data labels. In this case, you create three sensitivity levels and four groups (see the specification in the scenario). Use SQL*Plus and execute the `create_labels.sql` script.

```
SQL> connect lbacsys@pdb1_1
Enter password: ******
Connected.
SQL> @$HOME/labs/OLS/create_labels.sql
SQL> set echo on
SQL> -- ****************************************************
SQL> -- Connected as User LBACSYS in the PDB
SQL> -- ****************************************************
```

```
SQL> -- Dropping FACILITY and PRIVACY policies
SQL> --  in case they exist
SQL> -- ****************************************************
SQL> EXECUTE LBACSYS.SA_SYSDBA.DROP_POLICY('FACILITY',TRUE);
BEGIN LBACSYS.SA_SYSDBA.DROP_POLICY('FACILITY',TRUE); END;

*
ERROR at line 1:
ORA-12416: policy FACILITY not found
ORA-06512: at "LBACSYS.SA_SYSDBA", line 148
ORA-06512: at line 1


SQL> EXECUTE SA_SYSDBA.DROP_POLICY('PRIVACY',TRUE);
BEGIN SA_SYSDBA.DROP_POLICY('PRIVACY',TRUE); END;

*
ERROR at line 1:
ORA-12416: policy PRIVACY not found
ORA-06512: at "LBACSYS.SA_SYSDBA", line 148
ORA-06512: at line 1


SQL> --
SQL> -- ****************************************************
SQL> -- Creating FACILITY Policy
SQL> -- ****************************************************
SQL> BEGIN
  2   SA_SYSDBA.CREATE_POLICY('FACILITY','FACLAB',
  3            'READ_CONTROL,CHECK_CONTROL,LABEL_DEFAULT,HIDE');
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL> --
SQL> -- ****************************************************
SQL> -- Adding sensitivity levels to FACILITY policy:
SQL> -- ****************************************************
SQL> BEGIN
  2    SA_COMPONENTS.CREATE_LEVEL('FACILITY',
  3                        1000,'P','PUBLIC');
```

```
  4    SA_COMPONENTS.CREATE_LEVEL('FACILITY',
  5                       2000,'S','SENSITIVE');
  6    SA_COMPONENTS.CREATE_LEVEL('FACILITY',
  7                       3000,'HS','HIGHLY_SENSITIVE');
  8  END;
  9  /

PL/SQL procedure successfully completed.

SQL> --
SQL> -- ****************************************************
SQL> -- Adding groups to FACILITY policy:
SQL> -- ****************************************************
SQL> BEGIN
  2    SA_COMPONENTS.CREATE_GROUP('FACILITY',
  3                       1000,'Global','Global');
  4    SA_COMPONENTS.CREATE_GROUP('FACILITY',
  5                  101,'US','United States','GLOBAL');
  6    SA_COMPONENTS.CREATE_GROUP('FACILITY',
  7                  102,'EU','Europe','GLOBAL');
  8    SA_COMPONENTS.CREATE_GROUP('FACILITY',
  9                  103,'Asia','Asia','GLOBAL');
 10  END;
 11  /

PL/SQL procedure successfully completed.

SQL> --
SQL> -- ****************************************************
SQL> -- Creating Labels for FACILITY policy
SQL> -- ****************************************************
SQL> EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',-
>                      1000,'P');

PL/SQL procedure successfully completed.

SQL> EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',-
>                      2101,'S::US');

PL/SQL procedure successfully completed.

SQL> EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',-
```

```
>                       3101,'HS::US');

PL/SQL procedure successfully completed.

SQL> EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',-
>                       2103,'S::ASIA');

PL/SQL procedure successfully completed.

SQL> EXECUTE SA_LABEL_ADMIN.CREATE_LABEL('FACILITY',-
>                       3103,'HS::ASIA');

PL/SQL procedure successfully completed.

SQL>
```

4. Set up the `LBACSYS` user to use the Enterprise Manager Cloud Control (EM CC) for the target.
   a. If the `cdb1` target is not yet configured as a managed target in EM CC, proceed first with the steps described in Practice 3-2, step 9. To unlock the `DBSNMP` user, connect to the root as `SYSDBA` and use the following statement:

      ```
      alter user dbsnmp identified by oracle_4U account unlock
      container=all;
      ```

   b. Then log in as `LBACSYS` as follows:

Page Description	Choices or Value
Step 1: Enterprise Summary	Click “Targets” and then click “Databases.”
Step 2: Databases	Select “`cdb1`” and click the “`cdb1`” link. If the `cdb1` link does not appear, refresh.
Step 3: Pluggable Databases (bottom left)	Click the “`cdb1_PDB1_1`” link.
Step 4: `cdb1` / `PDB1_1`	Click “Administration”, click “Security”, and then click “Oracle Label Security.”
Step 5: Database Login	Database: `PDB1_1` is displayed. Username: `LBACSYS` Password: `oracle_4U` Select: Save As “`lbacsys_cred`” Click **Login**.

   You get an error indicating that the application requires more database privileges than you have currently been granted.

   c. Grant the `SELECT ANY DICTIONARY` system privilege to `LBACSYS`.

```
SQL> connect system@pdb1_1
```

```
Enter password: ******
Connected.
SQL> GRANT select any dictionary TO lbacsys;

Grant succeeded.

SQL>
```

d. Retry the operation that failed in the previous step.

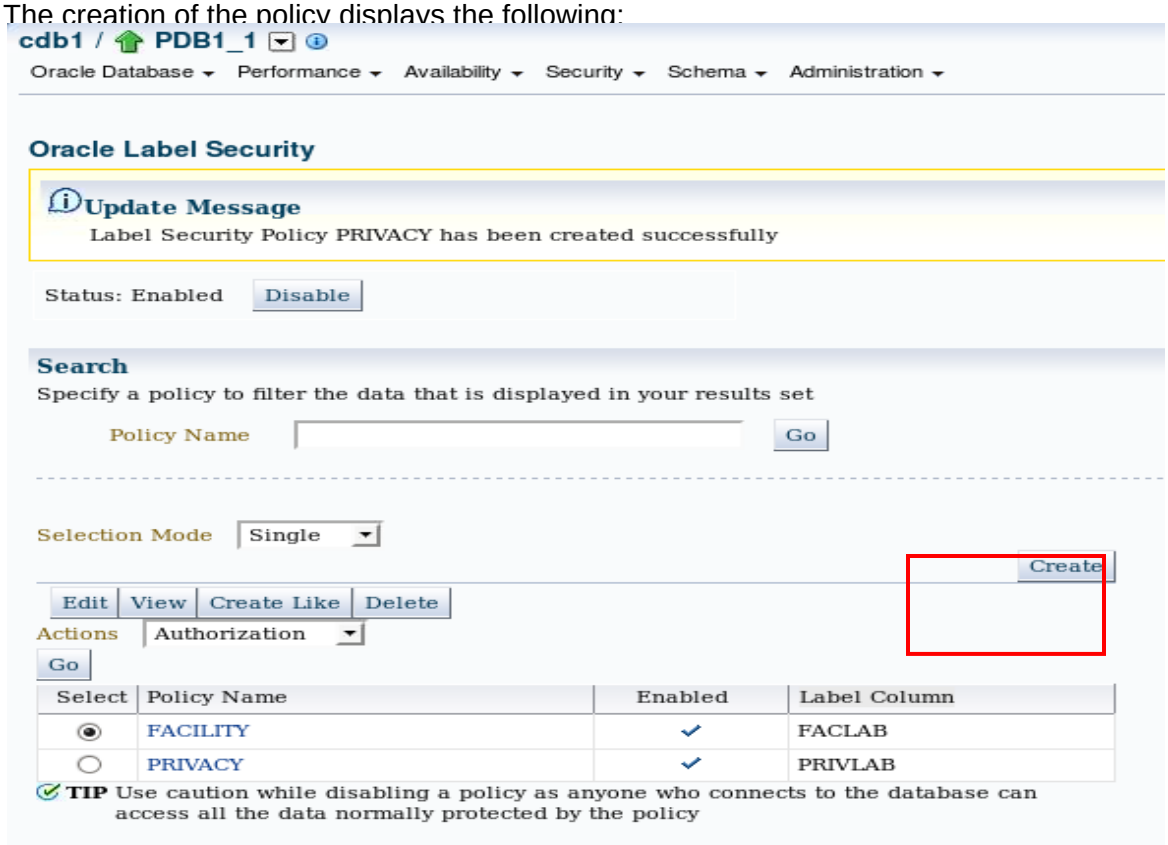
Page Description	Choices or Value
Step 5: Database Login	Click OK to close the error message. Database: `PDB1_1` is displayed. Username: `LBACSYS` Password: `oracle_4U` Select: Save As "`lbacsys_cred`" Click **Login**.

It succeeded. The `FACILITY` policy is displayed as the single existing OLS policy in `pdb1_1`.

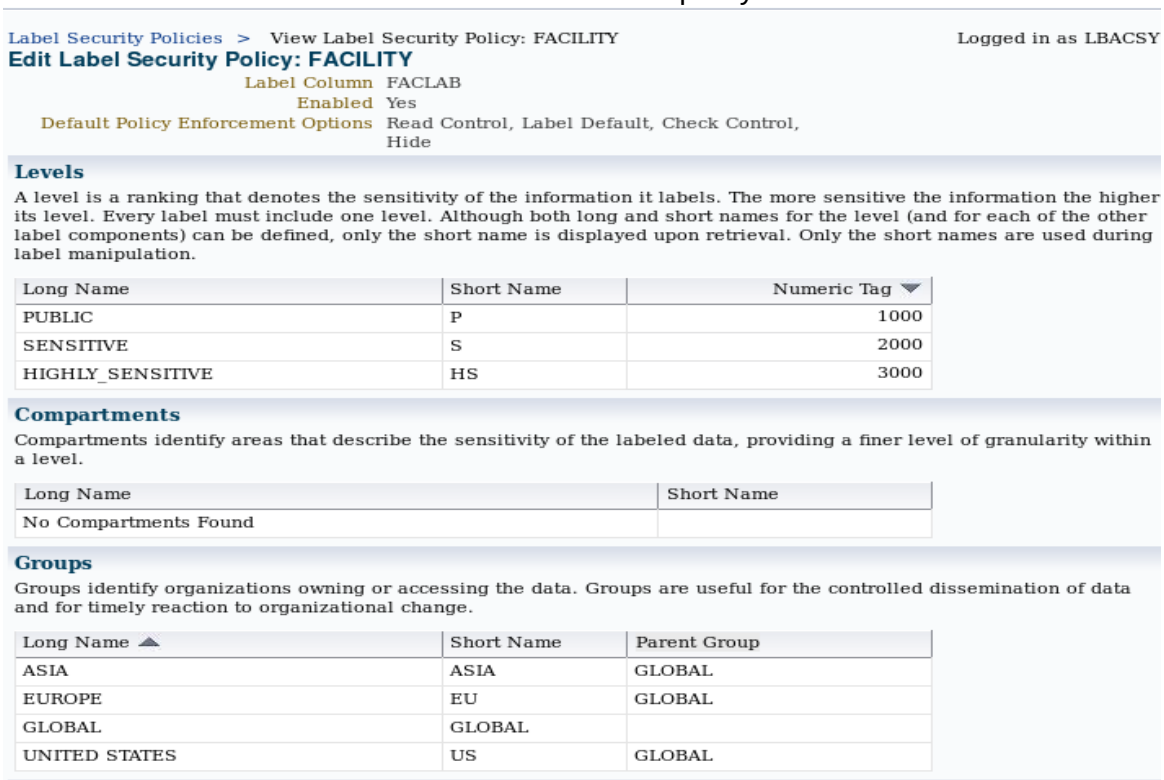
5. Create the `PRIVACY` policy using EM CC. You create two sensitivity levels as described in the specification presented in the scenario.

Step	Page	Action
a.	Oracle Label Security	Click **Create**.
b.	Create Label Security Policy	Enter the following details: Name: **`PRIVACY`** Label Column: **`PRIVLAB`** Select "**Hide Label Column**."
c.	Create Label Security Policy	In the Default Policy Enforcement Options section: Select **Apply Policy Enforcement**. Select "**For all queries (READ_CONTROL)**." Select "**For update and insert operations so that modified or new rows are read accessible (CHECK_CONTROL)**." Click **OK**.
d.	Oracle Label Security	Update message: Label Security Policy `PRIVACY` has been created successfully

a. The creation of the policy displays the following:



b. Click the `FACILITY` link to view the attributes of the policy.



6. Create the labels for the `PRIVACY` policy as shown in the preceding specification by using EM CC. Click the locator link at the top of the page: “Oracle Label Security”

Step	Page	Action
a.	Oracle Label Security	Select `PRIVACY`. Click **Edit**.
b.	Edit Label Security Policy: `PRIVACY`	Click the Label Components tab. In the Levels section, click **Add 5 Rows**.

c.	Edit Label Security Policy: `PRIVACY`	Enter the following information.
		Long Name : Short Name: Numeric Tag
		`CONFIDENTIAL : C : 1000`
		`SENSITIVE : S : 2000`
		Remove the extra three rows. (Select the empty rows and click **Delete**.)
		Click **Apply**.

d.	Edit Label Security Policy: `PRIVACY`	Update message: `Label Security Policy PRIVACY has been modified successfully` Click the locator link at the top of the page: Label Security Policies.
e.	Data Labels: PRIVACY	Click **Add**.
f.	Create Data Label	Enter the following details: Numeric Tag: `101000` Level: `C` **Note:** You can also click the Flashlight icon next to the Level field and select the value from the page that is displayed. Click **OK**.
g.	Data Labels: Privacy	Click Add.
h.	Create Data Label	Enter the following details: Numeric Tag: `102000` Level: `S` Click **OK**.
i.	Data Labels: Privacy	Update message: The object has been created successfully Click the locator link at the top of the page: Label Security Policies.

7. Using a terminal window, set the user authorizations for the `FACILITY` and `PRIVACY` policies. Specify the user's initial session label and an initial default row label when setting up user authorizations. These authorizations are kept in the OLS data dictionary tables for each user. Using SQL*Plus, execute `users_auth.sql`. This sets the user authorization labels for the three users: `MYCO_EMP`, `MYCO_PLANNING`, and `MYCO_MGR`. Later, data access rights will be limited by applying the labels to the data.

```
SQL> set echo on
```

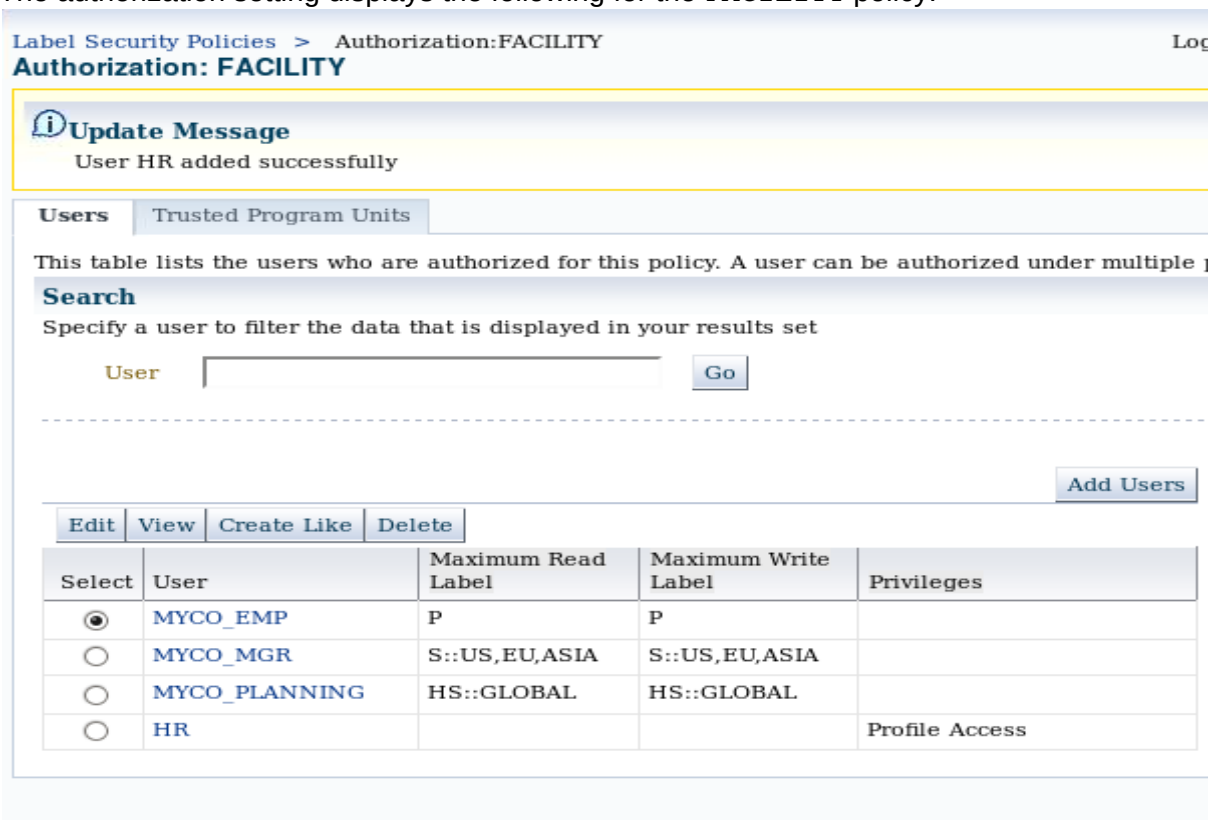
```
SQL> @$HOME/labs/OLS/users_auth.sql
SQL> SET ECHO OFF
SQL>
SQL> -- ***************************************************
SQL> -- Setting User Authorizations for users:
SQL> -- MYCO_EMP
SQL> -- MYCO_MGR
SQL> -- MYCO_PLANNING
SQL> -- ***************************************************
SQL> CONNECT lbacsys/oracle_4U@localhost:1521/pdb1_1
Enter password: ******
Connected.
SQL> -- ***************************************************
SQL> -- Setting MYCO_EMP user label authorizations
SQL> -- Setting MYCO_MGR user label authorizations
SQL> -- Setting MYCO_PLANNING user label authorizations
SQL> -- ***************************************************
SQL> BEGIN
  2     SA_USER_ADMIN.SET_USER_LABELS ('PRIVACY',
  3                                    'MYCO_MGR','C');
  4     SA_USER_ADMIN.SET_USER_LABELS ('FACILITY',
                                        'MYCO_EMP','P');
  5     SA_USER_ADMIN.SET_USER_LABELS ('FACILITY',
                              'MYCO_MGR','S::US,EU,ASIA');
  6     SA_USER_ADMIN.SET_USER_LABELS ('FACILITY',
  7                           'MYCO_PLANNING','HS::GLOBAL');
  8  END;
  9  /

PL/SQL procedure successfully completed.

SQL>
```

8. Set the user authorizations for the `HR` user by using Enterprise Manager Cloud Control. The `HR` user needs full read and write access (`FULL`) to the data and must be able to change the session labels and session privileges to those of another user (`PROFILE_ACCESS`) for both the `FACILITY` and `PRIVACY` policies.

Step	Page	Action
a.	Data Labels: `PRIVACY`	Click the Oracle Label Security link.
b.	Oracle Label Security	Select the **`FACILITY`** policy. Select **Authorization** from the **Actions** menu. Click **Go**.

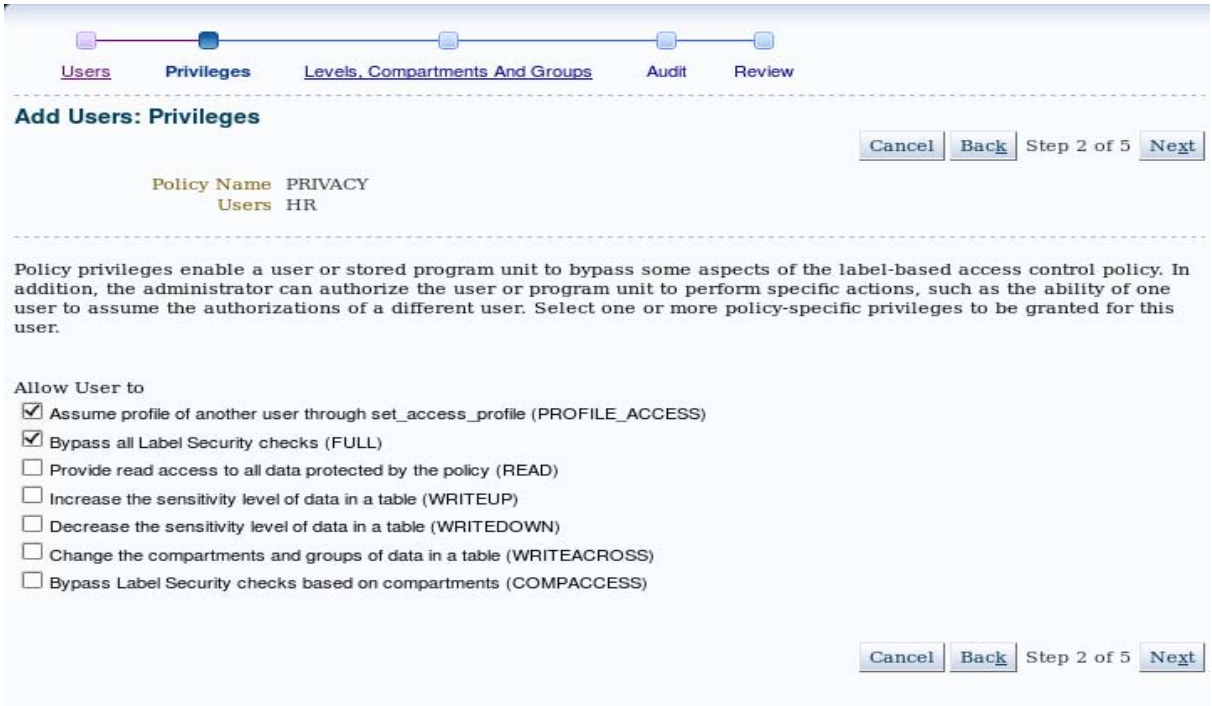
c.	Authorization: `FACILITY`	Click **Add Users**.
d.	Add Users: Users	Click **Add**.
e.	Search and Select: User	Select the **`HR`** user and click **Select**.
f.	Add Users: Users	Click **Next**.
g.	Add Users: Privileges	Select "**Assume profile of another user through set_access_profile (PROFILE_ACCESS)**." Select "**Bypass all Label Security checks (FULL)**." Click **Next**.
h.	Add Users: Levels, Compartments And Groups	Click **Next**.
i.	Add Users: Audit	Click **Next**.
j.	Add Users: Review	Click **Finish**.
k.	Authorization: `FACILITY`	Update message: User `HR` added successfully

The authorization setting displays the following for the `FACILITY` policy:



9. Repeat the procedure to set user authorizations for the `HR` user for the `PRIVACY` policy. Click **Label Security Policies** to return to the Label Security Policies page. Give the `HR` user the `PROFILE_ACCESS` and `FULL` privileges on the `PRIVACY` policy.

   The authorization setting displays the following at step g:



When complete, the authorization setting displays the following for the `PRIVACY` policy:

10. Apply the `FACILITY` policy to the `LOCATIONS` table. You can apply Oracle Label Security policies to entire application schemes or individual application tables. In a SQL*Plus session, execute the `apply_FAC_locations.sql` script.

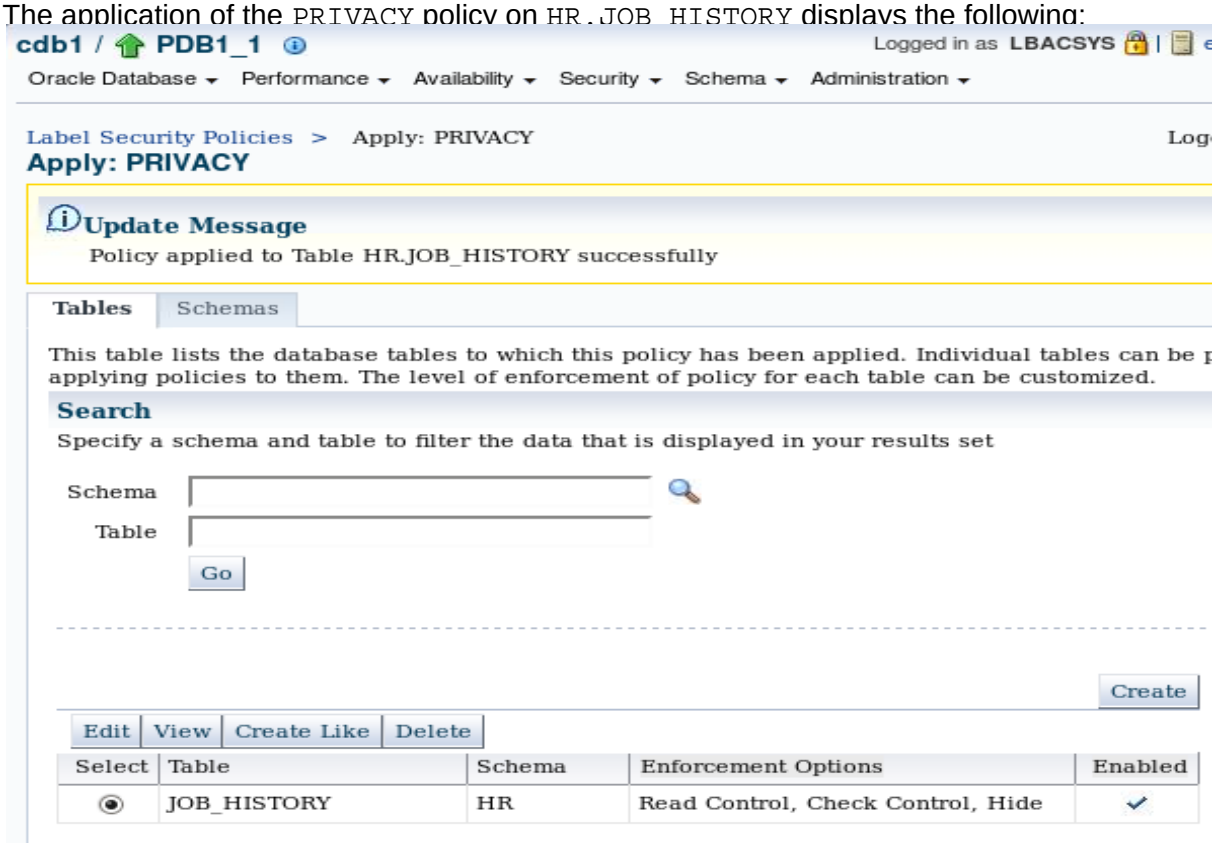
```
SQL> @$HOME/labs/OLS/apply_FAC_locations
SQL> set echo on
SQL> --
SQL> -- ************************************************
```

```
SQL> -- Applying FACILITY policy to hr.locations table.
SQL> -- ************************************************
SQL>
SQL> CONNECT lbacsys/oracle_4U@localhost:1521/pdb1_1
Enter password: *****
Connected.
SQL>
SQL> Begin
  2     sa_policy_admin.apply_table_policy (
  3       POLICY_NAME => 'FACILITY',
  4       SCHEMA_NAME => 'HR',
  5       TABLE_NAME  => 'LOCATIONS',
  6       TABLE_OPTIONS => NULL,
  7       LABEL_FUNCTION => NULL);
 10  END;
 11  /

PL/SQL procedure successfully completed.

SQL>
```

11. Apply the `PRIVACY` policy to the `JOB_HISTORY` table. Use Enterprise Manager Cloud Control to apply the policy.

Step	Page	Action
a.	Authorization: `PRIVACY`	Click the Label Security Policies link.
b.	Oracle Label Security	Select the `PRIVACY` policy. Select **Apply** from the Actions menu. Click **Go**.
c.	Apply: `PRIVACY`	Click **Create**.
d.	Add Table	Enter `HR.JOB_HISTORY` in the Table field. Click **OK**.
e.	Apply: `PRIVACY`	An update message is displayed. Click the **Label Security Policies** link to return to the Label Security Policies page.

The application of the `PRIVACY` policy on `HR.JOB_HISTORY` displays the following:



12. View the protection options of the policies that you created.
    a. On the Label Security Policies page, click the `FACILITY` policy.

Label Security Policies > View Label Security Policy: FACILITY

View Label Security Policy: FACILITY

Label Column FACLAB
Enabled Yes
Default Policy Enforcement Options Read Control, Label Default, Check Control, Hide

Levels

A level is a ranking that denotes the sensitivity of the information it labels. The more sensitive higher its level. Every label must include one level. Although both long and short names for the of the other label components) can be defined, only the short name is displayed upon retrieval. names are used during label manipulation.

Long Name	Short Name	Numeric Tag
PUBLIC	P	1000
SENSITIVE	S	2000
HIGHLY_SENSITIVE	HS	3000

Compartments

Compartments identify areas that describe the sensitivity of the labeled data, providing a finer l within a level.

Long Name	Short Name
No Compartments Found	

Groups

Groups identify organizations owning or accessing the data. Groups are useful for the controlle data and for timely reaction to organizational change.

Long Name	Short Name	Parent Group
ASIA	ASIA	GLOBAL
EUROPE	EU	GLOBAL
GLOBAL	GLOBAL	
UNITED STATES	US	GLOBAL

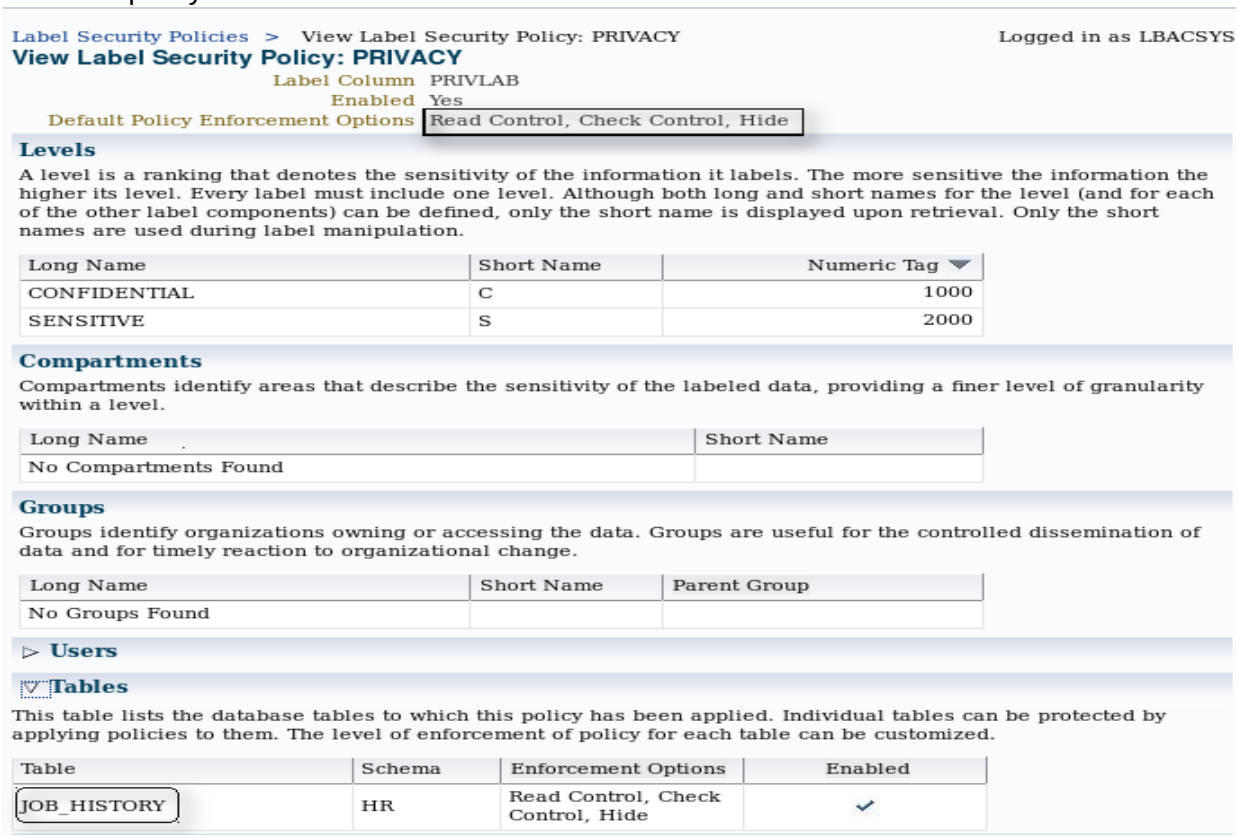
b. Note how the policy is enforced for the `LOCATIONS` table. Expand the Tables folder at the bottom of the page.

Tables

This table lists the database tables to which this policy has been applied. Individual tables can be protected by applying policies to them. The level of enforcement of policy for each table can be customized.

Table	Schema	Enforcement Options	Enabled
LOCATIONS	HR	Read Control, Label Default, Check Control, Hide	✓

c. Click the Label Security Policies link. On the Label Security Policies page, click the `PRIVACY` policy.



d. Log out of Enterprise Manager and close the browser.

13. Before you can test the policy, you must add labels to the data. In SQL*Plus, execute the `add_labels.sql` script, which adds the labels to the rows of data in the protected tables. This update is done by the `SYS` user who has the `EXEMPT ACCESS POLICY` system privilege and `HR` user who has `FULL` access rights in `PRIVACY` policy.

    Note: The number of rows updated in the `JOB_HISTORY` table varies depending on the current date; any row with an `END_DATE` more than 10 years is given a `SENSITIVE` label.

```
SQL> @$HOME/labs/OLS/add_labels.sql
SQL> set echo on
SQL>
SQL> SPOOL ols_add_labels_to_data.log
SQL>
SQL> -- ***************************************************
SQL> -- Populating Data - Enter password for HR schema
SQL> -- ***************************************************
SQL>
SQL> connect sys/oracle_4U@localhost:1521/pdb1_1 as sysdba
Connected.
SQL>
SQL> -- ****************************************************
SQL> -- SETTING LABELS FOR FACILITY POLICY
```

```
SQL> -- ****************************************************
SQL>
SQL> -- ****************************************************
SQL> -- Update Labels for Sites In ASIA
SQL> -- ****************************************************
SQL>
SQL> update hr.locations
  2       set faclab = char_to_label('FACILITY','S::ASIA')
  3       where upper(city) in
  4            ('BEIJING','TOKYO','SINGAPORE');

3 rows updated.

SQL>
SQL> -- ****************************************************
SQL> -- Update Labels for Sites In US
SQL> -- ****************************************************
SQL>
SQL> update hr.locations
  2       set faclab = char_to_label('FACILITY','HS::US')
  3       where upper(city) in ('SOUTH SAN FRANCISCO');

1 row updated.

SQL>
SQL> -- ****************************************************
SQL> -- Update Labels for all remaining locations
SQL> -- ****************************************************
SQL>
SQL> update hr.locations
  2       set faclab = char_to_label('FACILITY','P')
  3       where faclab is NULL;

19 rows updated.

SQL>
SQL> -- ****************************************************
SQL> -- SETTING LABELS FOR PRIVACY POLICY
SQL> -- ****************************************************
SQL> connect hr/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
```

```
SQL> update hr.job_history
  2       set privlab = char_to_label('PRIVACY','S')
  3       where ((to_char(sysdate,'YYYY')
  4             - to_char(end_date,'YYYY')) > 10);

2 rows updated.

SQL>
SQL> update hr.job_history
  2       set privlab = char_to_label('PRIVACY','C')
  3       where ((to_char(sysdate,'YYYY')
  4             - to_char(end_date,'YYYY')) <= 10);

8 rows updated.

SQL>
SQL> COMMIT;

Commit complete.

SQL>
SQL> Spool off;
SQL>
```

14. Test the `FACILITY` policy implementation. After establishing policies to tables and users, and adding labels to the data, you can now test them. To test the access for each user, execute the `test_loc.sql` script.

```
SQL> @$HOME/labs/OLS/test_loc.sql
SQL> set echo on
SQL>
SQL> spool ols_test_facility.log
SQL>
SQL> set linesize 57
SQL> set pagesize 100
SQL> col "FACILITY LABEL" format a8 heading "FACILITY|LABEL"
SQL> col street_address format a20 word_wrap
SQL> col city format a10 word_wrap
SQL> col state_province format a12 truncate
SQL> col postal_code format a8 truncate
SQL> col location_id format 9999 heading "LOC"
SQL>
SQL> set echo on
SQL> -- ****************************************************
```

```
SQL> -- * Connect to the Oracle pluggable database PDB1_1 as
SQL> -- * Application User  myco_emp
SQL> -- *
SQL> -- * select locations.*, label_to_char(faclab)
SQL> -- *  "FACILITY LABEL" from hr.locations;
SQL> -- *
SQL> -- ****************************************************
SQL>
SQL>
SQL> Pause Hit Return To Continue
Hit Return To Continue

SQL>
SQL>
SQL> connect myco_emp/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
SQL> select locations.*, label_to_char(faclab)
  2    "FACILITY LABEL" from hr.locations;

  LOC STREET_ADDRESS       POSTAL_C CITY
----- -------------------- -------- ----------
                FACILITY
STATE_PROVIN CO LABEL
------------ -- --------
 1000 1297 Via Cola di Rie 00989    Roma
             IT P

 1100 93091 Calle della    10934    Venice
      Testa
             IT P

 1300 9450 Kamiya-cho      6823     Hiroshima
             JP P

 1400 2014 Jabberwocky Rd  26192    Southlake
Texas        US P

 1600 2007 Zagora St       50090    South
                                    Brunswick
New Jersey   US P
```

```
 1700 2004 Charade Rd      98199    Seattle
Washington   US P

 1800 147 Spadina Ave      M5V 2L7  Toronto
Ontario      CA P

 1900 6092 Boxwood St      YSW 9T2  Whitehorse
Yukon        CA P

 2100 1298 Vileparle (E)   490231   Bombay
Maharashtra  IN P

 2200 12-98 Victoria       2901     Sydney
      Street
New South Wa AU P

 2400 8204 Arthur St                London
            UK P

 2500 Magdalen Centre, The OX9 9ZB  Oxford
      Oxford Science Park
Oxford       UK P

 2600 9702 Chester Road    09629850 Stretford
Manchester   UK P

 2700 Schwanthalerstr.     80925    Munich
      7031
Bavaria      DE P

 2800 Rua Frei Caneca 1360 01307-00 Sao Paulo

Sao Paulo    BR P

 2900 20 Rue des           1730     Geneva
      Corps-Saints
Geneve       CH P

 3000 Murtenstrasse 921    3095     Bern
BE           CH P

 3100 Pieter               3029SK   Utrecht
```

```
      Breughelstraat 837
Utrecht      NL P

 3200 Mariano Escobedo     11932    Mexico
      9991                          City
Distrito Fed MX P


19 rows selected.

SQL>
SQL> Pause  Hit Return To Continue
Hit Return To Continue

SQL>
SQL> -- ****************************************************
SQL> -- * Connect to the Oracle pluggable database PDB1_1 as
SQL> -- * Application User  myco_mgr
SQL> -- *
SQL> -- * select locations.*, label_to_char(faclab)
SQL> -- *  "FACILITY LABEL" from hr.locations;
SQL> -- *
SQL> -- ****************************************************
SQL>
SQL>
SQL> Pause Hit Return To Continue
Hit Return To Continue

SQL>
SQL> connect myco_mgr/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
SQL> select locations.*, label_to_char(faclab)
  2    "FACILITY LABEL" from hr.locations;

  LOC STREET_ADDRESS       POSTAL_C CITY
----- -------------------- -------- ----------
                FACILITY
STATE_PROVIN CO LABEL
------------ -- --------
 1000 1297 Via Cola di Rie 00989    Roma
             IT P
```

```
 1100 93091 Calle della    10934    Venice
      Testa
             IT P

 1200 2017 Shinjuku-ku     1689     Tokyo
Tokyo Prefec JP S::ASIA

 1300 9450 Kamiya-cho      6823     Hiroshima
             JP P

 1400 2014 Jabberwocky Rd  26192    Southlake
Texas        US P

 1600 2007 Zagora St       50090    South
                                    Brunswick
New Jersey   US P

 1700 2004 Charade Rd      98199    Seattle
Washington   US P

 1800 147 Spadina Ave      M5V 2L7  Toronto
Ontario      CA P

 1900 6092 Boxwood St      YSW 9T2  Whitehorse
Yukon        CA P

 2000 40-5-12 Laogianggen  190518   Beijing
             CN S::ASIA

 2100 1298 Vileparle (E)   490231   Bombay
Maharashtra  IN P

 2200 12-98 Victoria       2901     Sydney
      Street
New South Wa AU P

 2300 198 Clementi North   540198   Singapore
             SG S::ASIA

 2400 8204 Arthur St                London
             UK P
```

```
 2500 Magdalen Centre, The OX9 9ZB  Oxford
      Oxford Science Park
Oxford       UK P

 2600 9702 Chester Road    09629850 Stretford
Manchester   UK P

 2700 Schwanthalerstr.     80925    Munich
      7031
Bavaria      DE P

 2800 Rua Frei Caneca 1360 01307-00 Sao Paulo

Sao Paulo    BR P

 2900 20 Rue des           1730     Geneva
      Corps-Saints
Geneve       CH P

 3000 Murtenstrasse 921    3095     Bern
BE           CH P

 3100 Pieter               3029SK   Utrecht
      Breughelstraat 837
Utrecht      NL P

 3200 Mariano Escobedo     11932    Mexico
      9991                          City
Distrito Fed MX P


22 rows selected.

SQL>
SQL> Pause Hit Return To Continue
Hit Return To Continue

SQL>
SQL> -- ****************************************************
SQL> -- * Connect to the Oracle pluggable database PDB1_1 as
SQL> -- * Application User  myco_planning
```

```
SQL> -- *
SQL> -- * select locations.*, label_to_char(faclab)
SQL> -- *  "FACILITY LABEL" from hr.locations;
SQL> -- *
SQL> -- ****************************************************
SQL>
SQL> Pause Hit Return To Continue
Hit Return To Continue

SQL>
SQL> connect myco_planning/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
SQL> select locations.*, label_to_char(faclab)
  2    "FACILITY LABEL" from hr.locations;

  LOC STREET_ADDRESS       POSTAL_C CITY
----- -------------------- -------- ----------
                FACILITY
STATE_PROVIN CO LABEL
------------ -- --------
 1000 1297 Via Cola di Rie 00989    Roma
            IT P

 1100 93091 Calle della    10934    Venice
      Testa
            IT P

 1200 2017 Shinjuku-ku     1689     Tokyo
Tokyo Prefec JP S::ASIA

 1300 9450 Kamiya-cho      6823     Hiroshima
            JP P

 1400 2014 Jabberwocky Rd  26192    Southlake
Texas        US P

 1500 2011 Interiors Blvd  99236    South San
                                    Francisco
California   US HS::US

 1600 2007 Zagora St       50090    South
```

```
                                        Brunswick
New Jersey   US P

 1700 2004 Charade Rd      98199    Seattle
Washington   US P

 1800 147 Spadina Ave      M5V 2L7  Toronto
Ontario      CA P

 1900 6092 Boxwood St      YSW 9T2  Whitehorse
Yukon        CA P

 2000 40-5-12 Laogianggen  190518   Beijing
             CN S::ASIA

 2100 1298 Vileparle (E)   490231   Bombay
Maharashtra  IN P

 2200 12-98 Victoria       2901     Sydney
      Street
New South Wa AU P

 2300 198 Clementi North   540198   Singapore
             SG S::ASIA

 2400 8204 Arthur St                London
             UK P

 2500 Magdalen Centre, The OX9 9ZB  Oxford
      Oxford Science Park
Oxford       UK P

 2600 9702 Chester Road    09629850 Stretford
Manchester   UK P

 2700 Schwanthalerstr.     80925    Munich
      7031
Bavaria      DE P

 2800 Rua Frei Caneca 1360 01307-00 Sao Paulo

Sao Paulo    BR P
```

```
 2900 20 Rue des           1730     Geneva
      Corps-Saints
Geneve       CH P

 3000 Murtenstrasse 921    3095     Bern
BE           CH P

 3100 Pieter               3029SK   Utrecht
      Breughelstraat 837
Utrecht      NL P

 3200 Mariano Escobedo     11932    Mexico
      9991                          City
Distrito Fed MX P


23 rows selected.

SQL>
SQL> spool off;
SQL>
```

15. Test the `PRIVACY` policy implementation. After establishing policies for tables and users, and adding labels to the data, you can now test them. To test the access for each user, execute the `test_hist.sql` script. The number of rows returned for `MYCO_EMP` and `MYCO_MGR` vary based on `SYSDATE`; rows with `END_DATE` greater than 10 years will have a `SENSITIVE` label.

```
SQL> @$HOME/labs/OLS/test_hist.sql
SQL> set echo on
SQL>
SQL> set linesize 57
SQL> set pagesize 32
SQL> col "PRIVACY LABEL" format a8 HEADING "PRIVACY|LABEL"
SQL> col org_name format a10
SQL> col org_id format 9999
SQL> col hours format 9999
SQL> col expenses format 99999
SQL>
SQL>
SQL> -- ******************************************************
SQL> -- * Connect to the Oracle pluggable database PDB1_1 as
SQL> -- * Application User  myco_emp
SQL> -- *
```

```
SQL> -- * select job_history.*, label_to_char(PRIVLAB)
SQL> -- *  "PRIVACY LABEL" from hr.job_history;
SQL> --
SQL> -- ****************************************************
SQL>
SQL> -- Hit Return To Continue
SQL> PAUSE

SQL>
SQL> connect myco_emp/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
SQL> select job_history.*, label_to_char(PRIVLAB)
  2    "PRIVACY LABEL" from hr.job_history;

no rows selected

SQL>
SQL>
SQL> -- ****************************************************
SQL> -- * Connect to the Oracle pluggable database PDB1_1 as
SQL> -- * Application User  myco_mgr
SQL> -- *
SQL> -- * select job_history.*, label_to_char(PRIVLAB)
SQL> -- *  "PRIVACY LABEL" from hr.job_history;
SQL> --
SQL> -- ****************************************************
SQL>
SQL> -- Hit Return To Continue
SQL> PAUSE

SQL>
SQL> connect myco_mgr/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
SQL> select job_history.*, label_to_char(PRIVLAB)
  2    "PRIVACY LABEL" from hr.job_history;

EMPLOYEE_ID START_DAT END_DATE  JOB_ID     DEPARTMENT_ID
----------- --------- --------- ---------- -------------
PRIVACY
LABEL
```

```
--------
     102 13-JAN-93 24-JUL-98 IT_PROG                 60
C

    101 28-OCT-01 15-MAR-05 AC_MGR                   110
C

     201 17-FEB-96 19-DEC-99 MK_REP                   20
C

     114 24-MAR-98 31-DEC-99 ST_CLERK                 50
C

     122 01-JAN-99 31-DEC-99 ST_CLERK                 50
C

     176 24-MAR-98 31-DEC-98 SA_REP                   80
C

     176 01-JAN-99 31-DEC-99 SA_MAN                   80
C

     200 01-JUL-94 31-DEC-98 AC_ACCOUNT               90
C


8 rows selected.

SQL>
SQL> -- Hit Return To Continue
SQL> PAUSE

SQL>
SQL>
SQL>
SQL> -- ****************************************************
SQL> -- * Connect to the Oracle pluggable database PDB1_1 as
SQL> -- *  Application User HR
SQL> -- ****************************************************
SQL>
SQL> connect hr/oracle_4U@localhost:1521/pdb1_1
Connected.
```

```
SQL>
SQL> -- ***********************************************
SQL> -- * User HR has Oracle Label Security FULL and
SQL> -- * PROFILE_ACCESS privileges on policies FACILITY
SQL> -- * and PRIVACY
SQL> -- *
SQL> -- * select job_history.*, label_to_char(PRIVLAB)
SQL> -- * "PRIVACY LABEL" from hr.job_history;
SQL> -- *
SQL> -- ***********************************************
SQL>
SQL> select job_history.*, label_to_char(PRIVLAB)
  2    "PRIVACY LABEL" from hr.job_history;

EMPLOYEE_ID START_DAT END_DATE  JOB_ID     DEPARTMENT_ID
----------- --------- --------- ---------- -------------
PRIVACY
LABEL
--------
        102 13-JAN-93 24-JUL-98 IT_PROG              60
C

        101 21-SEP-89 27-OCT-93 AC_ACCOUNT          110
S

        101 28-OCT-93 15-MAR-97 AC_MGR              110
C

        201 17-FEB-96 19-DEC-99 MK_REP               20
C

        114 24-MAR-98 31-DEC-99 ST_CLERK             50
C

        122 01-JAN-99 31-DEC-99 ST_CLERK             50
C

        200 17-SEP-87 17-JUN-93 AD_ASST              90
S

        176 24-MAR-98 31-DEC-98 SA_REP               80
C
```

```
        176 01-JAN-99 31-DEC-99 SA_MAN                80
C

        200 01-JUL-94 31-DEC-98 AC_ACCOUNT            90
C


10 rows selected.

SQL>
```

Practice 13-3: Cleaning Up OLS Policies

Overview

In this practice, you clean up all OLS policies.

Tasks

1. Drop all OLS policies by running the `cleanup_OLS.sql` cleanup script.

   Note: You are prompted first for the `SYSTEM` user password, and then for the `LBACSYS` user password.

```
SQL> @$HOME/labs/OLS/cleanup_OLS.sql
SQL> SET ECHO ON
SQL> CONNECT system/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
SQL> -- Cleanup from previous runs
SQL>
SQL> DROP USER myco_EMP;

User dropped.

SQL> DROP USER myco_MGR;

User dropped.

SQL> DROP USER myco_PLANNING;

User dropped.

SQL>
SQL> CONNECT lbacsys/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL>
SQL> EXECUTE SA_SYSDBA.DROP_POLICY('FACILITY',TRUE);

PL/SQL procedure successfully completed.

SQL> EXECUTE SA_SYSDBA.DROP_POLICY('PRIVACY',TRUE);

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

Practices for Lesson 14: Oracle Data Redaction

Chapter 14

Practices for Lesson 14: Overview

Practices Overview

In the practice for this lesson, you use Oracle Data Redaction to redact values of shielded columns of the `HR.EMPLOYEES` table.

Practice 14-1: Redacting Protected Column Values with FULL Redaction

Overview

In this practice you use FULL data redaction to display:

- The employees' salary from the HR.EMPLOYEES as 0 instead of the real values
- The employees' last name as blank. Louise is the only exception to be allowed to view the employees' last names.

Tasks

1. Display the current values from the HR.EMPLOYEES table before redaction.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus sec

Enter password: ******
Last Successful login time: Mon Jun 17 2013 23:54:00 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> CREATE USER louise IDENTIFIED BY oracle_4U;

User created.

SQL> GRANT create session TO louise;

Grant succeeded.

SQL> GRANT select ON hr.employees TO louise;

Grant succeeded.

SQL> col first_name format A12
SQL> col last_name format A10
SQL> col salary format 999999
SQL> SELECT employee_id, last_name, salary, commission_pct
```

```
     FROM  hr.employees
     WHERE department_id = 100;
  2    3
EMPLOYEE_ID LAST_NAME  SALARY COMMISSION_PCT
----------- ---------- ------- --------------
    108     Greenberg    12008
    109     Faviet        9000
    110     Chen          8200
    111     Sciarra       7700
    112     Urman         7800
    113     Popp          6900

6 rows selected.

SQL>
```

2. Define a redaction policy for the `HR.EMPLOYEES` table specifying full redacting for the `SALARY` column. `SALARY` is defined as `NUMBER(8,2)`. In this example, by setting `EXPRESSION` to `1=1`, redaction is always performed because the expression always evaluates to true.
   The policy is enabled by default.

```
BEGIN
  DBMS_REDACT.ADD_POLICY
  (object_schema => 'HR',
   object_name   => 'EMPLOYEES',
   policy_name   => 'EMP_POLICY',
   column_name   => 'SALARY',
   function_type => DBMS_REDACT.FULL,
   expression    => '1=1');
  END;
 /
```

   a. The `SEC` user also needs the privilege to create redaction policies. Grant `SEC` the ability to execute the package that creates redaction policies.

```
SQL> CONNECT / AS SYSDBA

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL>
SQL> GRANT execute ON dbms_redact TO sec;
```

```
Grant succeeded.

SQL>
```

b. Connect as `SEC` to create the redaction policy.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> BEGIN
  DBMS_REDACT.ADD_POLICY
  (object_schema => 'HR',
   object_name   => 'EMPLOYEES',
   policy_name   => 'EMP_POLICY',
   column_name   => 'SALARY',
   function_type => DBMS_REDACT.FULL,
   expression    => '1=1');
  END;
/
  2    3    4    5    6    7    8    9   10
PL/SQL procedure successfully completed.

SQL>
```

3. Query `REDACTION_POLICIES` to verify that the policy has been created and is enabled. This view also shows under what condition the redaction will be performed as shown in the `EXPRESSION` column.

```
SQL> COL object_owner FORMAT A12
SQL> COL object_name FORMAT A12
SQL> COL policy_name FORMAT A14
SQL> COL expression FORMAT A12
SQL> COL enable FORMAT A6
SQL> COL policy_description FORMAT A10
SQL> SELECT * FROM redaction_policies;

OBJECT_OWNER OBJECT_NAME  POLICY_NAME    EXPRESSION   ENABLE
------------ ------------ -------------- ------------ ------
POLICY_DES
----------
HR           EMPLOYEES    EMP_POLICY     1=1          YES


SQL>
```

4. Display which columns will be redacted and what type of redaction will take place.

```
SQL> COL column_name FORMAT A14
```

```
SQL> COL function_type FORMAT A18
SQL> COL function_parameters FORMAT A20
SQL> SELECT object_owner, object_name, column_name,
     function_type, function_parameters
     FROM redaction_columns;
  2    3
OBJECT_OWNER OBJECT_NAME      COLUMN_NAME    FUNCTION_TYPE
------------ ---------------- -------------- ------------------
FUNCTION_PARAMETERS
--------------------
HR           EMPLOYEES        SALARY         FULL REDACTION

SQL>
```

5. Now query the `HR.EMPLOYEES` table again and note that the value of the `SALARY` column is `0` for all displayed rows.
   a. First grant the `SELECT` privilege to `SH`.

```
SQL> GRANT select ON hr.employees TO sh;

Grant succeeded.

SQL>
```

   b. Connect as `SH`. If `SH` is locked, unlock the account.

```
SQL> ALTER USER sh IDENTIFIED BY oracle_4U ACCOUNT UNLOCK;

User altered.

SQL> CONNECT sh
Enter password: ******
Connected.
SQL>
```

   c. Run the same select as in task 1.

```
SQL> SELECT employee_id, last_name, salary, commission_pct
     FROM hr.employees
     WHERE department_id = 100;
  2    3
EMPLOYEE_ID LAST_NAME  SALARY COMMISSION_PCT
----------- ---------- ------- --------------
    108     Greenberg        0
    109     Faviet           0
    110     Chen             0
    111     Sciarra          0
```

```
     112     Urman             0
     113     Popp              0

6 rows selected.

SQL>
```

Why the value displayed for the `SALARY` column is `0`?
The default value for all `NUMBER` data type columns when full redacted is `0`.

```
SQL> SELECT number_value FROM REDACTION_VALUES_FOR_TYPE_FULL;

NUMBER_VALUE
------------
           0

SQL>
```

6. If you query as `SYSDBA`, the "real" value is displayed, not the redacted value as shown in this example. Any user who is granted the `EXEMPT REDACTION POLICY` privilege bypasses any redaction policy.
   a. Connect as `SYSDBA`.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
```

   b. Run the same select as in task 1.

```
SQL> SELECT employee_id, last_name, salary, commission_pct
     FROM hr.employees
     WHERE department_id = 100;
  2    3
EMPLOYEE_ID LAST_NAME  SALARY COMMISSION_PCT
----------- ---------- ------- --------------
     108     Greenberg    12008
     109     Faviet        9000
     110     Chen          8200
     111     Sciarra       7700
     112     Urman         7800
     113     Popp          6900

6 rows selected.

SQL> SELECT * FROM session_privs
     WHERE  privilege like 'EXEMP%';
  2
```

```
PRIVILEGE
----------------------------------------
EXEMPT ACCESS POLICY
EXEMPT IDENTITY POLICY
EXEMPT REDACTION POLICY
EXEMPT DML REDACTION POLICY
EXEMPT DDL REDACTION POLICY

SQL>
```

7. Display the last and first names of all employees.

```
SQL> CONNECT louise
Enter password: ******
Connected.
SQL> SELECT first_name, last_name FROM hr.employees
     WHERE  substr(first_name,1,1)= 'L';
  2
FIRST_NAME   LAST_NAME
------------ ----------
Laura        Bissot
Lex          De Haan
Louise       Doran
Lisa         Ozer
Luis         Popp
Lindsey      Smith

6 rows selected.

SQL>
```

The `LAST_NAME` column is not under full redaction yet.

a. Add the `LAST_NAME` column to the policy for full redaction except for the `Louise` user.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> BEGIN
  DBMS_REDACT.ALTER_POLICY
  (object_schema       => 'HR',
   object_name         => 'EMPLOYEES',
   policy_name         => 'EMP_POLICY',
   action              =>  DBMS_REDACT.ADD_COLUMN,
   column_name         => 'LAST_NAME',
   expression          =>
'SYS_CONTEXT(''USERENV'',''SESSION_USER'')!=''LOUISE''');
```

```
 END;
/
  2    3    4    5    6    7    8    9   10
PL/SQL procedure successfully completed.

SQL>
```

b. The `REDACTION_COLUMNS` view shows masking functions defined on the `HR.EMPLOYEES` table.

```
SQL> SELECT object_owner, object_name, column_name,
     function_type
     FROM redaction_columns;
  2    3
OBJECT_OWNER OBJECT_NAME  COLUMN_NAME    FUNCTION_TYPE
------------ ------------ -------------- ------------------
HR           EMPLOYEES    SALARY         FULL REDACTION
HR           EMPLOYEES    LAST_NAME      FULL REDACTION

SQL>
```

c. Display the values of the `LAST_NAME` column. First connect as `LOUISE` then as `SH`.

```
SQL> CONNECT louise
Enter password: ******
Connected.
SQL> SELECT first_name, last_name FROM hr.employees
     WHERE  substr(first_name,1,1)= 'L'
     ORDER BY 1;

FIRST_NAME   LAST_NAME
------------ -----------
Laura
Lex
Lindsey
Lisa
Louise
Luis

6 rows selected.

SQL> connect sh
Enter password: ******
Connected.
SQL> /
```

```
FIRST_NAME   LAST_NAME
------------ ----------
Laura
Lex
Lindsey
Lisa
Louise
Luis

6 rows selected.

SQL>
```

The result is not fully the expected one. The default value for full redaction applies to the values of all rows omitting the expression of the policy.

The expression of the redaction policy is still set to `1=1`.

d. Modify the expression.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> BEGIN
  DBMS_REDACT.ALTER_POLICY
  (object_schema       => 'HR',
   object_name         => 'EMPLOYEES',
   policy_name         => 'EMP_POLICY',
   action              =>  DBMS_REDACT.MODIFY_EXPRESSION,
   expression          =>
'SYS_CONTEXT(''USERENV'',''SESSION_USER'')!=''LOUISE''');
 END;
/
2    3    4    5    6    7    8    9
PL/SQL procedure successfully completed.

SQL> COL expression format A48
SQL> SELECT policy_name, expression FROM redaction_policies;

POLICY_NAME    EXPRESSION
-------------- ------------------------------------------------
EMP_POLICY     SYS_CONTEXT('USERENV','SESSION_USER')!='LOUISE'

SQL>
```

e. Retest.

```
SQL> CONNECT louise
```

```
Enter password: ******
Connected.
SQL> SELECT first_name, last_name FROM hr.employees
     WHERE  substr(first_name,1,1)= 'L'
     ORDER BY 1;

FIRST_NAME   LAST_NAME
------------ ----------
Laura        Bissot
Lex          De Haan
Lindsey      Smith
Lisa         Ozer
Louise       Doran
Luis         Popp

6 rows selected.

SQL> connect sh
Enter password: ******
Connected.
SQL> /

FIRST_NAME   LAST_NAME
------------ ----------
Laura
Lex
Lindsey
Lisa
Louise
Luis

6 rows selected.

SQL>
```

Practice 14-2: Redacting Protected Column Values with `PARTIAL` Redaction

Overview

In this practice, you use `PARTIAL` data redaction to display the `HIRE_DATE` column values from the `HR.EMPLOYEES` as a partially redacted value instead of the real values.

Tasks

1. Query the `HR.EMPLOYEES` table again and display the `HIRE_DATE` column.

```
SQL> CONNECT louise
enter password: ******
Connected.
SQL> SELECT employee_id, last_name, hire_date
     FROM hr.employees
     WHERE department_id = 100;
  2    3
EMPLOYEE_ID LAST_NAME  HIRE_DATE
----------- ---------- ---------
        108 Greenberg  17-AUG-02
        109 Faviet     16-AUG-02
        110 Chen       28-SEP-05
        111 Sciarra    30-SEP-05
        112 Urman      07-MAR-06
        113 Popp       07-DEC-07

6 rows selected.

SQL>
```

2. Alter the masking policy to redact the `HIRE_DATE` column. In this example, partial redaction is used to mask the actual year of hire.

```
BEGIN
  DBMS_REDACT.ALTER_POLICY
  (object_schema       => 'HR',
   object_name         => 'EMPLOYEES',
   policy_name         => 'EMP_POLICY',
   action              =>  DBMS_REDACT.ADD_COLUMN,
   column_name         => 'HIRE_DATE',
   function_type       =>  DBMS_REDACT.PARTIAL,
   function_parameters => 'MDy2012',
   expression          => '1=1');
 END;
 /
```

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> BEGIN
  DBMS_REDACT.ALTER_POLICY
  (object_schema       => 'HR',
   object_name         => 'EMPLOYEES',
   policy_name         => 'EMP_POLICY',
   action              =>  DBMS_REDACT.ADD_COLUMN,
   column_name         => 'HIRE_DATE',
   function_type       =>  DBMS_REDACT.PARTIAL,
   function_parameters => 'MDy2012',
   expression          => '1=1');
 END;
/
  2    3    4    5    6    7    8    9   10   11   12
PL/SQL procedure successfully completed.

SQL>
```

3. Query `REDACTION_COLUMNS` view to show both masking functions defined on the `HR.EMPLOYEES` table.

```
SQL> SELECT object_owner, object_name, column_name,
     function_type, function_parameters
     FROM redaction_columns;
  2    3
OBJECT_OWNER OBJECT_NAME COLUMN_NAME  FUNCTION_TYPE
FUNCTION_PARAMETERS
------------ ----------- ------------ -----------------
-----------------
HR           EMPLOYEES   SALARY   FULL REDACTION
HR           EMPLOYEES   HIRE_DATE     PARTIAL REDACTION
MDy2012
HR           EMPLOYEES   LAST_NAME     FULL REDACTION

SQL>
```

4. Query `HR.EMPLOYEES` again as the `SH` user. '12' is displayed as the hire year for all the rows selected.

```
SQL> CONNECT sh
Enter password: ******
Connected.
SQL> select employee_id, last_name, hire_date
     from hr.employees
     where department_id = 100;
```

```
  2    3
EMPLOYEE_ID LAST_NAME     HIRE_DATE
----------- ------------- ---------
        108               17-AUG-12
        109               16-AUG-12
        110               28-SEP-12
        111               30-SEP-12
        112               07-MAR-12
        113               07-DEC-12

6 rows selected.

SQL>
```

Practice 14-3: Changing the Default Value for FULL Redaction

Overview

In this practice, you use full redaction to redact the returned data to a fixed value.

You will modify the default value for full redaction of:

- Number data to 10 for the commission percentage of all employees
- Date time data to the first of June, 2005 for the hire date of all employees

Tasks

1. Modify the default value to 10 for full redaction of the commission percentage of all employees.
   a. Display the information from the data dictionary view before updating the default value.

```
SQL> SELECT number_value FROM REDACTION_VALUES_FOR_TYPE_FULL;

NUMBER_VALUE
------------
           0

SQL>
```

   b. Modify the default value.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> exec DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES( -
         NUMBER_VAL => 10)

PL/SQL procedure successfully completed.

SQL>
```

   c. Display the information from the data dictionary view.

```
SQL> SELECT number_value FROM REDACTION_VALUES_FOR_TYPE_FULL;

NUMBER_VALUE
------------
          10

SQL>
```

   d. Add the `COMMISSION_PCT` column to the policy for full redaction.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> BEGIN
  DBMS_REDACT.ALTER_POLICY
```

```
  (object_schema       => 'HR',
   object_name         => 'EMPLOYEES',
   policy_name         => 'EMP_POLICY',
   action              =>  DBMS_REDACT.ADD_COLUMN,
   column_name         => 'COMMISSION_PCT',
   expression          => '1=1');
 END;
/
  2    3    4    5    6    7    8    9   10
PL/SQL procedure successfully completed.

SQL>
```

e. The `REDACTION_COLUMNS` view shows masking functions defined on the `HR.EMPLOYEES` table.

```
SQL> SELECT object_owner, object_name, column_name,
     function_type
     FROM redaction_columns;
  2    3
OBJECT_OWNER OBJECT_NAME  COLUMN_NAMEFUNCTION_TYPE
------------ ------------ --------------- ------------------
HR           EMPLOYEES    COMMISSION_PCT  FULL REDACTION
HR           EMPLOYEES    SALARY          FULL REDACTION
HR           EMPLOYEES    HIRE_DATE       PARTIAL REDACTION
HR           EMPLOYEES    LAST_NAME       FULL REDACTION

SQL>
```

f. Display the values of the `COMMISSION_PCT` column of all employees.

```
SQL> CONNECT sh
Enter password: ******
Connected.
SQL> SELECT commission_pct, first_name FROM hr.employees
     ORDER BY 1 DESC;
   2
```

```
COMMISSION_PCT FIRST_NAME
-------------- --------------------
… rows deleted …
               Shelley
               William
             0 John
             0 Allan
             0 Patrick
             0 Ellen
… rows deleted …
             0 Sundar
             0 Charles
             0 Sundita
             0 Amit

83 rows selected.

SQL>
```

The result still displays the value `0`. After you modify a value, you must restart the database for it to take effect. If you only flush the buffer cache, the real value of the column will be displayed.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289968 bytes
Variable Size             264244944 bytes
Database Buffers          226492416 bytes
Redo Buffers                8032256 bytes
Database mounted.
Database opened.
SQL>
```

g. Display the values of the `COMMISSION_PCT` column of all employees.

```
SQL> CONNECT sh
Enter password: ******
Connected.
```

```
SQL> SELECT commission_pct, first_name FROM hr.employees
     ORDER BY 1 DESC;
  2
```

```
COMMISSION_PCT FIRST_NAME
-------------- --------------------
… rows deleted …
               Shelley
               William
            10 John
            10 Allan
            10 Patrick
            10 Ellen
… rows deleted …
            10 Sundar
            10 Charles
            10 Sundita
            10 Amit

83 rows selected.

SQL>
```

Notice that the default value is only applied to the values that are not NULL.

Question: When you updated the default value to a single, blank space for full redaction of the character data type, you did not restart the instance to get the right result.

Answer: The original default value is the same as the one you set. You just activated the default value for full redaction policies. Whereas, in this current case, the default value for the number data type is different from the original default value.

2. Modify the default value to a 1st of June 2005 for full redaction of the hire date of all employees.
   a. Modify the default value.

```
SQL> CONNECT / AS SYSDBA
Connected.

SQL> SELECT date_value FROM REDACTION_VALUES_FOR_TYPE_FULL;

DATE_VALU
---------
01-JAN-01

SQL> exec DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES( -
          DATE_VAL => '01-JUN-05')

PL/SQL procedure successfully completed.

SQL>
```

b. Display the information from the data dictionary view.

```
SQL> SELECT date_value FROM REDACTION_VALUES_FOR_TYPE_FULL;

DATE_VALU
---------
01-JUN-05

SQL>
```

c. Display the first names and hire dates of all employees.

```
SQL> CONNECT sh
Enter password: ******
Connected.

SQL> SELECT first_name, hire_date FROM hr.employees;

FIRST_NAME           HIRE_DATE
------------------- ---------
… rows deleted …
Michael              17-FEB-12
Pat                  17-AUG-12
Susan                07-JUN-12
Hermann              07-JUN-12
Shelley              07-JUN-12
William              07-JUN-12

83 rows selected.

SQL>
```

The result still displays the previous redacted values, and does not use the default redacting value. The `HIRE_DATE` column is a column to be partially redacted by the `EMP_POLICY` policy. The `HIRE_DATE` column is not under full redaction yet.

d. Add the `HIRE_DATE` column to the policy for full redaction.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> BEGIN
  DBMS_REDACT.ALTER_POLICY
  (object_schema       => 'HR',
   object_name         => 'EMPLOYEES',
   policy_name         => 'EMP_POLICY',
   action              =>  DBMS_REDACT.ADD_COLUMN,
   column_name         => 'HIRE_DATE',
   expression          => '1=1');
```

```
 END;
/
  2    3    4    5    6    7    8    9   10  BEGIN
*
ERROR at line 1:
ORA-28060: A data redaction policy already exists on this
column.
ORA-06512: at "SYS.DBMS_REDACT_INT", line 69
ORA-06512: at "SYS.DBMS_REDACT", line 172
ORA-06512: at line 2

SQL>
```

e. Drop the `HIRE_DATE` column from the policy and add it back for full redaction.

```
SQL> BEGIN
  DBMS_REDACT.ALTER_POLICY
  (object_schema       => 'HR',
   object_name         => 'EMPLOYEES',
   policy_name         => 'EMP_POLICY',
   action              =>  DBMS_REDACT.DROP_COLUMN,
   column_name         => 'HIRE_DATE');
 END;
/
  2    3    4    5    6    7    8    9
PL/SQL procedure successfully completed.

SQL> BEGIN
  DBMS_REDACT.ALTER_POLICY
  (object_schema       => 'HR',
   object_name         => 'EMPLOYEES',
   policy_name         => 'EMP_POLICY',
   action              =>  DBMS_REDACT.ADD_COLUMN,
   column_name         => 'HIRE_DATE',
   expression          => '1=1');
 END;
/
  2    3    4    5    6    7    8    9   10
PL/SQL procedure successfully completed.

SQL>
```

f. Display the first names and hire dates of all employees.

```
SQL> CONNECT sh
Enter password: ******
```

```
Connected.
SQL> SELECT first_name, hire_date FROM hr.employees;

FIRST_NAME   HIRE_DATE
------------ ---------
… rows deleted …
```

```
Michael      01-JAN-01
Pat          01-JAN-01
Susan        01-JAN-01
Hermann      01-JAN-01
Shelley      01-JAN-01
William      01-JAN-01

83 rows selected.

SQL>
```

The result uses the original default value but not the updated default value. After you modify a value, you must restart the database for it to take effect.

g. Restart the instance.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289968 bytes
Variable Size             264244944 bytes
Database Buffers          226492416 bytes
Redo Buffers                8032256 bytes
Database mounted.
Database opened.
SQL>
```

h. View the result.

```
SQL> CONNECT sh
Enter password: ******
Connected.
SQL> SELECT first_name, hire_date FROM hr.employees;
```

```
FIRST_NAME   HIRE_DATE
------------ ---------
… rows deleted …
Michael      01-JUN-05
Pat          01-JUN-05
Susan        01-JUN-05
Hermann      01-JUN-05
Shelley      01-JUN-05
William      01-JUN-05

83 rows selected.

SQL>
```

Practice 14-4: Cleaning Up Redaction Policies

Overview

In this practice, you clean up the redaction policy applied on the `HR.EMPLOYEES` table.

Tasks

1. Drop the redaction policy.

```
BEGIN
  DBMS_REDACT.DROP_POLICY
  ( object_schema => 'HR',
    object_name   => 'EMPLOYEES',
    policy_name   => 'EMP_POLICY');
  END;
/
```

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> BEGIN
      DBMS_REDACT.DROP_POLICY
      ( object_schema => 'HR',
        object_name   => 'EMPLOYEES',
        policy_name   => 'EMP_POLICY');
    END;
  /
2    3    4    5    6    7
PL/SQL procedure successfully completed.

SQL> SELECT object_owner, object_name, column_name,
     function_type
     FROM redaction_columns;
  2    3
no rows selected

SQL> SELECT * FROM redaction_policies;

no rows selected

SQL>
```

2. Reset the default values of full redaction for the `VARCHAR`, `NUMBER` and `DATA` data types to the default.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> exec SYS.DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES( -
```

```
          VARCHAR_VAL => ' ')

PL/SQL procedure successfully completed.

SQL> exec SYS.DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES( -
          NUMBER_VAL => 0)

PL/SQL procedure successfully completed.

SQL> exec SYS.DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES( -
          DATE_VAL => '01-JAN-01')

PL/SQL procedure successfully completed.

SQL> SELECT varchar_value, number_value, date_value
     FROM REDACTION_VALUES_FOR_TYPE_FULL;
  2
V NUMBER_VALUE DATE_VALU
- ------------ ---------
             0 01-JAN-01

SQL>
```

3. *Question*: If you create another full redaction policy, which values are displayed for the full redacted columns of `HR.EMPLOYEES`?

   Answer: The values displayed for the full redacted columns of `HR.EMPLOYEES` use the default values that you had set in the previous practices. They are still in effect until you restart the instance.

4. Restart the instance.

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289968 bytes
Variable Size             264244944 bytes
Database Buffers          226492416 bytes
Redo Buffers                8032256 bytes
Database mounted.
Database opened.
```

```
SQL>
```

5. Check that the values for the `SALARY` and `HIRE_DATE` columns are displayed without redaction.

```
SQL> CONNECT sh
Enter password: ******
Connected.
SQL> SELECT    first_name, last_name, salary, commission_pct,
               hire_date
     FROM      hr.employees
     WHERE     department_id = 100
     OR        first_name    = 'Louise'
     ORDER BY  4 DESC;
  2    3    4    5    6
FIRST_NAME   LAST_NAME  SALARY  COMMISSION_PCT HIRE_DATE
------------ ---------- ------- -------------- ---------
John         Chen           8200                28-SEP-05
Nancy        Greenberg     12008                17-AUG-02
Daniel       Faviet         9000                16-AUG-02
Luis         Popp           6900                07-DEC-07
Ismael       Sciarra        7700                30-SEP-05
Jose Manuel  Urman          7800                07-MAR-06
Louise       Doran          7500             .3 15-DEC-05

7 rows selected.

SQL>
```

6. Drop the `LOUISE` user.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> DROP USER louise;

User dropped.

SQL> EXIT
$
```

Practices for Lesson 15: ADM and Data Masking

Chapter 15

Practices for Lesson 15: Overview

Lesson Overview

In these practices, you will create an application data model (ADM) containing `HR` and `OE` schemas information, their relationships, and sensitive columns from the `orcl` database. The ADM will then be used for masking `HR` and `OE` data in the `orcl` database.

Practice 15-1: Creating the Packages for Library Formats

Overview

In this practice, you will create the packages required for data masking to use the predefined format entries.

Tasks

1. Execute the `HR_tab_setup.sh` script to create new tables in the `HR` schema, add columns and data to the `HR.EMPLOYEES` table, and create a function named `HR.EMAIL_MASK`.

   a. Make sure you are in the `~/labs/DM` directory.

```
$ cd ~/labs/DM
$
```

   b. Ensure that your environment points to the `orcl` instance.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base remains unchanged with value /u01/app/oracle
$
```

   c. Execute the `HR_tab_setup.sh` script.

```
$ ./HR_tab_setup.sh
SQL*Plus: Release 12.1.0.1.0 Production on Tue May 28 01:03:12
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.



Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL> connect sys/oracle_4U@em12rep as sysdba
Connected.
SQL> grant execute on dbms_crypto to sysman;

Grant succeeded.

SQL> connect sys/oracle_4U@orcl as sysdba
Connected.
SQL> grant select on oe.customers to hr;

Grant succeeded.
```

```
SQL> ALTER USER dbsnmp IDENTIFIED BY oracle_4U ACCOUNT UNLOCK;

User altered.

SQL>
SQL> connect hr/oracle_4U@orcl
Connected.
SQL> update locations set state_province = city where
state_province is null;

6 rows updated.

SQL> update locations set postal_code =
trunc(dbms_random.value(300000, 400000)) where postal_code is
null;

1 row updated.

SQL>
SQL> -- masking modifications
SQL> CREATE TABLE hr.employees_backup
     AS  SELECT * FROM hr.employees;
2
Table created.

SQL>
SQL> alter table employees add (
  2           national_id varchar2(100)
  3         , street_address varchar2(40)
  4         , postal_code    varchar2(12)
  5         , city      varchar2(30)
  6         , state_province varchar2(10)
  7         , country_id     char(2)
  8  );

Table altered.

SQL>
SQL> update employees set national_id =
trunc(dbms_random.value(100,999)) || '-' ||
trunc(dbms_random.value(10,99)) || '-' ||
trunc(dbms_random.value(1000,9999)) where employee_id in (select
e.employee_id from employees e, departments d, locations l,
```

```
countries c where e.department_id = d.department_id and
d.location_id = l.location_id and l.country_id = c.country_id
and c.country_id = 'US');

44 rows updated.

SQL>
SQL> update employees set national_id =
dbms_random.string('U',2) || ' ' ||
trunc(dbms_random.value(10,99)) || ' ' ||
trunc(dbms_random.value(10,99))|| ' ' ||
trunc(dbms_random.value(10,99)) || ' ' ||
dbms_random.string('U',1) where employee_id in (select
e.employee_id from employees e, departments d, locations l,
countries c where e.department_id = d.department_id and
d.location_id = l.location_id and l.country_id = c.country_id
and c.country_id = 'UK');

35 rows updated.

SQL>
SQL> update employees set national_id =
trunc(dbms_random.value(100,999)) || '-' ||
trunc(dbms_random.value(100,999)) || '-' ||
trunc(dbms_random.value(100,999)) where employee_id in (select
e.employee_id from employees e, departments d, locations l,
countries c where e.department_id = d.department_id and
d.location_id = l.location_id and l.country_id = c.country_id
and c.country_id = 'CA');

2 rows updated.

SQL>
SQL> update employees set national_id =
trunc(dbms_random.value(10,99)) || '-' ||
trunc(dbms_random.value(10,99)) || '-' ||
trunc(dbms_random.value(10,99)) || '-' ||
trunc(dbms_random.value(10,99)) where employee_id in (select
e.employee_id from employees e, departments d, locations l,
countries c where e.department_id = d.department_id and
d.location_id = l.location_id and l.country_id = c.country_id
and c.country_id not in ( 'US', 'UK'));

3 rows updated.

SQL> update employees e set     (
  2   e.street_address, e.postal_code, e.city, e.state_province,
e.country_id) =
```

```
  3       ( select l.street_address, l.postal_code, l.city,
l.state_province, l.country_id
  4  from locations l, departments d
  5  where l.location_id = d.location_id
  6  and e.department_id = d.department_id);

83 rows updated.

SQL> create table MANAGERS (
      MGR_ID              NUMBER,
      APPROVAL_LIMIT      NUMBER,
      MGR_COST_CENTER     NUMBER);
  Table created.

SQL> insert into MANAGERS values ( 100, 1000, 1001000);

1 row created.

SQL> insert into MANAGERS values ( 107, 7000, 1077000);

1 row created.

SQL> insert into MANAGERS values ( 200, 2000, 2002000);

1 row created.

SQL> COMMIT;
Commit complete.

SQL>
SQL> create table mask_data as
  2  select cust_first_name first_name, cust_last_name
last_name,
  3        c.cust_email email,
  4   (select p.column_value as phone_numbers from table
(phone_numbers) p where rownum = 1) phone_number,
  5        c.CUST_ADDRESS.STREET_ADDRESS street_address,
  6        c.CUST_ADDRESS.CITY city,
  7        c.CUST_ADDRESS.STATE_PROVINCE state_province,
  8        c.CUST_ADDRESS.POSTAL_CODE postal_code,
  9        c.CUST_ADDRESS.country_id country_id
 10  from oe.customers c;
```

```
Table created.

SQL>
SQL> create or replace function email_mask
  2  (rid rowid, col_name varchar2, orig_value varchar2) return
varchar2
  3  is
  4        emailadd varchar2(100);
  5  begin
  6        select first_name || '.' || employee_id || '.' ||
last_name || '@anyco.com' into emailadd
  7        from HR.employees
  8        where email = orig_value;
  9        return emailadd;
 10  end;
 11  /

Function created.

SQL>
SQL> DROP TABLE hr.code;
DROP TABLE hr.code
             *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> DROP TABLE hr.pass_tab;
DROP TABLE hr.pass_tab
             *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
SQL> CREATE TABLE hr.code (pass_code NUMBER, pass_value NUMBER);

Table created.

SQL> CREATE TABLE hr.pass_tab (code NUMBER, value NUMBER);

Table created.

SQL>
SQL> INSERT INTO hr.code VALUES (1234, 12345);
```

```
1 row created.

SQL> INSERT INTO hr.code VALUES (12345, 123456);

1 row created.

SQL>
SQL> INSERT INTO hr.pass_tab VALUES (1234, 12345);

1 row created.

SQL> INSERT INTO hr.pass_tab VALUES (12345, 123456);

1 row created.

SQL>
SQL> COMMIT;

Commit complete.

SQL>
```

d. Connect as SYS to grant the execute privilege on UTL_FILE to SYSTEM.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> GRANT execute on UTL_FILE to SYSTEM;

Grant succeeded.

SQL> EXIT
$
```

2. You will use the predefined masking formats. These use functions defined in the DM_FMTLIB package. This package is automatically installed in the DBSNMP schema of your Enterprise Manager em12rep repository database. To use the predefined masking formats on the orcl target database, you must manually install the DM_FMTLIB package on the orcl database. Some of the predefined functions require the EXECUTE privilege.

   a. Locate the following scripts in your Enterprise Manager installation. Connect to the em12rep repository database.

```
$ . oraenv
ORACLE_SID = [orcl] ? em12rep
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
```

```
$
```

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL> select PLUGIN_HOME from sysman.gc_current_deployed_plugin
     where plugin_id='oracle.sysman.db'
     and destination_type='OMS';
  2    3
PLUGIN_HOME
----------------------------------------------------------------
/u01/app/oracle/product/middleware/plugins/oracle.sysman.db.oms.
plugin_12.1.0.3.0

SQL> exit
$
```

b. Log in to the `orcl` database as a user who has the privilege to create the `DM_FMTLIB` package in the `DBSNMP` schema and then execute the scripts.

```
$ . oraenv
ORACLE_SID = [em12rep] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

```
$ export
PLUGIN_HOME=/u01/app/oracle/product/middleware/plugins/oracle.sy
sman.db.oms.plugin_12.1.0.3.0
$ cd $PLUGIN_HOME/sql/db/latest/masking
$ sqlplus dbsnmp
Enter password: ******

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
```

```
SQL> set termout off
SQL> @dm_fmtlib_pkgdef.sql
SQL> @dm_fmtlib_pkgbody.sql
SQL>
```

c. To avoid an “Insufficient privileges” error message when you use the predefined functions, grant the following privileges to the `SYSTEM` user. You will connect as `SYSTEM` to create the Data Masking Definition.

```
SQL> CONNECT / AS SYSDBA
Connected.
```

```
SQL> GRANT execute ON dbms_crypto TO system;

Grant succeeded.

SQL> GRANT execute ON dbms_random TO system;

Grant succeeded.

SQL> GRANT execute ON utl_recomp TO system;

Grant succeeded.

SQL> EXIT
```

Practice 15-2: Creating an ADM

Overview

In this practice, you create an application data model (ADM) containing `HR` and `OE` schemas information and their relationships from the `orcl` database.

1. Use Enterprise Manager Cloud Control to create the `HR_OE_ADM`.

Step	Page	Action
a.	Enterprise Summary	Click the **Enterprise** tab, then click the **Quality Management** option, and then click the **Data Discovery and Modeling**.
b	Data Discovery and Modeling	Click **Create**.
c.	Create Application Data Model Application Data Model Properties: General	Name: `HR_OE_ADM` Source Database: use the magnifying glass and select `orcl`. Click **Select**. Click **Continue**.
d.	Application Data Model Properties: Schemas	Database Credentials: Click **Named**. Use `credorcl`. Click **Login**. From the Available schemas list, select `HR`, click the arrow. From the Available schemas list, select `OE`, click the arrow. Click **Continue**.
e.	Application Data Model Properties: Schedule	Job Name: `METADATA_COLLECTION_hr_oe`. Click **Submit**.
f.	Data Discovery and Modeling	The `HR_OE_ADM` appears in the list of Application Data Models. Click Refresh until the column “Most Recent Job Status” shows Succeeded.

2. View the content of the `HR_OE_ADM`.

Step	Page	Action
a.	Data Discovery and Modeling	Select `HR_OE_ADM`. Click **Edit**.
b.	Database Login	Enter credentials to login to selected database. Click **Named**. Use `credorcl`. Click **Continue**.
c.	Edit Application Data Model: `HR_OE_ADM`	In “Application and Tables” tab: Expand `HR`.

		Expand `OE`. Expand `PM`. You could add or remove applications and tables manually.
d.	Edit Application Data Model: `HR_OE_ADM`	Click “Referential Relationships” tab: Expand `HR`. Expand each table to find the relationship with another schema table. You could add or remove referential relationships manually.
e.	Edit Application Data Model: `HR_OE_ADM`	Click “Sensitive Columns” tab: There are no sensitive columns discovered yet because the Sensitive Columns Discovery job has not yet been executed. Click **Save and Return**.

Practice 15-3: Creating Sensitive Column Types

Overview

In this practice, you will create new sensitive column types by using regular expressions to define the pattern for searching column name, comment, or data.

Tasks

1. There are some predefined types. Display the predefined types.

Step	Page	Action
a.	Data Discovery and Modeling	Click **Actions**. Then click **Sensitive Column Types**.
b.	Data Discovery and Modeling: Sensitive Column Types	A list of predefined Sensitive Column Types appears. There are some predefined types such as `ISBN_13`. Move your cursor to the **`Description`** text to get some samples. This type would allow to search for columns whose name looks like ISBN.* or BOOK.* or SBN.* and whose value could be something like the following complex expression ^(ISBN(-13)?:?)?97[89]([-0-9]{14}|[0-9]{10})$. Click the **`ISBN_13`** name to get all the attributes that define the type.

Notice that the author of all predefined types is Oracle and hence these types are not editable.

2. Create a sensitive column type to search for columns whose name contains the `PASS` string. To facilitate the creation of the new type, use the **Create Like** option to retrieve existing attributes, and especially the Column Data attribute, which can use complex patterns. In this case, the Column Data is not relevant: only searching columns whose name is not secure is relevant.

Step	Page	Action
a.	Data Discovery and Modeling: Sensitive Column Types	Select `EMAIL_ID` type name. Click **Create Like**.
b.	Create Sensitive Column Type	Name: `PASS` Search Patterns: Column Name: `PASS.*`; Column Comment: `Pass column names` Column Data: Click **OK**. Click **Data Discovery and Modeling** link.

Practice 15-4: Discovering Sensitive Columns in an ADM

Overview

In this practice, you will discover all sensitive columns in the `HR_OE_ADM` ADM according to the specific criteria of sensitiveness that you defined in the previous practice.

Tasks

1. Submit a sensitive column discovery job on the `HR_OE_ADM`. The job applies the search criteria on the applications selected from the `HR_OE_ADM` ADM, in our case `HR` and or `OE` and or `PM` (according to your choice), and will identify the columns as being sensitive. The sensitive column type selected is `PASS`, which has been created for the purpose of the example.

Step	Page	Action
a.	Data Discovery and Modeling	Select `HR_OE_ADM` ADM. Click **Edit**.
b.	Database Login	If required, enter credentials to login to selected database. Click **Named**. Use `credorcl`. Click **Continue**.
c.	Edit Application Data Model: HR_OE_ADM	Click "**Sensitive Columns**" tab. Then click **Create Discovery Job…**
d.	Create Sensitive Column Discovery Job: Parameters	Select `HR` only because the tables in this schema are confidential. Select **`PASS`** Sensitive Column Type. Check **Scan Empty Tables**. Click **Continue**.
e.	Create Sensitive Column Discovery Job: Schedule	Job Name: `SENSITIVE_COLUMN_DISCOVERY_pass` Click **Submit**. Click **View Job Details**.
f.	Job Run: `SENSITIVE_COLUMN_DISCOVERY_pass`	When the status displays Succeeded, click the **Log Report** button to get the number of discovered sensitive columns. Two columns are discovered. Click **Done**. Click **Enterprise**, then **Quality Management** and then **Data Discovery and Modeling**.

2. View the sensitive columns discovered.

a.	Data Discovery and Modeling	Select `HR_OE_ADM` ADM. Click **Edit**.
b.	Database Login	Enter credentials to login to selected

		database. Click **Named**. Use `credorcl`. Click **Continue**.
c.	Edit Application Data Model: HR_OE_ADM	Click "**Sensitive Columns**" tab. Then click **Discovery Results…**
d.	Sensitive Column Discovery Results	Expand the **`PASS`** Type to view the detailed result from the discovery job.

Explanation: 2 columns are displayed. There are two columns in the `HR.CODE` table: `PASS_CODE` and `PASS_VALUE`. These two columns have been discovered because the names of the two columns match the Column Name criteria of search. But the values in the rows do not match the criteria of Column Data.

3. You decide to keep the columns whose names match the Column Name criteria.

a.	Sensitive Column Discovery Results	Select `CODE` table and `PASS_CODE` column. Click **Set Sensitive Status**. Select **Sensitive**.
b.	Sensitive Column Discovery Results	Select `CODE` table and `PASS_VALUE` column. Click **Set Sensitive Status**. Select **Sensitive**. Click **OK**.

4. In the next practice, you will use Data Masking on top of the ADM to mask sensitive data. Therefore, you will manually identify columns with sensitive data and add these columns as sensitive in the ADM. Sensitive data is defined as personally identifiable information (PII). Name, home address, phone number, national identification number, and credit card number, compensation salary and commission, and email address are considered to be PII. Office address is not considered sensitive.

a.	Edit Application Data Model: HR_OE_ADM	Click + **Add…**.
b.	Add Sensitive Columns	Application: `HR`. Click **Search**.
c.	Sensitive Column Discovery Results	Select from `EMPLOYEES` table the columns: `EMPLOYEE_ID`, `FIRST_NAME`, `LAST_NAME`, `PHONE_NUMBER`, `EMAIL`, `NATIONAL_ID`, `CITY`, `POSTAL_CODE`, `STATE_PROVINCE`, `STREET_ADDRESS`, `COMMISSION_PCT`, `SALARY`.

		Select from `MANAGERS` table the columns `MGR_ID` Click **OK**. Columns that have referential relationships are automatically added. Click **Save and Return**.

Practice 15-5: Implementing Data Masking

Overview

In this practice, you apply Data Masking to the `HR` schema and other tables which have a referential relationship with the columns masked.

Task

1. Identify the data mask specifications for the columns that need to be masked in the `HR.EMPLOYEES` table. The HR Masking policies include the following specifications for the `EMPLOYEES` table:
   - `EMPLOYEE_ID`: This is a random six-digit number that maintains uniqueness for the primary key. All foreign keys that depend on this value are also masked with the same value.
   - `FIRST_NAME`: Replace the values with a common name in North America.
   - `LAST_NAME`: Replace the values with a common surname in North America.
   - `PHONE_NUMBER`: Replace the values with a valid format for phone number in North America.
   - `EMAIL`: Replace the values with an already masked name, with a proper format but nonexistent domain.
   - `CITY`, `POSTAL_CODE`, `STATE_PROVINCE`, `STREET_ADDRESS`: Replace these values all together using the `SHUFFLE` routine.
   - `COMMISSION_PCT`: Replace the values with NULL value.
   - `SALARY`: Replace the values using the `SHUFFLE` routine.
   - `NATIONAL_ID`: Replace the values by using distinct predefined formats based on different conditions.
   - `MANAGERS.MGR_ID`: This column is not declared as a foreign key, but is dependent on `EMPLOYEES.EMPLOYEE_ID` at the application level, and should be masked the same way as the parent `EMPLOYEES.EMPLOYEE_ID` column.
2. Start creating the data masking definition.

Step	Page	Action
a.	Data Discovery and Modeling	Click the **Enterprise** tab, then click the **Quality Management** option, and then click the **Data Masking Definitions**. Then click **Create**.
b.	Data Masking Definitions Create Masking Definition	Name: **`HR Employee Mask`** Application Data Model: `HR_OE_ADM` Reference Database: `orcl` Click **OK**. The Database Login page appears with the `CREDORCL` credentials. Click **Login**. An Error message appears to warn that no columns are found in the masking definition.

		Click **Add**.
c.	Add Columns	Search: Schema: **HR** Click **Search**. All columns from the ADM sensitive columns list are displayed.
d.	Add Columns	Select the **EMPLOYEE_ID** column of the **EMPLOYEES** table. Click **Define Format And Add**.
e.	Define Column Mask	An Information message appears to provide the list of foreign key columns dependent on **EMPLOYEES.EMPLOYEE_ID** that will be masked the same way as the parent column. Select **Random Numbers** from the Format Entry list. Click **Add**.
f.	Define Column Mask	Enter the following information: Start value: **100000** End value: **999999** Click **Refresh** in the Sample column to see samples of the masked data. If you get an error, you encounter a known bug that will be fixed in the next release. This does not prevent data masking. Click **OK**.
g.	Create Masking Definition Columns section	Five other foreign key columns are automatically masked. But the **MANAGERS.MGR_ID** column needs to be masked in the same way as the parent column. But there is no referential relationship with the parent column. Add the dependent column manually so that it benefits the same mask. In the Columns section, click **Add**.
h.	Add Columns	Search: Schema: **HR** Click **Search**. All columns from the ADM sensitive columns list are displayed. Select the **MGR_ID** column of the **MANAGERS** table. Click **Define Format And Add**.
i.	Define Column Mask	Enter the following details: Select **Random Numbers** from the Format Entry list. Click **Add**.

j.	Define Column Mask	Enter the following information: Start value: **`100000`** End value: **`999999`** Click **OK**.
k.	Create Masking Definition	Click **OK**.

Create a mask format, store in the format library, and reuse it for the **`MANAGERS.MGR_ID`** column.

3. Create the masking formats for the `EMPLOYEES.FIRST_NAME` and `EMPLOYEES.LAST_NAME` columns by using the `HR.MASK_DATA` table as source of masking data. These steps illustrate how you would use a data table from a commercial data provider to mask confidential data such as names.

Step	Page	Action
a.	Data Masking Definitions	Click the **Enterprise** tab, then click the **Quality Management** option, and then click the **Data Masking Formats** link.
b.	Format Library	Click **Create**.
c.	Create Format	Enter the following information: Name: **`Anglo-American First Name`** Sensitive Column Type: **`UNDEFINED`** Description: **`Masking format for first name`** Select **Table Column** in the Format Entries list and click **Go**.
d.	Create Format Table Column	Enter the following information: Table Name: **`hr.mask_data`** Column Name: **`first_name`** Note that the message "Unable to generate Sample Masked Data. The Use Defined Function, Post Processing Function, or the Table Column that this format relies on may not exist in the repository. Make sure that they exist in the database to be masked." is to explain that the Defined Function, Post Processing Function, or the Table Column used in the format creation cannot produce any sample data for the moment because these functions and table columns do not exist in the `em12rep` repository database. But they do exist in the `orcl` database. Click **OK**.
e.	Create Format	Click **OK**.
f.	Format Library	Confirmation message is displayed. Click **Create**.
g.	Create Format	Enter the following information: Name: **`Anglo-American Last Name`**

		Sensitive Column Type: **`UNDEFINED`** Description: **`Masking format for last name`** Select **Table Column** in the list and click **Go**.
h.	Create Format	Enter the following information: Table Name: **`hr.mask_data`** Column Name: **`last_name`** Note that the message "Unable to generate Sample Masked Data. The Use Defined Function, Post Processing Function, or the Table Column that this format relies on may not exist in the repository. Make sure that they exist in the database to be masked." is to explain that the Defined Function, Post Processing Function, or the Table Column used in the format creation cannot produce any sample data for the moment because these functions and table columns do not exist in the `em12rep` repository database. But they do exist in the `orcl` database. Click **OK**.
i.	Create Format	Click **OK**.
j.	Format Library	Confirmation message is displayed.

4. Create the masking definitions for the `EMPLOYEES.FIRST_NAME` and `EMPLOYEES.LAST_NAME` columns. Use the mask formats that you defined in the previous step.

Step	Page	Action
a.	Format Library	In the **Enterprise** tab, click **Quality Management**, then click **Data Masking Definitions**.
b.	Data Masking Definitions	Select **HR Employee Mask**. Click **Edit**.
c.	Edit Masking Definition: HR Employee Mask	Click **Add**.
d.	Add Columns	Enter the following information: Schema: **`HR`** Table Name: **`Employees`** Click **Search**.
e.	Add Columns	Select the **`FIRST_NAME`** and **`LAST_NAME`** columns. Click **Add**.
f.	Edit Masking Definition: HR Employee Mask	Click the **Format icon** in the `FIRST_NAME` row.
g.	Define Column mask	Click **Import Format**.
h.	Import Format	Select **Anglo-American First Name**. Click **Import**.
i.	Define Column Mask	In the Sample column, click refresh on the curved arrow to view a masked data sample. Click **OK**.

j.	Edit Masking Definition: HR Employee Mask	Click the **Format icon** in the `LAST_NAME` row.
k.	Define Column Mask	Click **Import Format**.
l.	Import Format	Select **Anglo-American Last Name**. Click **Import**.
m.	Define Column Mask	Click **OK**.
n.	Data Masking Definitions	Shows 4 columns masked.

5. Add the `EMPLOYEES.SALARY` column to the HR Employee Mask masking definition and specify the Shuffle mask format.

Step	Page	Action
a.	Edit Masking Definition: HR Employee Mask	Click **Add**.
b.	Add Columns	Enter the following information: Schema: **`hr`** Table Name: **`employees`** Click **Search**.
c.	Add Columns	Select the **`SALARY`** column. Click **Define Format And Add**.
d.	Define Column Mask	Select **Shuffle** from the Format Entry list. Click **Add**.
e.	Define Column Mask	Click **OK**.

6. Add the `EMPLOYEES.COMMISSION_PCT` column to the HR Employee Mask masking definition and specify the `Null Value` mask format.

Step	Page	Action
a.	Edit Masking Definition: HR Employee Mask	Click **Add**.
b.	Add Columns	Enter the following information: Schema: **`hr`** Table Name: **`employees`** Click **Search**.
c.	Add Columns	Select the **`COMMISSION_PCT`** column. Click **Define Format And Add**.
d.	Define Column Mask	Select **Null Value** from the Format Entry list. Click **Add**.
e.	Define Column Mask	Click **OK**.

7. Feel free to add the `EMPLOYEES.PHONE_NUMBER` column to the HR Employee Mask masking definition. For the purpose of this practice, specify the `USA Phone Number`

`Formatted` mask format from the Format Library. The sensitive column type is not `UNDEFINED`. Therefore, erase `UNDEFINED` in the sensitive column type field. Refer to step 4 for details about using a mask format from the Format Library.

8. Feel free to add the `EMPLOYEES.EMAIL` column to the HR Employee Mask masking definition. The mask for this column requires a post-processing function. Use the `HR.HR_MASK_EMAIL` function. The `HR_MASK_EMAIL` function retrieves the `FIRST_NAME`, `LAST_NAME`, and `EMPLOYEE_ID`, and constructs a properly formatted email address. The post-processing function requires that a masking format be applied first. Use the Preserve masking format.

Step	Page	Action
a.	Edit Masking Definition: HR Employee Mask	Click **Add** in the Columns section.
b.	Add Columns	Enter the following information: Schema: **`hr`** Table: **`employees`** Click **Search**.
c.	Add Columns	Select **`EMAIL`**. Click **Define Format And Add**.
d.	Define Column Mask	Select **Preserve Original Data** in the Format Entry list. Click **Add**.
e.	Define Column Mask	Select **Post-Processing Function** in the Format Entry list. Click **Add**.
f.	Define Column Mask	Enter the following information: Function Name: **`HR.EMAIL_MASK`** Click **OK**.

9. Feel free to implement condition-based masking for the `NATIONAL_ID` column. Configure the masking so that the `NATIONAL_ID` column is masked with the National Insurance Number Formatted format for UK employees and Social Security Number Formatted for US employees. The `NATIONAL_ID` column for employees from other countries does not need to be masked.

Step	Page	Action
a.	Edit Masking Definition: HR Employee Mask	Click **Add** in the Columns section.
b.	Add Columns	Enter the following information: Schema: **`hr`** Table: **`employees`** Column: **`national_id`** Click **Search**.
c.	Add Columns	Select **`NATIONAL_ID`**. Click **Define Format And Add**.
d.	Define Column Mask	Click **Add Condition**.

e.	Define Column Mask	Enter the following SQL query in the Condition field: **`country_id = 'UK'`** Click **Import Format**.
f.	Import Format	Sensitive Column Type: erase `UNDEFINED`. Click **Search**. Select **National Insurance Number Formatted**. Click **Import**.
g.	Define Column Mask	Click **Add Condition**.
h.	Define Column Mask	Enter the following SQL query in the Condition field: **`country_id = 'US'`** Click **Import Format**.
i.	Import Format	Sensitive Column Type: erase `UNDEFINED`. Select **Social Security Number Formatted**. Click **Import**.
j.	Define Column Mask	Select **Default Condition**. (Click the radio button) Select **Preserve Original Data** in the Format Entry list. Click **Add**.
k.	Define Column Mask	Click **OK**.

10. Feel free to add the `EMPLOYEES.CITY`, `EMPLOYEES.POSTAL_CODE`, `EMPLOYEES.STATE_PROVINCE`, `EMPLOYEES.STREET_ADDRESS` columns to the HR Employee Mask masking definition as a group and specify the Shuffle mask format. These columns need to be masked all together.

Step	Page	Action
a.	Edit Masking Definition: HR Employee Mask	Click **Add**.
b.	Add Columns	Enter the following information: Schema: **`hr`** Table Name: **`employees`** Click **Search**.
c.	Add Columns	Select the `CITY`, `POSTAL_CODE`, `STATE_PROVINCE`, `STREET_ADDRESS` columns. Check the “Mask selected columns as a group.” Click **Define Format And Add**.
d.	Define Group Mask	Select **Shuffle** from the Format Type list.
e.	Define Column	Click **OK**.

	Mask	
f.	Masking Definition: Define Format	The four columns in the group are listed with a Column Group number 1. Click **OK**.

Practice 15-6: Applying a Masking Definition

Overview

In this practice, you will use the masking definition to mask data in `HR` and `OE` tables.

Assumptions

The previous practices 15-1, 15-2, 15-3, 15-4, and 15-5 successfully completed.

Tasks

1. Before performing the masking operation, query the `HR.EMPLOYEES` table to view the data before masking. Invoke SQL*Plus and connect as the `HR` user. The output shows a query for the employees in department 30. You may want to execute additional queries before masking the data to view the unmasked data.

```
$ sqlplus hr/oracle_4U

Connected to:
With the Partitioning, Automatic Storage Management, Oracle
Label Security, OLAP, Data Mining, Oracle Database Vault and
Real Application Testing options

SQL> select employee_id, last_name, salary, email, national_id
     from   hr.employees
     where  department_id = 30;
  2    3

(output formatted for clarity)

EMPLOYEE_ID LAST_NAME       SALARY  EMAIL     NATIONAL_ID
----------- --------------- ------- --------- -----------
        114 Raphaely          11000 DRAPHEAL  664-31-8359
        115 Khoo               3100 AKHOO     828-52-1644

2 rows selected.

SQL>
```

2. Return to Enterprise Manager Cloud Control to generate the data-masking script and schedule the data masking job.

Step	Page	Action
a.	Data Masking Definitions	Select HR Employee Mask. Click **Generate Script**.
b.	Schedule Script Generation Job: HR Employee Mask	Use Named Credentials: Named: `credorcl` Select Immediately in the Start section.

		Click **Submit**.
c.	Data Masking Definitions	Message: `Job Submitted Successfully` Click **View Job Details**.
d.	Job Run: MASKING_JOB_*NNN*	Refresh until the Status is Succeeded.

By iewing the job details or clicking the **Impact Report**, you get the following report:

```
Validation of masking script completed.
Generating script.....
Script generation succeeded with the following messages
.............................................................
.............................................................
INFORMATION  TABLESPACE EXAMPLE Sufficient free space in
Tablespace EXAMPLE.
Starting Freespace with automatic extension: 33223MB.
Ending Freespace: 33223MB.
Lowest Freespace: 33223MB.
.............................................................
.............................................................
INFORMATION  USER HR Sufficient tablespace quota for User HR.
.............................................................
.............................................................
INFORMATION  USER OE Sufficient tablespace quota for User OE.
.............................................................
.............................................................
INFORMATION  USER SYS Sufficient tablespace quota for User SYS.
.............................................................
.............................................................
INFORMATION  TABLESPACE SYSTEM Sufficient free space in
Tablespace SYSTEM.
Starting Freespace with automatic extension: 32759MB.
Ending Freespace: 32758MB.
Lowest Freespace: 32758MB.
.............................................................
.............................................................
INFORMATION  TABLESPACE TEMP Sufficient free space in Tablespace
TEMP.
Starting Freespace with automatic extension: 33464MB.
Ending Freespace: 33463MB.
Lowest Freespace: 33463MB.
.............................................................
.............................................................
INFORMATION  TABLESPACE USERS Sufficient free space in
Tablespace USERS.
```

```
Starting Freespace with automatic extension: 33552MB.
Ending Freespace: 33552MB.
Lowest Freespace: 33552MB.
```

3. Return to SQL*Plus and check that the columns are not yet masked. Query the same rows you looked at in step 1.

```
SQL> select employee_id, last_name, salary, email, national_id
     from   hr.employees
     where  department_id = 30;

(output formatted for clarity and ease of comparison)

EMPLOYEE_ID LAST_NAME       SALARY  EMAIL         NATIONAL_ID
----------- --------------- ------- ------------- -----------
        114 Raphaely          11000 DRAPHEAL     664-31-8359
        115 Khoo               3100 AKHOO        828-52-1644

2 rows selected.

SQL>
```

4. Execute the script against the database to mask data.

Step	Page	Action
a.	Job Activity: Job Run MASKING_JOB_*NNN*	Click **Enterprise**, then **Quality Management** and **Data Masking Definitions**.
b.	Data Masking Definitions	Select HR Employee Mask. Click **Schedule Job**.
c.	Schedule Data Masking Job: HR Employee Mask Host Credentials section	Click **New**. UserName: `oracle` Password: `oracle` Confirm Password: `oracle` Save As: `CREDOS` Click **Test**.
d.	Schedule Data Masking Job: HR Employee Mask Database Credentials section	Click **Named**.
e.	Schedule Data Masking Job: HR Employee Mask	Click **Submit**.
f.	Data Masking Definitions	Message: `Job Submitted Successfully` Click **View Job Details**.
g.	Job Run:	Refresh until the Status is Succeeded.

	MASKING_JOB_*NNN*	

5. Return to SQL*Plus and check that the columns are now masked. Query the same rows you looked at in steps 1. The value are different and you will get different results than the ones below. Using random functions, the results are always different.

```
$ sqlplus HR/oracle_4U

Connected to:
With the Partitioning, Automatic Storage Management, Oracle
Label Security, OLAP, Data Mining, Oracle Database Vault and
Real Application Testing options

SQL> select employee_id, last_name, salary, email, national_id
     from   hr.employees
     where  department_id = 30;

(output formatted for clarity and ease of comparison)

EMPLOYEE_ID LAST_NAME   SALARY EMAIL
NATIONAL_ID
----------- --------- -------- --------------------------- -----
------
     571018 Goodman      24000 Guy.118.Himuro@anyco.com    225-
98-0011
     571016 Heard        13000 Shelli.116.Baida@anyco.com  100-
07-7080

2 rows selected.

SQL>
```

6. Restore the content of the `HR.EMPLOYEES` table to what it was originally.

```
SQL> DROP TABLE hr.employees CASCADE CONSTRAINTS;

Table dropped.

SQL> ALTER TABLE hr.employees_backup RENAME TO employees;

Table altered.

SQL> EXIT
$
```

Practices for Lesson 16: Transparent Sensitive Data Protection

Chapter 16

Practices for Lesson 16: Overview

Lesson Overview

In these practices, you will use TSDP to define sensitive column types and configure a TSDP policy to protect the sensitive column data matching these sensitive column types using a VPD policy. Then you will use the predefined TSDP `REDACT_AUDIT` policy to protect other sensitive columns.

Practice 16-1: Implementing a TSDP Policy

Overview

In this practice, you create a TSDP policy to protect the sensitive column data matching sensitive column types in the `orcl` database. Then you will configure the TSDP to protect `HR` and `OE` sensitive columns using a VPD policy.

Tasks

1. Create tables with sensitive columns. Use the `$HOME/labs/TSDP/create_tables.sql` script.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production

SQL> @$HOME/labs/TSDP/create_tables.sql
SQL> drop table oe.customers_info;
drop table oe.customers_info
             *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
SQL> CREATE TABLE oe.customers_info (
  2   CUSTOMER_ID               NUMBER(6) NOT NULL,
  3   CUST_FIRST_NAME                 VARCHAR2(20),
  4   CUST_LAST_NAME            VARCHAR2(20),
  5   CCN_TYPE            VARCHAR2(6),
  6   CCN            NUMBER(30),
  7   SSN            NUMBER(9));

Table created.

SQL>
SQL> INSERT INTO oe.customers_info VALUES (
  2           110,
'Adam','X','CARD',5105105105105100,987654320);
```

```
1 row created.

SQL> INSERT INTO oe.customers_info VALUES (
  2           109,
'Christian','Y','CARD',6011111111111117,987654321);

1 row created.

SQL> INSERT INTO oe.customers_info VALUES (
  2  108, 'Meenakshi','W','AMEX',378282246310005,987654322);

1 row created.

SQL> INSERT INTO oe.customers_info VALUES (
  2           107,
'Peter','A','CARD',6011000000000004,987654323);

1 row created.

SQL> INSERT INTO oe.customers_info VALUES (
  2           106,
'Peter','B','VISA',4111111111111111,987654324);

1 row created.

SQL> INSERT INTO oe.customers_info VALUES (
  2           105,
'Peter','C','MASTER',5105105105105100,987654325);

1 row created.

SQL> INSERT INTO oe.customers_info VALUES (
  2           104,
'Harrison','D','VISA',4222222222222,987654326);

1 row created.

SQL> INSERT INTO oe.customers_info VALUES (
  2           103,
'Manisha','E','AMEX',343434343434343,987654327);

1 row created.
```

```
SQL> INSERT INTO oe.customers_info VALUES (
  2            102,
'Harrison','F','CARD',6011000990139424,987654328);

1 row created.

SQL> INSERT INTO oe.customers_info VALUES (
  2            101,
'Constantin','G','MASTER',5111111111111118,987654329);

1 row created.

SQL>
SQL> COMMIT;

Commit complete.

SQL>
```

2. Connect as SYSDBA to grant the SEC user the execute privilege on the SYS.DBMS_TSDP_MANAGE and SYS.DBMS_TSDP_PROTECT packages.

```
SQL> grant execute ON DBMS_TSDP_MANAGE to SEC;

Grant succeeded.

SQL> grant execute ON DBMS_TSDP_PROTECT to SEC;

Grant succeeded.

SQL>
```

3. Because the TSDP policy will be set for VPD, grant the SEC user the execute privilege on the SYS.DBMS_RLS package if not already done for a previous practice.

```
SQL> grant execute on DBMS_RLS to SEC;

Grant succeeded.

SQL>
```

4. Define two sensitive column types. Create the 'Sensitive_Numbers' type. Columns like CCN (credit card number) or SSN (social security number) of OE.CUSTOMERS_INFO table will match this sensitive type. Create the 'Income' type. Columns like SALARY or COMMISSION_PCT of HR.EMPLOYEES table will match this sensitive type.

   Remark: Do not use blank spaces in the type name.

```
SQL> CONNECT sec
```

```
Enter password: ******
Connected.
SQL> exec DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE (-
     sensitive_type   => 'Sensitive_Numbers')
> BEGIN DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE (    sensitive_type
=> 'Sensitive_Numbers'); END;

*
ERROR at line 1:
ORA-45610: Sensitive type Sensitive_Numbers does not exist.
ORA-06512: at "SYS.DBMS_TSDP_MANAGE", line 348
ORA-06512: at line 1

SQL> exec DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE (-
     sensitive_type   => 'Income')
> BEGIN DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE (    sensitive_type
=> 'Income'); END;

*
ERROR at line 1:
ORA-45610: Sensitive type Income does not exist.
ORA-06512: at "SYS.DBMS_TSDP_MANAGE", line 348
ORA-06512: at line 1

SQL> exec DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (-
     sensitive_type   => 'Sensitive_Numbers',-
     user_comment     => 'Type for credit card numbers, -
           social security numbers using a number data type' )
> >
PL/SQL procedure successfully completed.

SQL> exec DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (-
     sensitive_type   => 'Income',-
     user_comment     => 'Type for salary, commission' )
> >
PL/SQL procedure successfully completed.

SQL>
```

5. Display the list of sensitive column types.

```
SQL> COL name FORMAT A18
SQL> COL source_type FORMAT A3
SQL> SELECT name, user_comment, source_type
```

```
     FROM   dba_sensitive_column_types
     WHERE  source_type='DB';
  2    3
NAME
------------------
USER_COMMENT
-----------------------------------------------------------------
SOU
---
Sensitive_Numbers
Type for credit card numbers, social security numbers using a
number data type
DB

Income
Type for salary, commission
DB

SQL>
```

6. You identified the list of sensitive columns:
   - OE.CUSTOMERS_INFO.CCN
   - OE.CUSTOMERS_INFO.SSN
   - HR.EMPLOYEES.SALARY
   - HR.EMPLOYEES.COMMISSION_PCT

   Associate OE.CUSTOMERS_INFO.CCN and OE.CUSTOMERS_INFO.SSN columns with the 'Sensitive_Numbers' sensitive type.

   Associate HR.EMPLOYEES.SALARY and HR.EMPLOYEES.COMMISSION_PCT columns with the 'Income' sensitive type.

```
SQL> exec DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN(-
       schema_name        => 'OE', -
       table_name         => 'CUSTOMERS_INFO', -
       column_name        => 'CCN', -
       sensitive_type     => 'Sensitive_Numbers')
> > > >

PL/SQL procedure successfully completed.

SQL> exec DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN(-
       schema_name        => 'OE', -
       table_name         => 'CUSTOMERS_INFO', -
       column_name        => 'SSN', -
       sensitive_type     => 'Sensitive_Numbers')
```

```
> > > >

PL/SQL procedure successfully completed.

SQL> exec DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN(-
       schema_name        => 'HR', -
       table_name         => 'EMPLOYEES', -
       column_name        => 'SALARY', -
       sensitive_type     => 'Income')
> > > >

PL/SQL procedure successfully completed.

SQL> exec DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN(-
       schema_name        => 'HR', -
       table_name         => 'EMPLOYEES', -
       column_name        => 'COMMISSION_PCT', -
       sensitive_type     => 'Income')
> > > >

PL/SQL procedure successfully completed.
```

7. Display the list of identified columns as confidential.

```
SQL> COL SCHEMA_NAME FORMAT A10
SQL> COL SENSITIVE_TYPE FORMAT A20
SQL> COL TABLE_NAME FORMAT A16
SQL> COL COLUMN_NAME FORMAT A14
SQL> SELECT SCHEMA_NAME, TABLE_NAME,
            COLUMN_NAME, SENSITIVE_TYPE
     FROM   dba_sensitive_data;
2    3
SCHEMA_NAM TABLE_NAME            COLUMN_NAME     SENSITIVE_TYPE
---------- ---------------- --------------  --------------------
HR         EMPLOYEES         SALARY          Income
HR         EMPLOYEES         COMMISSION_PCT  Income
OE         CUSTOMERS_INFO    CCN             Sensitive_Numbers
OE         CUSTOMERS_INFO    SSN             Sensitive_Numbers

SQL>
```

8. Create the TSDP policy. You can configure it for the Virtual Private Database or Oracle Data Redaction settings that you want to use, and then apply these settings to a TSDP policy.

Because you are using Oracle Virtual Private Database for your policy, a specification of the VPD settings must be set. You can also specify conditions to test when the policy is enabled. For example, the data type of the column which should be satisfied before the policy can be enabled.

a. Create a user that exists in the values of the `HR.EMPLOYEES.FIRST_NAME` column as well as in `OE.CUSTOMERS_INFO` table. And also grant `SCOTT` the same privileges for the further tests.

```
SQL> CREATE USER peter IDENTIFIED BY oracle_4U;

User created.

SQL> GRANT create session TO peter;

Grant succeeded.

SQL> GRANT select ON hr.employees TO peter, scott;

Grant succeeded.

SQL> GRANT select ON oe.customers_info TO peter, scott;

Grant succeeded.

SQL>
```

b. Create the VPD function that TSDP will associate with the VPD policy that will be automatically created when you enable the TSDP policy.

```
SQL> CREATE OR REPLACE FUNCTION vpd_tsdp_function (
       v_schema IN VARCHAR2,  v_objname IN VARCHAR2)
     RETURN VARCHAR2 AS
      BEGIN
       RETURN 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') =
''PETER''';
END vpd_tsdp_function;
/
  2    3    4    5    6    7
Function created.

SQL>
```

c. Create the TSDP policy. You must at least define the name of the VPD function as one of the `VPD_FEATURE_OPTIONS`. All other options have default values.
When the TSDP policy is enabled, the VPD policy that is automatically created will have its `sec_relevant_cols` parameter (of `DBMS_RLS.ADD_POLICY`) set to the name of the sensitive column on which TSDP enables the VPD policy. If you had not

set the `sec_relevant_cols_opt` parameter, then TSDP would not have used the `DBMS_RLS.ADD_POLICY sec_relevant_cols_opt` parameter.

```
SQL> DECLARE
  vpd_feature_options SYS.DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
  policy_conditions SYS.DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
  vpd_feature_options ('policy_function') :=
                                          'vpd_tsdp_function';
  vpd_feature_options ('sec_relevant_cols_opt') :=
                                          'DBMS_RLS.ALL_ROWS';
  vpd_feature_options ('statement_types') := 'SELECT';
  policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
  DBMS_TSDP_PROTECT.ADD_POLICY('tsdp_vpd',
                               DBMS_TSDP_PROTECT.VPD,
                               vpd_feature_options,
                               policy_conditions);
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16
PL/SQL procedure successfully completed.

SQL>
```

d. Display all information related to the new TSDP policy, like parameters, conditions, features.

```
SQL> COL POLICY_NAME FORMAT A16
SQL> COL PROPERTY FORMAT A12
SQL> COL VALUE FORMAT A12
SQL> SELECT * FROM DBA_TSDP_POLICY_FEATURE;

POLICY_NAME      SECURITY_FEA
---------------- ------------
REDACT_AUDIT     REDACT_AUDIT
tsdp_vpd         VPD

SQL> SELECT * FROM DBA_TSDP_POLICY_CONDITION;

POLICY_NAME      SUB_POLICY PROPERTY     VALUE
---------------- ---------- ------------ ------------
tsdp_vpd                 1     DATATYPE     NUMBER

SQL> col parameter format A24
```

```
SQL> col value format A20
SQL> SELECT policy_name, parameter, value, default_option
     FROM DBA_TSDP_POLICY_PARAMETER;
  2
POLICY_NAME  PARAMETER             VALUE                DEFAU
------------ -------------------- -------------------- -----
REDACT_AUDIT ORA$TSDP_DEFAULT     ORA$TSDP_DEFAULT     TRUE
tsdp_vpd     policy_function      vpd_tsdp_function    FALSE
tsdp_vpd     sec_relevant_cols_opt DBMS_RLS.ALL_ROWS   FALSE
tsdp_vpd     statement_types      SELECT               FALSE

SQL>
```

9. Associate the TSDP policy with the '`Sensitive_Numbers`' and '`Income`' sensitive types.

```
SQL> exec DBMS_TSDP_PROTECT.ASSOCIATE_POLICY( -
          policy_name        => 'tsdp_vpd', -
          sensitive_type     => 'Sensitive_Numbers', -
          associate          => TRUE)
> > >
PL/SQL procedure successfully completed.

SQL> exec DBMS_TSDP_PROTECT.ASSOCIATE_POLICY( -
          policy_name        => 'tsdp_vpd', -
          sensitive_type     => 'Income', -
          associate          => TRUE)
> > >
PL/SQL procedure successfully completed.

SQL> col sensitive_type format A30
SQL> select * from DBA_TSDP_POLICY_TYPE;

POLICY_NAME      SENSITIVE_TYPE
---------------- ------------------------------
REDACT_AUDIT     Sensitive_Numbers
REDACT_AUDIT     Income
tsdp_vpd         Sensitive_Numbers
tsdp_vpd         Income

SQL>
```

10. Enable the TSDP policy protections at the sensitive type level.

```
SQL> exec DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE( -
           sensitive_type    => 'Sensitive_Numbers')
>
```

```
PL/SQL procedure successfully completed.

SQL> exec DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE( -
           sensitive_type    => 'Income')
>
PL/SQL procedure successfully completed.

SQL>
```

11. Display the protected columns.

```
SQL> COL SCHEMA_NAME FORMAT A4
SQL> COL TABLE_NAME FORMAT A14
SQL> COL COLUMN_NAME FORMAT A14
SQL> COL TSDP_POLICY FORMAT A14
SQL> COL SECURITY_FEATURE FORMAT A14
SQL> COL SECURITY_FEATURE_POLICY FORMAT A40
SQL> SELECT schema_name, table_name,
            column_name, tsdp_policy,
            security_feature, SECURITY_FEATURE_POLICY
     FROM   DBA_TSDP_POLICY_PROTECTION;
  2   3   4   5
SCHE TABLE_NAME     COLUMN_NAME     TSDP_POLICY   SECU
---- -------------- --------------  -------------- -------------
SECURITY_FEATURE_POLICY
----------------------------------------------------------------
OE   CUSTOMERS_INFO   SSN            REDACT_AUDIT  REDACT_AUDIT
REDACT_AUDIT_POLICY

HR   EMPLOYEES        COMMISSION_PCT REDACT_AUDIT  REDACT_AUDIT
REDACT_AUDIT_POLICY

OE   CUSTOMERS_INFO   CCN            REDACT_AUDIT  REDACT_AUDIT
REDACT_AUDIT_POLICY

HR   EMPLOYEES        SALARY         REDACT_AUDIT  REDACT_AUDIT
REDACT_AUDIT_POLICY

HR   EMPLOYEES       COMMISSION_PCT tsdp_vpd       VPD
ORA$VPD_185+uGj0bZ3Q61t4M8VzcA

HR   EMPLOYEES       SALARY         tsdp_vpd       VPD
ORA$VPD_OB41Bady5I5jx4iYxsjT6w
```

```
OE   CUSTOMERS_INFO  SSN            tsdp_vpd       VPD
ORA$VPD_DeR0fkLChcNgHhUWVV6p/g

OE   CUSTOMERS_INFO  CCN            tsdp_vpd       VPD
ORA$VPD_sijJYeNDqu61N4q8RQ3+QA

SQL>
```

12. Display VPD policies created.

```
SQL> set pages 100
SQL> COL function FORMAT A18
SQL> COL pf_owner FORMAT A4
SQL> COL package FORMAT A4
SQL> COL policy_group FORMAT A12
SQL> COL policy_name FORMAT A32
SQL> COL object_owner FORMAT A12
SQL> COL object_name FORMAT A20
SQL> select * from dba_policies
     where function='VPD_TSDP_FUNCTION';
  2
OBJECT_OWNER
-----------------------------------------------------------------
OBJECT_NAME
-----------------------------------------------------------------
POLICY_GROUP
-----------------------------------------------------------------
POLICY_NAME
-----------------------------------------------------------------
PF_OWNER
-----------------------------------------------------------------
PACKAGE
-----------------------------------------------------------------
FUNCTION
-----------------------------------------------------------------
SEL INS UPD DEL IDX CHK ENA STA POLICY_TYPE              LON
--- --- --- --- --- --- --- --- ------------------------ ---
HR
EMPLOYEES
SYS_DEFAULT
ORA$VPD_185+uGj0bZ3Q61t4M8VzcA
SEC

VPD_TSDP_FUNCTION
```

```
YES NO    NO  NO     NO  NO     YES NO     DYNAMIC          NO

HR
EMPLOYEES
SYS_DEFAULT
ORA$VPD_OB41Bady5I5jx4iYxsjT6w
SEC

VPD_TSDP_FUNCTION
YES NO    NO  NO     NO  NO     YES NO     DYNAMIC          NO

OE
CUSTOMERS_INFO
SYS_DEFAULT
ORA$VPD_DeR0fkLChcNgHhUWVV6p/g
SEC

VPD_TSDP_FUNCTION
YES NO    NO  NO     NO  NO     YES NO     DYNAMIC          NO

OE
CUSTOMERS_INFO
SYS_DEFAULT
ORA$VPD_sijJYeNDqu61N4q8RQ3+QA
SEC

VPD_TSDP_FUNCTION
YES NO    NO  NO     NO  NO     YES NO     DYNAMIC          NO


SQL>
```

13. Test if the VPD and TSDP protect the four columns identified as sensitive.
    a. Connect as PETER.

```
SQL> CONNECT peter
Enter password: ******
Connected.
SQL> COL ssn  FORMAT 99999999999
SQL> COL ccn FORMAT 999999999999999999
SQL> SELECT ccn, ssn FROM oe.customers_info;

                CCN          SSN
------------------- ------------
```

```
   5105105105105100    987654320
   6011111111111117    987654321
    378282246310005    987654322
   6011000000000004    987654323
   4111111111111111    987654324
   5105105105105100    987654325
      4222222222222    987654326
    343434343434343    987654327
   6011000990139424    987654328
   5111111111111118    987654329

10 rows selected.

SQL> SELECT last_name, salary, commission_pct
     FROM   hr.employees;
   2
LAST_NAME                    SALARY  COMMISSION_PCT
------------------------- --------- --------------
… rows deleted …
Bloom                         10000            .2
Kumar                          6100            .1
Livingston                     8400            .2
Taylor                         8600
Smith                          8000

83 rows selected.

SQL>
```

Notice that Peter can see all rows and all values of the sensitive type columns, `CCN` and `SSN` columns from `OE.CUSTOMERS_INFO` and `SALARY` and `COMMISSION_PCT` columns from `HR.EMPLOYEES`.

b. Connect as `SCOTT`.

```
SQL> CONNECT scott
Enter password: ******
Connected.
SQL> SELECT cust_last_name, ccn, ssn FROM oe.customers_info;

CUST_LAST_NAME                       CCN           SSN
-------------------- ------------------- ------------
X
Y
W
```

```
A
B
C
D
E
F
G

10 rows selected.

SQL> SELECT last_name, salary, commission_pct
     FROM hr.employees;
  2
LAST_NAME                     SALARY COMMISSION_PCT
------------------------- ---------- --------------
… rows deleted …

Kumar
Livingston
Taylor
Sullivan
Sarchand
Cabrio
Dilly
Perkins
Everett
Walsh

83 rows selected.

SQL>
```

Notice that Scott cannot see any value of any the sensitive type columns from both `OE.CUSTOMERS_INFO` and `HR.EMPLOYEES` tables.

Practice 16-2: Using REDACT_ AUDIT Policy

Overview

In this practice, you will display or mask the bind variable values used in statements where sensitive columns are protected by TSPD policy. You will disable and re-enable the predefined `REDACT_AUDIT` policy to display or mask the values.

Tasks

1. Audit any `SELECT` operation on `OE.CUSTOMERS_INFO` table executed by `PETER`. The `AUDIT` command uses a new syntax that will be covered in the lesson 21 and practice 21-2.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> create audit policy pol_sel actions select on
oe.customers_info;

Audit policy created.

SQL> audit policy pol_sel by peter;

Audit succeeded.

SQL>
```

2. Verify that there is a `REDACT_AUDIT` policy associated with the `SSN` sensitive column.

```
SQL> SELECT schema_name, table_name,
            column_name, tsdp_policy,
            security_feature, SECURITY_FEATURE_POLICY
     FROM   DBA_TSDP_POLICY_PROTECTION
     WHERE  tsdp_policy <> 'tsdp_vpd';
  2   3   4   5
SCHE TABLE_NAME     COLUMN_NAME    TSDP_POLICY    SECU
---- -------------- -------------- -------------- -------------
SECURITY_FEATURE_POLICY
----------------------------------------------------------------
OE   CUSTOMERS_INFO  SSN            REDACT_AUDIT  REDACT_AUDIT
REDACT_AUDIT_POLICY

HR   EMPLOYEES       COMMISSION_PCT REDACT_AUDIT  REDACT_AUDIT
REDACT_AUDIT_POLICY

OE   CUSTOMERS_INFO  CCN            REDACT_AUDIT  REDACT_AUDIT
REDACT_AUDIT_POLICY
```

```
HR   EMPLOYEES        SALARY         REDACT_AUDIT  REDACT_AUDIT
REDACT_AUDIT_POLICY

SQL>
```

3. The REDACT_AUDIT policy masks bind values of bind variables that are considered to be sensitive or "associated" with sensitive columns with an '*' value. In comparison conditions, when a sensitive column and a bind variable appear in the expressions that are being compared, the bind value is masked. In our case, the bind value of the SSN_VAR bind variable will be masked by the REDACT_AUDIT policy because SSN_VAR, and SSN appear as arguments to the equality condition. Connect as PETER, set a bind variable and execute a SELECT statement on the SSN protected sensitive column.

```
SQL> connect peter
Enter password: ******
Connected.
SQL> VAR SSN_VAR NUMBER;
SQL> exec :SSN_VAR:=987654323

PL/SQL procedure successfully completed.

SQL> SELECT ccn, ssn FROM oe.customers_info where ssn=:SSN_VAR;

                CCN          SSN
------------------- ------------
   6011000000000004    987654323

SQL>
```

4. Verify that the audited action does not display the bind value for the SSN_VAR bind variable.

```
SQL> connect / as sysdba
Connected.
SQL> select SQL_TEXT, SQL_BINDS from unified_audit_trail
     where DBUSERNAME='PETER';

SQL_TEXT
-----------------------------------------------------------------
SQL_BINDS
-----------------------------------------------------------------
… rows deleted

SELECT ccn, ssn FROM oe.customers_info where ssn=:SSN_VAR
 #1(1):*

SQL>
```

5. Disable the REDACT_AUDIT policy for the SSN sensitive column.

```
SQL> exec DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN -
          ( schema_name => 'OE', -
            table_name  => 'CUSTOMERS_INFO', -
            column_name => 'SSN', -
            policy      => 'REDACT_AUDIT')
> > > >
PL/SQL procedure successfully completed.

SQL>
```

6. Reconnect as PETER, set a bind variable and reexecute a SELECT statement on the SSN protected sensitive column.

```
SQL> connect peter
Enter password: ******
Connected.
SQL> VAR SSN_VAR NUMBER;
SQL> exec :SSN_VAR:=987654323

PL/SQL procedure successfully completed.

SQL> SELECT ccn, ssn FROM oe.customers_info where ssn=:SSN_VAR;

                CCN          SSN
------------------- ------------
   6011000000000004    987654323

SQL>
```

7. Display the bind value for the SSN_VAR bind variable.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> select SQL_TEXT, SQL_BINDS from unified_audit_trail where
DBUSERNAME='PETER';

SQL_TEXT
-----------------------------------------------------------------
SQL_BINDS
-----------------------------------------------------------------
… rows deleted

SELECT ccn, ssn FROM oe.customers_info where ssn=:SSN_VAR
 #1(1):*

SELECT ccn, ssn FROM oe.customers_info where ssn=:SSN_VAR
```

```
 #1(9):987654323

SQL>
```

8. Drop the audit policy.

```
SQL> noaudit policy pol_sel by peter;

Noaudit succeeded.

SQL> drop audit policy pol_sel;

Audit Policy dropped.

SQL>
```

Practice 16-3: Disabling TSDP Policies

Overview

In this practice, you will disable the TSDP policy created in practice 16-1.

Tasks

1. You can imagine that in a production environment, you need to perform some maintenance operations and momentarily disable the protection.

   Disable the protection at the sensitive type level.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> exec DBMS_TSDP_PROTECT.DISABLE_PROTECTION_TYPE( -
          'Sensitive_Numbers')

PL/SQL procedure successfully completed.

SQL> exec DBMS_TSDP_PROTECT.DISABLE_PROTECTION_TYPE( -
          'Income')

PL/SQL procedure successfully completed.

SQL>
```

2. Connect as `SCOTT` and check that the user can easily view any data.

```
SQL> CONNECT scott
Enter password: ******
Connected.
SQL> SELECT ccn, ssn FROM oe.customers_info;

                 CCN          SSN
------------------- ------------
   5105105105105100    987654320
   6011111111111117    987654321
    378282246310005    987654322
   6011000000000004    987654323
   4111111111111111    987654324
   5105105105105100    987654325
      4222222222222    987654326
    343434343434343    987654327
   6011000990139424    987654328
   5111111111111118    987654329
```

```
10 rows selected.

SQL> SELECT last_name, salary, commission_pct
     FROM hr.employees;
   2
LAST_NAME                    SALARY COMMISSION_PCT
------------------------- --------- --------------
… rows deleted …
Kumar                          6300             .1
Livingston                     8600             .2
Taylor                         3400
Sullivan                       2700
Sarchand                       4400
Cabrio                         3200
Dilly                          3800
Perkins                        2700
Everett                        4100
Walsh                          3300

83 rows selected.

SQL>
SQL> EXIT
$
```

Practices for Lesson 17: Encryption Concepts

Chapter 17

Practices for Lesson 17: Overview

Lesson Overview

There are no practices for this lesson.

Practices for Lesson 18: Using Application-Based Encryption

Chapter 18

Practice 18-1: Using DBMS_CRYPTO for Encryption

Overview

In this practice, you create functions to encrypt and decrypt data, and create a `KEYS` table. Then, by using the functions, you encrypt and decrypt column data. You also apply an `SHA-1` message digest to the column to verify integrity.

Tasks

1. Review and execute the `crypto_random.sql` script in the `/home/oracle/labs/ENC` directory, which performs the following actions:
   a. Adds a credit card column to the `CUSTOMERS` table
   b. Creates the `ENCRYPT` function for AES encryption
   c. Creates the `DECRYPT` function for AES decryption
   d. Creates a `KEYS` table to hold a 128-bit key value `(KEY RAW (16))`
   e. Inserts a key value generated by `DBMS_CRYTPO.RANDOM_BYTES`
   f. Shows the key value

```
$ cd ~/labs/ENC
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus /nolog @$HOME/labs/ENC/crypto_random.sql

SQL*Plus: Release 12.1.0.1.0 Production on Tue May 28 08:10:00
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

SQL>
SQL> --- Grant Execute on DBMS_CRYPTO TO OE ---
SQL>
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> GRANT EXECUTE ON DBMS_CRYPTO TO OE;

Grant succeeded.

SQL>
SQL>
SQL> CONNECT oe/oracle_4U
Connected.
```

```
SQL>
SQL> -- Alter the customers table to hold an
SQL> --  encrypted CREDITCARD Number
SQL>
SQL> ALTER TABLE customers DROP column credit_card_num;
ALTER TABLE customers DROP column credit_card_num
                                 *
ERROR at line 1:
ORA-00904: "CREDIT_CARD_NUM": invalid identifier


SQL>
SQL> ALTER TABLE customers ADD credit_card_num RAW(2000);

Table altered.

SQL>
SQL>
SQL> --- Create the encrypt_value and
SQL> --  decrypt_value functions
SQL>
SQL> create or replace function encrypt_value
  2       (
  3          p_in    in varchar2,
  4          p_key   in raw
  5       )
  6       return raw is
  7          l_enc_val raw (2000);
  8          l_mod     number := dbms_crypto.ENCRYPT_AES128
  9                      + dbms_crypto.CHAIN_CBC
 10                      + dbms_crypto.PAD_PKCS5;
 11       begin
 12          l_enc_val := dbms_crypto.encrypt
 13             (
 14             UTL_I18N.STRING_TO_RAW
 15                (p_in, 'AL32UTF8'),
 16             l_mod,
 17             p_key
 18             );
 19          return l_enc_val;
 20    end;
 21  /
```

```
Function created.

SQL>
SQL>
SQL> create or replace function decrypt_value
  2    (
  3         p_in       in raw,
  4         p_key   in raw
  5    )
  6    return varchar2
  7    is
  8         l_ret     varchar2 (2000);
  9         l_dec_val raw (2000);
 10         l_mod     number := dbms_crypto.ENCRYPT_AES128
 11                     + dbms_crypto.CHAIN_CBC
 12                     + dbms_crypto.PAD_PKCS5;
 13    begin
 14         l_dec_val := dbms_crypto.decrypt
 15            (
 16            p_in,
 17            l_mod,
 18            p_key
 19            );
 20            l_ret:= UTL_I18N.RAW_TO_CHAR
 21                    (l_dec_val, 'AL32UTF8');
 22              return l_ret;
 23   end;
 24  /

Function created.

SQL>
SQL>
SQL> -- Create KEYS table
SQL> DROP TABLE KEYS;
DROP TABLE KEYS
           *
ERROR at line 1:
ORA-00942: table or view does not exist
```

```
SQL>
SQL>
SQL> CREATE TABLE KEYS (KEY_VALUE RAW(16));

Table created.

SQL>
SQL> -- get a KEY and store it in KEYS
SQL>
SQL> INSERT INTO KEYS
  2         SELECT DBMS_CRYPTO.RANDOMBYTES(16) FROM DUAL;

1 row created.

SQL>
SQL>
SQL>  COMMIT;

Commit complete.

SQL>
SQL>  SELECT * FROM KEYS;

KEY_VALUE
--------------------------------
AD4C95D0E9D1F31DE5106463F3C103AB

SQL>
```

2. Update one of the customer's rows with a credit card number.

```
SQL> UPDATE customers
            SET credit_card_num = '12345678901234567890123 4'
     WHERE  customer_id = 101;
  2    3
1 row updated.

SQL>  COMMIT;

Commit complete.

SQL>
```

3. Verify the update by selecting the credit card number of the row just updated. Save this script because you will select this column several times in this practice.

```
SQL> SELECT credit_card_num
     FROM   customers
     WHERE  customer_id = 101;

CREDIT_CARD_NUM
------------------------------------------------------------
12345678901234567890123 4

SQL>
```

4. Encrypt the credit card number by using the function created in step 1.

```
SQL> DECLARE
      l_key RAW(16);
     BEGIN
      SELECT key_value INTO l_key FROM KEYS;

      UPDATE customers
      SET    credit_card_num
             = encrypt_value(credit_card_num, l_key)
      WHERE  customer_id = 101;

     COMMIT;
    END;
   /
2    3    4    5    6    7    8    9   10   11   12   13
PL/SQL procedure successfully completed.

SQL>
```

5. Verify the encryption by selecting the credit card number of the row just updated.

```
SQL> SELECT UTL_I18N.RAW_TO_CHAR(credit_card_num, 'AL32UTF8')
     FROM   customers
     WHERE  customer_id = 101;

UTL_I18N.RAW_TO_CHAR(CREDIT_CARD_NUM,'AL32UTF8')
----------------------------------------------------------------
?,C??V<???O)>?P?E????

SQL>
```

6. Using the function created in step 1, select the decrypted column.

```
SQL> SELECT  decrypt_value(credit_card_num,
                           (SELECT key_value FROM KEYS))
     FROM    customers
     WHERE   customer_id = 101;
```

```
DECRYPT_VALUE(CREDIT_CARD_NUM,(SELECTKEY_VALUEFROMKEYS))
--------------------------------------------------------
12345678901234567890 1234

SQL>
```

7. Update the `CUSTOMERS` table with the decrypted credit card number.

```
SQL> UPDATE customers
      SET credit_card_num=decrypt_value(credit_card_num,
                          (SELECT key_value FROM keys))
      WHERE customer_id = 101;
  2    3    4
1 row updated.

SQL> commit;

Commit complete.

SQL>
```

8. Verify that the update worked by selecting the credit card number.

```
SQL> SELECT  credit_card_num
     FROM    customers
     WHERE   customer_id = 101;

CREDIT_CARD_NUM
--------------------------------------------------------
12345678901234567890 1234

SQL>
```

Practice 18-2: Checksumming by Using the HASH Function

Overview

In this practice, you will checksum a credit card number value by using the HASH function of DBMS_CRYPTO package.

Tasks

1. What happens when you try to produce an SHA-1 checksum on the CREDIT_CARD_NUM column? Why?

   Because the procedures and functions in DBMS_CRYPTO are overloaded, the Oracle instance cannot determine the correct version of the function to call. To correct this, wrap the call in a PL/SQL function (as was done with encryption and decryption in the first step of this practice).

```
SQL> SELECT DBMS_CRYPTO.HASH(credit_card_num,
                             DBMS_CRYPTO.HASH_SH1)
     FROM   customers
     WHERE  customer_id = 101;
                        DBMS_CRYPTO.HASH_SH1)
                        *
ERROR at line 2:
ORA-06553: PLS-221: 'HASH_SH1' is not a procedure or is
undefined



SQL>
```

2. The hash.sql script creates a function called CHECKSUM that produces an SHA-1 hash of the input. Review and execute hash.sql.

```
SQL> @$HOME/labs/ENC/hash.sql
SQL> SET ECHO OFF
SQL>
SQL> CONNECT oe
Enter password: *****
Connected.
SQL>
SQL> CREATE OR REPLACE FUNCTION checksum (
  2    p_raw_input        RAW)
  3    RETURN RAW
  4  IS
  5    v_checksum         RAW(20);
  6  BEGIN
  7    v_checksum :=
  8      DBMS_CRYPTO.HASH(
  9         src => p_raw_input,
```

```
 10         typ => DBMS_CRYPTO.HASH_SH1);
 11
 12    RETURN v_checksum;
 13  END;
 14  /

Function created.
SQL>
```

3. Use the function created in the previous step to produce a checksum for the credit card number.

```
SQL> SELECT  checksum (credit_card_num)
     FROM    customers
     WHERE   customer_id = 101;

CHECKSUM(CREDIT_CARD_NUM)
----------------------------------------------------------
196FB5FB06A63A73D0F1D31D6E985C996C3AEFE9

SQL>
```

4. Change the credit card number in the table.

```
SQL> UPDATE customers
     SET    credit_card_num = '123456789A12345678901234'
     WHERE  customer_id = 101;
  2    3
1 row updated.

SQL> COMMIT;

Commit complete.

SQL>
```

5. Verify that the checksum has changed by using the function created in step 2. Compare the checksum to the value produced in step 3.

```
SQL> SELECT  checksum (credit_card_num)
     FROM    customers
     WHERE   customer_id = 101;

CHECKSUM(CREDIT_CARD_NUM)
----------------------------------------------------------
C2578E5407A57A042B24EC0CFBDF418DB62F526F

SQL> EXIT
$
```

Practices for Lesson 19: Applying Transparent Data Encryption

Chapter 19

Practice 19-1: Configuring the Password-Based Keystore for TDE

Overview

In this practice, you will configure a password-based keystore for a non-CDB and a password-based keystore for a CDB. Then you will set the master key for the non-CDB and the master key for each PDB of the CDB.

In the `sqlnet.ora` file, you must set the `ENCRYPTION_WALLET_LOCATION` parameter to specify the keystore location. When determining which keystore to use, Oracle Database searches for the keystore location in the following places, in this order:

1. First, it attempts to use the keystore in the location specified by the parameter `ENCRYPTION_WALLET_LOCATION` in the `sqlnet.ora` file.
2. If the `ENCRYPTION_WALLET_LOCATION` parameter is not set, then it attempts to use the keystore in the location that is specified by the parameter `WALLET_LOCATION`.
3. If the `WALLET_LOCATION` parameter is also not set, then Oracle Database looks for a keystore at the default database location, which is `$ORACLE_BASE/admin/DB_UNIQUE_NAME/wallet` or `$ORACLE_HOME/admin/DB_UNIQUE_NAME/wallet`. (`DB_UNIQUE_NAME` is the unique name of the database specified in the initialization parameter file.) When the keystore location is not set in the `sqlnet.ora` file, then the `V$ENCRYPTION_WALLET` view displays the default location. You can check the location and status of the keystore in the `V$ENCRYPTION_WALLET` view.

Task

1. Prepare the `orcl` database for encryption.
   a. Create a directory for the unique Oracle password-based keystore for the database at `$ORACLE_BASE/admin/orcl/wallet` if it does not exist.

```
$ mkdir $ORACLE_BASE/admin/orcl/wallet
$
```

   b. Connect to the `orcl` database instance as a user who possesses the `SYSKM` privilege to create the password-based keystore.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as syskm

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE
```

```
        '/u01/app/oracle/admin/orcl/wallet'
         IDENTIFIED BY secret;
  2    3
keystore altered.

SQL> EXIT
$
```

c. Verify that the file is created in the appropriate directory.

```
$ ls -l /u01/app/oracle/admin/orcl/wallet
total 4
-rw-r--r-- 1 oracle oinstall 2408 Jun 18 06:46 ewallet.p12
$
```

d. Open the keystore.

```
$ sqlplus / as syskm

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                               IDENTIFIED BY secret;
  2
keystore altered.

SQL>
```

e. Generate the master encryption key. The clause `WITH BACKUP USING` is mandatory and creates a backup of the keystore before the master key is created and stored in the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY
                               IDENTIFIED BY secret
                               WITH BACKUP
                               USING 'for_12c';
2    3    4
keystore altered.

SQL>
```

f. Verify that the keystore has been backed up before the master key generation.

```
SQL> !ls -l /u01/app/oracle/admin/orcl/wallet
-rw-r--r-- 1 oracle oinstall 2408 Jun 18 06:48
ewallet_2013061806481418_for_12c.p12
```

```
-rw-r--r-- 1 oracle oinstall 4112 Jun 18 06:48 ewallet.p12

SQL>
```

Notice that if you regenerate the master key, the file grows. All previous master keys are kept for data which could have used the previous master keys.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY
                               IDENTIFIED BY secret;
  2  ADMINISTER KEY MANAGEMENT SET KEY
*
ERROR at line 1:
ORA-46631: keystore needs to be backed up

SQL> ADMINISTER KEY MANAGEMENT SET KEY
                               IDENTIFIED BY secret
                               WITH BACKUP;
  2    3
keystore altered.

SQL> !ls -l /u01/app/oracle/admin/orcl/wallet

-rw-r--r-- 1 oracle oinstall 2408 Jun 18 06:48
ewallet_2013061806481418_for_12c.p12
-rw-r--r-- 1 oracle oinstall 4112 Jun 18 06:48
ewallet_2013061806485974.p12
-rw-r--r-- 1 oracle oinstall 6312 Jun 18 06:48 ewallet.p12

SQL>
```

g. Back up the keystore which contains the current master key.

```
SQL> ADMINISTER KEY MANAGEMENT BACKUP KEYSTORE
                               IDENTIFIED BY secret;
  2
keystore altered.

SQL> !ls -l /u01/app/oracle/admin/orcl/wallet

-rw-r--r-- 1 oracle oinstall 2408 Jun 18 06:48
ewallet_2013061806481418_for_12c.p12
-rw-r--r-- 1 oracle oinstall 4112 Jun 18 06:48
ewallet_2013061806485974.p12
-rw-r--r-- 1 oracle oinstall 6312 Jun 18 06:50
ewallet_2013061806500437.p12
-rw-r--r-- 1 oracle oinstall 6312 Jun 18 06:50 ewallet.p12
```

Notice that both the current and the backup files have the same size.

h. View the keystore file location from the view.

```
SQL> SELECT WRL_PARAMETER, STATUS, WALLET_TYPE, CON_ID
     FROM V$ENCRYPTION_WALLET;
  2
WRL_PARAMETER                     STATUS WALLET_TYPE   CON_ID
--------------------------------- ------ -------------- ------
/u01/app/oracle/admin/orcl/wallet OPEN   PASSWORD           0

SQL> EXIT
$
```

2. Prepare the `cdb1` multitenant container database for encryption.

a. Create a directory for the unique Oracle password-based keystore for the CDB at `$ORACLE_BASE/admin/cdb1/wallet` if it does not exist.

```
$ mkdir $ORACLE_BASE/admin/cdb1/wallet
$
```

b. Connect to the `cdb1` instance as a user who has been granted the `SYSKM` privilege to create the password-based keystore.

```
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as syskm

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL>
```

c. Display the default keystore location.

```
SQL> SELECT WRL_PARAMETER, STATUS, WALLET_TYPE
     FROM V$ENCRYPTION_WALLET;
  2

WRL_PARAMETER                     STATUS         WALLET_TYPE
--------------------------------- -------------- --------------
/u01/app/oracle/admin/cdb1/wallet NOT_AVAILABLE UNKNOWN

SQL>
```

d. Create the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE
        '/u01/app/oracle/admin/cdb1/wallet'
         IDENTIFIED BY secret_cdb1;
  2    3
keystore altered.

SQL> !ls -l /u01/app/oracle/admin/cdb1/wallet

-rw-r--r-- 1 oracle oinstall 2400 Jun 18 06:52 ewallet.p12

SQL> SELECT WRL_PARAMETER, STATUS, WALLET_TYPE, CON_ID
     FROM V$ENCRYPTION_WALLET;
  2
WRL_PARAMETER                      STATUS        WALLET_TYPE
---------------------------------- ------------- -----------
    CON_ID
----------
/u01/app/oracle/admin/cdb1/wallet CLOSED         UNKNOWN
0

SQL>
```

e. Open the keystore for all PDBS.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                               IDENTIFIED BY secret_cdb1
                               CONTAINER = ALL;
  2   3
keystore altered.

SQL>
```

```
SQL> SELECT WRL_PARAMETER, STATUS, WALLET_TYPE, CON_ID
     FROM V$ENCRYPTION_WALLET;
  2

WRL_PARAMETER                      STATUS             WALLET_TYPE
---------------------------------- ------------------ ----------
-
    CON_ID
----------
/u01/app/oracle/admin/cdb1/wallet  OPEN_NO_MASTER_KEY PASSWORD
0
```

```
SQL>
```

Notice that the status explains that the master key has not yet been generated in the keystore.

f. The application data are stored in the PDBs. Generate a master key for each of the PDBs in `cdb1`.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT name FROM v$pdbs;

NAME
------------------------------
PDB$SEED
PDB1_2
PDB1_1

SQL>
```

g. Generate a master key for `pdb1_1`.

1) Grant the `SYSKM` privilege to the keystore manager of each PDB.

```
SQL> ALTER USER syskm IDENTIFIED BY oracle_4U ACCOUNT UNLOCK
CONTAINER=ALL;

User altered.

SQL> CREATE USER c##km IDENTIFIED BY oracle_4U;

User created.

SQL> GRANT syskm TO c##km CONTAINER=ALL;

Grant succeeded.

SQL>
```

2) Connect to the `pdb1_1` to generate the master key.

```
SQL> CONNECT c##km@pdb1_1 AS SYSKM
Enter password: ******
Connected.
SQL>
```

3) Generate the master key.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY
                               IDENTIFIED BY secret_cdb1
                               WITH BACKUP
                               CONTAINER=CURRENT;
```

```
  2    3    4  ADMINISTER KEY MANAGEMENT SET KEY
*
ERROR at line 1:
ORA-46671: master key not set in root container
```

```
SQL> CONNECT / AS SYSKM
Connected.
SQL> ADMINISTER KEY MANAGEMENT SET KEY
                               IDENTIFIED BY secret_cdb1
                               WITH BACKUP;
  2    3  ADMINISTER KEY MANAGEMENT SET KEY
*
ERROR at line 1:
ORA-28417: password-based keystore is not open

SQL>
```

Notice that the keystore was automatically closed.

```
SQL> SELECT WRL_PARAMETER, STATUS, WALLET_TYPE, CON_ID
     FROM V$ENCRYPTION_WALLET;
  2
WRL_PARAMETER                      STATUS WALLET_TYPE     CON_ID
---------------------------------- ------ ------------ ---------
-
/u01/app/oracle/admin/cdb1/wallet  CLOSED UNKNOWN             0
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                               IDENTIFIED BY secret_cdb1
                               CONTAINER = ALL;
  2   3
keystore altered.

SQL>
```

4) Generate the master key in the root container.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY
                               IDENTIFIED BY secret_cdb1
                               WITH BACKUP
                               CONTAINER = ALL;
  2   3   4
keystore altered.

SQL> SELECT WRL_PARAMETER, STATUS, WALLET_TYPE, CON_ID
     FROM V$ENCRYPTION_WALLET;
```

```
  2
WRL_PARAMETER                      STATUS WALLET_TYPE     CON_ID
---------------------------------- ------ ------------ ---------
-
/u01/app/oracle/admin/cdb1/wallet  OPEN   PASSWORD            0

SQL>
SQL> SELECT KEY_ID, KEYSTORE_TYPE, KEY_USE,
            ACTIVATING_DBNAME, ACTIVATING_PDBNAME
     FROM V$ENCRYPTION_KEYS;
  2   3

AYNanq9p0U8Qv2c7YAeWvUsAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SOFTWARE KEYSTORE TDE IN PDB cdb1
PDB1_1

AW8zkuYlvE+qv8UXJmOHdAsAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SOFTWARE KEYSTORE TDE IN PDB cdb1
CDB$ROOT

AVoXTNMhDE8lv3t9ZN08AToAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SOFTWARE KEYSTORE TDE IN PDB cdb1
PDB1_2

SQL>
```

Notice that the command generated one master key for each container including the `root` container.

5) Generate a master key for `pdb1_1`.

```
SQL> CONNECT c##km@pdb1_1 AS SYSKM
Enter password: ******
Connected.
SQL>
SQL> ADMINISTER KEY MANAGEMENT SET KEY
                               IDENTIFIED BY secret_cdb1
                               WITH BACKUP
                               CONTAINER = CURRENT;
  2   3   4
keystore altered.

SQL> SELECT KEY_ID, KEYSTORE_TYPE, KEY_USE,
            ACTIVATING_DBNAME, ACTIVATING_PDBNAME
     FROM V$ENCRYPTION_KEYS;
```

```
  2   3
KEY_ID
-----------------------------------------------------------------
-
KEYSTORE_TYPE    KEY_USE    ACTIVATING_DBNAME
----------------- ---------- ------------------------------
ACTIVATING_PDBNAME
------------------------------
AYNanq9p0U8Qv2c7YAeWvUsAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SOFTWARE KEYSTORE TDE IN PDB cdb1
PDB1_1

AYyDyiRzQE//v2yzAYkINbAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SOFTWARE KEYSTORE TDE IN PDB cdb1
PDB1_1

SQL>
```

Notice that the command generated another master key for the `pdb1_1` container.

h. Generate a master key for `pdb1_2`.

```
SQL> CONNECT c##km@pdb1_2 AS SYSKM
Enter password: ******
Connected.
SQL> ADMINISTER KEY MANAGEMENT SET KEY
                               IDENTIFIED BY secret_cdb1
                               WITH BACKUP
                               CONTAINER = CURRENT;
  2   3   4
keystore altered.

SQL> SELECT KEY_ID, KEYSTORE_TYPE, KEY_USE,
            ACTIVATING_DBNAME, ACTIVATING_PDBNAME
     FROM V$ENCRYPTION_KEYS;
  2   3
KEY_ID
-----------------------------------------------------------------
--------------
KEYSTORE_TYPE    KEY_USE    ACTIVATING_DBNAME
----------------- ---------- ------------------------------
ACTIVATING_PDBNAME
------------------------------
AU0iDQnVHk+zv2qpJbKlUvEAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SOFTWARE KEYSTORE TDE IN PDB cdb1
```

```
PDB1_2

AVoXTNMhDE8lv3t9ZN08AToAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SOFTWARE KEYSTORE TDE IN PDB cdb1
PDB1_2

SQL> EXIT
$
```

Notice that the command generated another master key for the `pdb1_2` container.

Practice 19-2: Implementing Table Column Encryption

Overview

In this practice, you will create a table that contains an encrypted column. You view the data in the format that is stored on disk before and after encryption. You create an index on the encrypted column. You demonstrate that range scans are possible. You grant access to the column for a particular user, and you demonstrate that any user with proper privileges can view the unencrypted data.

Tasks

1. When Transparent Data Encryption (TDE) is applied to columns in the database, what does the application developer do to be sure that the application can handle the encrypted columns?
   a. Increase the size of the fields and the variables holding the values from the encrypted columns.
   b. Add error handling for column overruns.
   c. Add error handling for missing keys.
   d. Nothing

   Answer: *d. Nothing*
2. Create a table in the `OE` schema that holds sensitive customer payment information. Use the `create_tables.sql` script in the `/home/oracle/labs/ENC` directory to create and populate a table named `OE.CUST_PAYMENT_INFO`.

```
$ cd ~/labs/ENC
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus /nolog @create_tables.sql
```

```
SQL> connect oe/oracle_4U@localhost:1521/orcl
Connected.

SQL> drop table cust_payment_info;
drop table cust_payment_info
           *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> create table cust_payment_info
  2    (first_name varchar2(11),
  3    last_name varchar2(10),
  4    order_number number(5),
  5    credit_card_number varchar2(20),
  6    active_card varchar2(3));
```

```
Table created.

SQL>
SQL> insert into cust_payment_info values
  2   ('Jon', 'Oldfield', 10001, 5105105105105100,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Chris', 'White', 10002, 6011111111111117,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Alan', 'Squire', 10003, 378282246310005,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Mike', 'Anderson', 10004, 6011000000000004,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Annie', 'Schmidt', 10005, 4111111111111111,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Elliott', 'Meyer', 10006, 4222222222222,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Celine', 'Smith', 10007, 343434343434343,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Steve', 'Haslam', 10008, 6011000990139424,'YES');
```

```
1 row created.

SQL> insert into cust_payment_info values
  2   ('Albert', 'Einstein', 10009, 5111111111111118,'YES');

1 row created.

SQL>
SQL> create index cust_payment_info_idx on
  2    cust_payment_info (credit_card_number);

Index created.

SQL>
```

3. Select the data from the `OE.CUST_PAYMENT_INFO` table.

```
SQL> column FIRST_NAME format A8 head 'First'
SQL> column ORDER_NUMBER format 999999 Head "Order#"
SQL> select * from oe.cust_payment_info;

First     LAST_NAME   Order# CREDIT_CARD_NUMBER  ACT
--------- ---------- ------- -------------------- ---
Jon       Oldfield     10001 5105105105105100    YES
Chris     White        10002 6011111111111117    YES
Alan      Squire       10003 378282246310005     YES
Mike      Anderson     10004 6011000000000004    YES
Annie     Schmidt      10005 4111111111111111    YES
Elliott   Meyer       10006 4222222222222        YES
Celine    Smith       10007 343434343434343      YES
Steve     Haslam       10008 6011000990139424    YES
Albert    Einstein     10009 5111111111111118    YES

9 rows selected.

SQL>
```

4. Dump the data blocks to see the data as it is stored in the file. Do this as the `SYS` user.
   a. Find the database address of the `OE.CUST_PAYMENT_INFO` table. The `$HOME/labs/ENC/dump_blocks.sql` script executes the following:

```
SELECT file_id FROM dba_data_files
WHERE RELATIVE_FNO =
  (SELECT distinct dbms_rowid.ROWID_RELATIVE_FNO(rowid) FILE#
   FROM   oe.cust_payment_info);
```

```
SELECT distinct dbms_rowid.rowid_block_number(rowid) BLOCK#
FROM   oe.cust_payment_info;
```

Execute the script and determine `file#` and `block#` for your table *(these numbers vary).*

```
SQL> @dump_blocks.sql
SQL> connect sys/oracle_4U@localhost:1521/orcl as sysdba
Connected.
SQL>
SQL> SELECT file_id FROM dba_data_files
  2  WHERE RELATIVE_FNO =
  3    (SELECT distinct dbms_rowid.ROWID_RELATIVE_FNO(rowid)
FILE#
  4       FROM   oe.cust_payment_info);

   FILE_ID
----------
         2

SQL>
SQL> SELECT distinct dbms_rowid.rowid_block_number(rowid) BLOCK#
  2  FROM   oe.cust_payment_info;

    BLOCK#
----------
     41389

SQL>
```

b. Set the `TRACEFILE_IDENTIFIER` initialization parameter so that the trace file can be found more easily by executing the following command:

```
ALTER SESSION SET TRACEFILE_IDENTIFIER=dp_block;
```

```
SQL> ALTER SESSION SET TRACEFILE_IDENTIFIER=dp_block;

Session altered.

SQL>
```

c. Dump the data block to a trace file. Substituting the `file#` and `block#` that you recorded with the previous command, execute the following command:

```
ALTER SYSTEM DUMP DATAFILE <file#> BLOCK <block#>;
```

```
SQL> ALTER SYSTEM DUMP DATAFILE 2 BLOCK 41389;

System altered.

SQL>
```

d. Find the trace file. In this listing, the block dump is in the `orcl_ora_<pid>_DP_BLOCK.trc` file.

```
SQL> show parameter user_dump_dest

NAME                 TYPE        VALUE
------------------- ----------- ------------------------------
user_dump_dest       string      /u01/app/oracle/diag/rdbms/orc
                                 l/orcl/trace

SQL> EXIT
$
$ cd /u01/app/oracle/diag/rdbms/orcl/orcl/trace
$ ls *DP_BLOCK*
orcl_ora_625_DP_BLOCK.trc  orcl_ora_625_DP_BLOCK.trm
$
```

e. View the dump file. The `less` utility enables you to scroll up and down the file to find data of interest. Note that the credit card numbers are clearly visible.

```
$ less orcl_ora_625_DP_BLOCK.trc
/* Rows deleted */
…
2C70FE70 02FFFF80 C30407C1 03252C02 024A09C2  [.........,%...J.]
2C70FE80 800102C1 0605012C 65626C41 45087472  [....,...Albert.E]
2C70FE90 74736E69 046E6965 0A0102C3 3031330F  [instein......511]
2C70FEA0 33343536 31343530 39383332 53455903
[1111111111118.YES]
2C70FEB0 0505012C 76657453 61480665 6D616C73  [,...Steve.Haslam]
2C70FEC0 0102C304 34330F09 35373930 33303039
[......60110009901]
2C70FED0 35383637 45590338 05012C53 6C654306  [39424.YES,...Cel]
2C70FEE0 05656E69 74696D53 02C30468 340D0801
[ine.Smith......34]
2C70FEF0 38363137 33353839 36333033 53455903
[3434343434343.YES]
2C70FF00 0705012C 696C6C45 0574746F 6579654D  [,...Elliott.Meye]
2C70FF10 02C30472 330F0701 36333437 39393536  [r...... 4222222]
2C70FF20 38313137 59033032 012C5345 6E410505  [222222.YES,...An]
2C70FF30 0765696E 6D686353 04746469 060102C3  [nie.Schmidt.....]
2C70FF40 35353410 38383936 32383037 30393633  [.411111111111111]
2C70FF50 45590332 05012C53 6B694D04 6E410865  [1.YES,...Mike.An]
2C70FF60 73726564 C3046E6F 10050102 39323934  [derson......6011]
2C70FF70 35393838 35333637 30303437 53455903  [000000000004.YES]
2C70FF80 0405012C 6E616C41 75715306 04657269  [,...Alan.Squire.]
2C70FF90 040102C3 39353510 38363935 37333439  [.....3782822463]
```

```
2C70FFA0 32393735 45590330 05012C53 72684305  [10005.YES,...Chr]
2C70FFB0 57057369 65746968 0102C304 31351003  [is.White......60]
2C70FFC0 35333232 36343038 35323830 59033036  [11111111111117.Y]
2C70FFD0 012C5345 6F4A0305 6C4F086E 65696664  [ES,...Jon.Oldfie]
2C70FFE0 C304646C 10020102 36343435 37393539  [ld......51051051]
2C70FFF0 31383830 35383932 53455903 49B2060B  [05105100.YES...I]
Block header dump:  0x01000198
…
q - to exit less

$
```

5. Alter the table to encrypt the credit card numbers with `NO SALT`.

```
$ sqlplus oe

SQL*Plus: Release 12.1.0.1.0 Production on Wed Aug 7 03:06:00
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter password: ******
Connected.
SQL> desc cust_payment_info
 Name                      Null?    Type
 ------------------------- -------- ------------------------
 FIRST_NAME                         VARCHAR2(11)
 LAST_NAME                          VARCHAR2(10)
 ORDER_NUMBER                       NUMBER(5)
 CREDIT_CARD_NUMBER                 VARCHAR2(20)
 ACTIVE_CARD                        VARCHAR2(3)

SQL> ALTER TABLE cust_payment_info
     MODIFY (CREDIT_CARD_NUMBER encrypt no salt);
  2
Table altered.

SQL>
```

6. Dump the data block and find the trace file. Change `TRACEFILE_IDENTIFIER` to `DUMP2`.
   a. Use the `$HOME/labs/ENC/dump_blocks.sql` script to find the data block address.

```
SQL> @$HOME/labs/ENC/dump_blocks.sql
SQL> connect / as sysdba
Connected.
```

```
SQL>
SQL> SELECT file_id FROM dba_data_files
  2  WHERE RELATIVE_FNO =
  3    (SELECT distinct dbms_rowid.ROWID_RELATIVE_FNO(rowid)
FILE#
  4       FROM   oe.cust_payment_info);

   FILE_ID
----------
      2

SQL>
SQL> SELECT distinct dbms_rowid.rowid_block_number(rowid) BLOCK#
  2  FROM   oe.cust_payment_info;

    BLOCK#
----------
     41389

SQL>
```

7. Set the `TRACEFILE_IDENTIFIER` initialization parameter so that the trace file can be found more easily.
   a. Use `ALTER SESSION SET TRACEFILE_IDENTIFIER=DUMP2;`

```
SQL> ALTER SESSION SET TRACEFILE_IDENTIFIER=DUMP2;

Session altered.

SQL>
```

   b. As the `SYS` user, dump the data block to a trace file. Substituting the `file#` and `block#` that you recorded with the previous command, execute the following command:
   `ALTER SYSTEM DUMP DATAFILE <file#> BLOCK <block#>;`

```
SQL> ALTER SYSTEM DUMP DATAFILE 2 BLOCK 41389;

System altered.

SQL> EXIT
$
```

   c. Find the trace file.

```
$ ls *DUMP*.trc
orcl_ora_5358_DUMP2.trc
$
```

d. View the trace file. Note that the unencrypted data remains.

```
$ less orcl_ora_5358_DUMP2.trc
…
7AA470 39141603 0301002C 053202C1 6E780700  [...9,.....2...xn]
7AA480 05160302 0605012C 65626C41 45087472  [....,...Albert.E]
7AA490 74736E69 046E6965 0A0102C3 3031330F  [instein...... 511]
7AA4A0 33343536 31343530 39383332 53455903  [1111111111118.YES]
7AA4B0 0505012C 76657453 61480665 6D616C73  [,...Steve.Haslam]
7AA4C0 0102C304 34330F09 35373930 33303039  [......60110009901]
7AA4D0 35383637 45590338 05012C53 6C654306  [39424.YES,...Cel]
7AA4E0 05656E69 74696D53 02C30468 340D0801  [ine.Smith......34]
7AA4F0 38363137 33353839 36333033 53455903  [3434343434343.YES]
7AA500 0705012C 696C6C45 0574746F 6579654D  [,...Elliott.Meye]
7AA510 02C30472 330F0701 36333437 39393536  [r...... 4222222]
7AA520 38313137 59033032 012C5345 6E410505  [222222.YES,...An]
7AA530 0765696E 6D686353 04746469 060102C3  [nie.Schmidt.....]
7AA540 35353410 38383936 32383037 30393633  [.411111111111111]
7AA550 45590332 05012C53 6B694D04 6E410865  [1.YES,...Mike.An]
7AA560 73726564 C3046E6F 10050102 39323934  [derson...... 6011]
7AA570 35393838 35333637 30303437 53455903  [000000000004.YES]
7AA580 0405012C 6E616C41 75715306 04657269  [,...Alan.Squire.]
7AA590 040102C3 39353510 38363935 37333439  [.....3782822463]
7AA5A0 32393735 45590330 05012C53 72684305  [10005.YES,...Chr]
7AA5B0 57057369 65746968 0102C304 31351003  [is.White......60]
7AA5C0 35333232 36343038 35323830 59033036  [11111111111117.Y]
7AA5D0 012C5345 6F4A0305 6C4F086E 65696664  [ES,...Jon.Oldfie]
7AA5E0 C304646C 10020102 36343435 37393539  [ld...... 51051051]
7AA5F0 31383830 35383932 53455903 50870601  [05105100.YES...P]
…
q  /* to exit less */
$
```

8. Move the `OE.CUST_PAYMENT_INFO` table. This causes the valid data to be written to new blocks. It also makes the index unusable, so you must rebuild the index.

```
$ sqlplus oe
Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> alter table oe.cust_payment_info move;


Table altered.
```

```
SQL> select index_name, table_name, status
     from   user_indexes
     where  table_name ='CUST_PAYMENT_INFO';
2    3
INDEX_NAME             TABLE_NAME              STATUS
---------------------- ----------------------- --------
CUST_PAYMENT_INFO_IDX  CUST_PAYMENT_INFO       UNUSABLE

SQL> ALTER INDEX CUST_PAYMENT_INFO_IDX REBUILD;

Index altered.

SQL> select index_name, table_name, status
     from   user_indexes
     where  table_name = 'CUST_PAYMENT_INFO';
  2    3
INDEX_NAME            TABLE_NAME                 STATUS
--------------------- -------------------------- --------
CUST_PAYMENT_INFO_IDX CUST_PAYMENT_INFO          VALID

SQL>
```

9. Find the new block location. Dump the block and view it. Are the credit card numbers visible?

```
SQL> @$HOME/labs/ENC/dump_blocks.sql
SQL> connect sys/oracle_4U@localhost:1521/orcl as sysdba
Connected.
SQL>
SQL> SELECT file_id FROM dba_data_files
  2  WHERE RELATIVE_FNO =
  3    (SELECT distinct dbms_rowid.ROWID_RELATIVE_FNO(rowid)
FILE#
  4       FROM   oe.cust_payment_info);

   FILE_ID
----------
         2

SQL>
SQL> SELECT distinct dbms_rowid.rowid_block_number(rowid) BLOCK#
  2  FROM   oe.cust_payment_info;

    BLOCK#
```

```
----------
     41403

SQL> ALTER SESSION SET TRACEFILE_IDENTIFIER=DUMP3;

Session altered.

SQL> ALTER SYSTEM DUMP DATAFILE 2 BLOCK 41403;

System altered.

SQL> EXIT
$
$ ls *DUMP3*
orcl_ora_6495_DUMP3.trc  orcl_ora_6495_DUMP3.trm
$ less orcl_ora_6495_DUMP3.trc
…
B7FBE370 00000000 00000000 002C0000 6C410605  [..........,...Al]
B7FBE380 74726562 6E694508 69657473 02C3046E  [bert.Einstein...]
B7FBE390 EF240A01 1D1CFDC1 EFDFBC07 DCC9D8CC  [..$............]
B7FBE3A0 9740148D 6FD843CE 22084248 2560D017  [..@..C.oHB."..`%]
B7FBE3B0 43DB5AFC 0308C8D3 2C534559 53050500  [.Z.C....YES,...S]
B7FBE3C0 65766574 73614806 046D616C 090102C3  [teve.Haslam.....]
B7FBE3D0 A76A7B24 C433FDA7 E55A9A17 435784E7  [${j...3...Z...WC]
B7FBE3E0 534B9E14 3C680588 0F0E50A7 06CDABCC  [..KS..h<.P......]
B7FBE3F0 3AF0D07A 45590382 05002C53 6C654306  [z..:..YES,...Cel]
B7FBE400 05656E69 74696D53 02C30468 4E240801  [ine.Smith.....$N]
B7FBE410 BDA6E5BC 908D0883 54413436 0716917E  [........64AT~...]
B7FBE420 E858F608 85B70F84 C1610063 1564A265  [..X.....c.a.e.d.]
B7FBE430 0331EC2B 2C534559 45070500 6F696C6C  [+.1.YES,...Ellio]
B7FBE440 4D057474 72657965 0102C304 09612407  [tt.Meyer.....$a.]
B7FBE450 B6017970 D4353AC8 2F8FD3B8 BB2568CA  [py...:5..../.h%.]
B7FBE460 D85FFBEB B91222F7 5FB559C2 D46D90D9  [.._.."...Y._..m.]
B7FBE470 59031345 002C5345 6E410505 0765696E  [E..YES,...Annie.]
B7FBE480 6D686353 04746469 060102C3 16F11234  [Schmidt.....4...]
B7FBE490 04D42CDA 9CFD4B27 417864FF 76918F95  [.,..'K...dxA...v]
B7FBE4A0 6807D5C6 AE87B5A4 C24EEFB6 15BA3F62  [...h......N.b?..]
B7FBE4B0 45490484 3D0C1844 E4BAA4CA FDB117D7  [..IED..=........]
B7FBE4C0 4559039F 05002C53 6B694D04 6E410865  [..YES,...Mike.An]
B7FBE4D0 73726564 C3046E6F 34050102 C48C24C3  [derson.....4.$..]
B7FBE4E0 5CA4D6BB 50C0BFF8 4092C385 E4F9A9F4  [...\...P...@....]
B7FBE4F0 0FE6A0E4 969ACC27 88F92DEC E6180192  [....'....-......]
```

```
B7FBE500 AAA8517D FC23C153 01A08F80 22EC508C  [}Q..S.#......P."]
B7FBE510 53455903 0405002C 6E616C41 75715306  [.YES,...Alan.Squ]
B7FBE520 04657269 040102C3 AF0E2734 885468AF  [ire.....4'...hT.]
B7FBE530 73D937F7 6D1925EF 2682FC5F A711AC6B  [.7.s.%.m_..&k...]
B7FBE540 DD61BDC0 9922B23E 3EDF5EA4 CBEE20FA  [..a.>."..^.>. ..]
B7FBE550 C2B7268D CB7EEC1A 14F098B8 45590312  [.&....~.......YE]
B7FBE560 05002C53 72684305 57057369 65746968  [S,...Chris.White]
B7FBE570 0102C304 A1C73403 0E3E6A61 12C64922  [.....4..aj>."I..]
B7FBE580 D0AE9D2C B4235D85 02AC9472 B18F63B3  [,....]#.r....c..]
B7FBE590 B0BD718C 901407AA EE961735 DBCDB4CF  [.q......5.......]
B7FBE5A0 BE2E6E65 DC081E9C 5903F5A7 002C5345  [en.........YES,.]
B7FBE5B0 6F4A0305 6C4F086E 65696664 C304646C  [..Jon.Oldfield..]
B7FBE5C0 34020102 4DBECDB5 D60B61DE 29A62975  [...4...M.a..u).)]
B7FBE5D0 F5CCBA5F 685E08DF C004E0E5 A2D8BC1E  [_.....^h........]
B7FBE5E0 5FEF3520 518764B7 F34C77C9 BFC861DE  [ 5._.d.Q.wL..a..]
B7FBE5F0 E56F540C 48B94BF9 53455903 816F0602  [.To..K.H.YES..o.]
Block header dump:  0x010001a4
…
q   /* to exit less */
$
```

10. Create the `LSMITH`, `LDORAN`, and `JKING` users by using the `/home/oracle/labs/ENC/create_users.sql` script. Grant each of them the `CREATE SESSION` privilege and grant `DBA` to `LSMITH`. Only `SYS` and `SYSTEM` have the privileges required to grant the DBA role.

```
$ sqlplus /nolog @$HOME/labs/ENC/create_users.sql

SQL*Plus: Release 12.1.0.1.0 Production on Thu May 30 01:39:57
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

SQL> connect  system/oracle_4U@localhost:1521/orcl
Connected.
SQL>
SQL> grant create session to JKING identified by oracle_4U;

Grant succeeded.

SQL> grant create session, DBA to LSMITH identified by
oracle_4U;

Grant succeeded.
```

```
SQL> grant create session to LDORAN identified by oracle_4U;

Grant succeeded.

SQL>
```

11. Grant privileges to the users on the `OE.CUST_PAYMENT_INFO` table. Grant the `SELECT` privilege to `LDORAN` and `JKING`. Grant `SELECT` and `UPDATE` privileges to `LSMITH`. Use the `privs.sql` script.

```
SQL> @$HOME/labs/ENC/privs
SQL> CONNECT OE/oracle_4U@localhost:1521/orcl
Connected.
SQL>
SQL> grant select on oe.CUST_PAYMENT_INFO to LDORAN;

Grant succeeded.

SQL> grant select, update on oe.CUST_PAYMENT_INFO to LSMITH;

Grant succeeded.

SQL> grant select on oe.CUST_PAYMENT_INFO to JKING;

Grant succeeded.

SQL>
```

12. Is an index range scan possible on an index over an encrypted column? As the `LSMITH` user, update a record based on the credit card number. View the explain plan for the update statement. Use the `scan.sql` script.

    The lab script uses the `WHERE` clause, where `CREDIT_CARD_NUMBER='6011111111111117'` to select the row to update. A range scan of the index is performed. The credit card number is stored as an encrypted value in both the column and the index; the literal value is encrypted before it is compared. The value is found in the index by using a range scan. The range scan is possible only when an equality predicate is used.

```
SQL> @$HOME/labs/ENC/scan.sql
SQL> SET ECHO ON
SQL> conn LSMITH/oracle_4U@localhost:1521/orcl
Connected.
SQL> update oe.CUST_PAYMENT_INFO set ACTIVE_CARD='NO'
  2       where CREDIT_CARD_NUMBER='6011111111111117';

1 row updated.
```

```
SQL>
SQL> PAUSE 'HIT Return to show execution plan'
'HIT Return to show execution plan'

SQL> Set pagesize 100
SQL> Set linesize 70
SQL> select * from table (dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------
-
SQL_ID    19g90uxc66plt, child number 0
-------------------------------------
update oe.CUST_PAYMENT_INFO set ACTIVE_CARD='NO'    where
CREDIT_CARD_NUMBER='6011111111111117'

Plan hash value: 2780468320
-----------------------------------------------------------------
-
| Id  | Operation      | Name           | Rows  | Bytes | Co
st (%CPU)| Time     |
-----------------------------------------------------------------
-
|   0 | UPDATE STATEMENT  |                  |     |     |
  2 (100)|          |
|   1 |  UPDATE        | CUST_PAYMENT_INFO   |     |     |
     |        |
|*  2 |   INDEX RANGE SCAN| CUST_PAYMENT_INFO_IDX |   1 |    49
|
  1   (0)| 00:00:01 |

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CREDIT_CARD_NUMBER"='6011111111111117')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)


24 rows selected.
```

```
SQL> EXIT;
$
```

13. Transparent Data Encryption is not visible to the end user. No changes are required to the application or SQL syntax. Any user that has been granted privileges to access the table or column can view the data in its unencrypted form. As the `LDORAN` user, select the `LAST_NAME` and `CREDIT_CARD_NUMBER` columns from the `OE.CUST_PAYMENT_INFO` table.

```
$ sqlplus ldoran@orcl
Enter password : ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> select last_name, credit_card_number
     from oe.cust_payment_info;
  2
LAST_NAME  CREDIT_CARD_NUMBER
---------- --------------------
Oldfield   5105105105105100
White      6011111111111117
Squire     378282246310005
Anderson   6011000000000004
Schmidt    4111111111111111
Meyer      4222222222222
Smith      343434343434343
Haslam     6011000990139424
Einstein   5111111111111118

9 rows selected.

SQL>
```

14. What should you do when the keystore is not available? Close the keystore.

```
SQL> CONNECT / as syskm
Connected.

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE
                               IDENTIFIED BY secret;
  2
keystore altered.

SQL>
```

15. Connect as the LSMITH user with the password oracle_4U. Attempt to select all columns from the OE.CUST_PAYMENT_INFO table. Then, attempt to select only the LAST_NAME column.

```
SQL> connect lsmith@orcl
Enter password: ******
Connected.
SQL> select * from oe.cust_payment_info;
select * from oe.cust_payment_info
                 *
ERROR at line 1:
ORA-28365: wallet is not open

SQL> select last_name from oe.cust_payment_info;

LAST_NAME
----------
Oldfield
White
Squire
Anderson
Schmidt
Meyer
Smith
Haslam
Einstein

9 rows selected.

SQL>
```

16. As the user who has been granted the SYSKM privilege, open the keystore.

```
SQL> connect / as syskm
Connected.
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                               IDENTIFIED BY secret;
  2
keystore altered.

SQL>
```

17. Connect again as the LSMITH user with the password oracle_4U. Attempt to select all the columns from the OE.CUST_PAYMENT_INFO table.

```
SQL> connect lsmith@orcl
Enter password: ******
```

```
Connected.
SQL> select * from oe.cust_payment_info;

FIRST     LAST_NAME  ORDER_NUMBER CREDIT_CARD_NUMBER   ACT
--------- ---------- ------------ -------------------- ---
Jon       Oldfield         10001  5105105105105100    YES
Chris     White             10002  6011111111111117   NO
Alan      Squire           10003  378282246310005     YES
Mike      Anderson         10004  6011000000000004    YES
Annie     Schmidt          10005  4111111111111111    YES
Elliott   Meyer             10006  4222222222222      YES
Celine    Smith             10007  343434343434343    YES
Steve     Haslam           10008  6011000990139424    YES
Albert    Einstein         10009  5111111111111118    YES

9 rows selected.

SQL>
```

18. Drop the `OE.CUST_PAYMENT_INFO` table and re-create it with `SALT`. Then, create an index on the encrypted column `CREDIT_CARD_NUMBER`. Use the `salt.sql` script. What happens when the `create index` command is issued?

.Execute the `salt.sql` script. An index cannot be created on a column with `SALT`.

```
SQL> @$HOME/labs/ENC/salt.sql
SQL> connect oe/oracle_4U@localhost:1521/orcl
Connected.
SQL> SQL> drop table cust_payment_info;

Table dropped.

SQL> create table cust_payment_info
  2    (first_name varchar2(11),
  3    last_name varchar2(10),
  4    order_number number(5),
  5    credit_card_number varchar2(20) encrypt SALT,
  6    active_card varchar2(3));

Table created.

SQL>
SQL> insert into cust_payment_info values
  2    ('Jon', 'Oldfield', 10001, 5446959708812985,'YES');
```

```
1 row created.

SQL> insert into cust_payment_info values
  2    ('Chris', 'White', 10002, 5122358046082560,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2    ('Alan', 'Squire', 10003, 5595968943757920,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2    ('Mike', 'Anderson', 10004, 4929889576357400,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2    ('Annie', 'Schmidt', 10005, 4556988708236902,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2    ('Elliott', 'Meyer', 10006, 374366599711820,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2    ('Celine', 'Smith', 10007, 4716898533036,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2    ('Steve', 'Haslam', 10008, 340975900376858,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2    ('Albert', 'Einstein', 10009, 310654305412389,'YES');

1 row created.
```

```
SQL> create index cust_payment_info_idx
  2  on cust_payment_info (credit_card_number);
on cust_payment_info (credit_card_number)
                      *
ERROR at line 2:
ORA-28338: Column(s) cannot be both indexed and encrypted with
salt

SQL> exit
$
```

Practice 19-3: Implementing Tablespace Encryption

Overview

In this practice, you create an encrypted tablespace and move several tables and the associated indexes to the encrypted tablespace.

Tasks

1. Create an encrypted tablespace named `ENCTBS`, with a file `enctbs01.dbf`, in the same directory with the rest of the data files: `/u01/app/oracle/oradata/orcl/enctbs01.dbf`. Use the `tablespace.sql` script to create the encrypted tablespace.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> @$HOME/labs/ENC/tablespace.sql
SQL> SET ECHO ON
SQL>
SQL> DROP TABLESPACE "ENCTBS"
  2  INCLUDING CONTENTS AND DATAFILES
  3  /
DROP TABLESPACE "ENCTBS"
*
ERROR at line 1:
ORA-00959: tablespace 'ENCTBS' does not exist



SQL>
SQL> CREATE TABLESPACE "ENCTBS"
  2  DATAFILE '/u01/app/oracle/oradata/orcl/enctbs01.dbf' SIZE
100M
  3   EXTENT MANAGEMENT LOCAL
  4   SEGMENT SPACE MANAGEMENT AUTO
  5   DEFAULT STORAGE (ENCRYPT)
  6   ENCRYPTION USING 'AES192'
  7  /

Tablespace created.
SQL>
```

2. Move the `HR` schema to `ENCTBS` by using Enterprise Manager Cloud Control.

Step	Page	Action
a.	Browser	Enter the following URL: https://localhost:7802/em Log in as `sysman` with password `Oracle123`
b.	Enterprise Summary	Click **Targets**, then click **Databases**. Select `orcl` and click.
b.	orcl	Click the **Schema** tab, then click **Database Objects**, then click **Reorganize Objects**.
c.	Database Login	Click **Login**.
d.	Reorganize Objects: Type	Select **Schema Objects**. Click **Next**.
e.	Reorganize Objects: Objects	Click **Add**.
f.	Objects: Add	Enter **`HR`** as the schema. Click **Search**.
g.	Objects: Add	Click **Select All**. Click **Next 10**. Click **Select All**. Click **Next 3**. Click **Select All**. Click **OK**.
h.	Reorganize Objects: Objects.	You should see 23 objects (only 10 will be displayed at a time). Click **Set Attributes By Type**.
i.	Objects: Set Attributes By Type	In the Destination Tablespace for Tables section, select “Relocate objects to another tablespace” and enter **`ENCTBS`**. In the Destination Tablespace for Indexes section, select “Relocate objects to another tablespace” and enter **`ENCTBS`**. Click **OK**.
j.	Reorganize Objects: Objects	Click **Next**.
k.	Reorganize Objects: Options	Accept the defaults. Click **Next**.
l	Reorganize Objects: Impact Report.	Check the report message. Click **Next**.
m	Reorganize Objects: Schedule	Click Named for the host credentials: `CREDOS` appears. Click **Next**.
n.	Reorganize Objects:	Click **Submit job**.

	Review	
o.	Confirmation	Click **REORGANIZE_***.
p.	Job Run: REORGANIZE_*	Click the Refresh button of the browser periodically until all the job steps show a status of Succeeded.
	Job Run: REORGANIZE_*	Click Log Report to read all commands executed by the job. Click **Done**.
q.	On the Job Run:… page	Click **Logout**. Close the browser.

3. Connect as `HR`, and describe and view the `EMPLOYEES` table. The encrypted tablespace, including the indexes, is completely transparent to the applications.

```
SQL> CONNECT hr@orcl
Enter password: ******
Connected.

SQL> desc employees
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
 EMPLOYEE_ID                                        NUMBER(6)
 FIRST_NAME                                         VARCHAR2(20)
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 EMAIL                                     NOT NULL VARCHAR2(25)
 PHONE_NUMBER                                       VARCHAR2(20)
 HIRE_DATE                                 NOT NULL DATE
 JOB_ID                                    NOT NULL VARCHAR2(10)
 SALARY                                             NUMBER(8,2)
 COMMISSION_PCT                                     NUMBER(2,2)
 MANAGER_ID                                         NUMBER(6)
 DEPARTMENT_ID                                      NUMBER(4)

SQL> SELECT * FROM employees
     WHERE  employee_id = 106;
  2
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- -------------------------
EMAIL                     PHONE_NUMBER         HIRE_DATE JOB_ID
SALARY
------------------------- -------------------- --------- -------
--- ----------
COMMISSION_PCT MANAGER_ID DEPARTMENT_ID
-------------- ---------- -------------
        106 Valli                Pataballa
```

```
VPATABAL                  590.423.4560         05-FEB-98 IT_PROG
4800
                      103            60



SQL> SELECT tablespace_name FROM user_segments
     WHERE segment_name='EMPLOYEES';
  2
TABLESPACE_NAME
------------------------------
ENCTBS

SQL> EXIT
$
```

4. Clean up the environment moving the HR schema back into the EXAMPLE tablespace.

   Note: This script was generated by the Reorganize Objects wizard in Enterprise Manager Cloud Control to move back all HR objects to the EXAMPLE tablespace.

```
$ $HOME/labs/ENC/back_to_example_tbs.sh
sqlplus sys/oracle_4U@localhost:1521/orcl as sysdba
@$HOME/labs/ENC/back_to_example_tbs.sql

SQL*Plus: Release 12.1.0.1.0 Production on Sun Jul 7 15:18:26
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.



Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics
and Real Application Testing options

Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics
and Real Application Testing options
$
```

Practices for Lesson 20: Applying File Encryption

Chapter 20

Practice 20-1: Using RMAN Backup File Encryption

Overview

Recovery Manager (RMAN) backups to disk can be encrypted.

Task

1. Configure Recovery Manager (RMAN) to use transparent encryption for the `orcl` database. Set the configuration to be a permanent configuration in the control file.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ rman target '"john@orcl AS SYSBACKUP"'

target database Password: ******
connected to target database: ORCL (DBID=1345659572)

RMAN> select user from dual;

using target database control file instead of recovery catalog
USER
------------------------------
SYSBACKUP

RMAN> show all;

RMAN configuration parameters for database with db_unique_name
ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO
'%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO
BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
```

```
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT'
OPTIMIZE FOR LOAD TRUE ; # default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/u01/app/oracle/product/12.1.0/dbhome_1/dbs/snapcf_orcl.f'; #
default

RMAN> CONFIGURE ENCRYPTION FOR DATABASE ON;

new RMAN configuration parameters:
CONFIGURE ENCRYPTION FOR DATABASE ON;
new RMAN configuration parameters are successfully stored

RMAN> EXIT
$
```

2. Back up the `EXAMPLE` tablespace by using transparent encryption.

   Note: The database is in `NOARCHIVELOG` mode, so an online backup is not possible.

   a. Create a directory to hold the backups.

```
$ mkdir $HOME/backup
$
```

   b. Shut down the database and issue `startup mount` to perform a cold backup.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> SHUTDOWN IMMEDIATE
database closed
database dismounted
Oracle instance shut down
SQL> STARTUP MOUNT
ORACLE instance started.

Total System Global Area 501059584 bytes
Fixed Size               2290024 bytes
Variable Size          264244888 bytes
Database Buffers       226492416 bytes
Redo Buffers             8032256 bytes
Database mounted.
SQL> EXIT
```

```
$
```

c. Use the RMAN `BACKUP` command to make a backup to `/home/oracle/backup/example001.bck`. Set `tag = transparent` so that it can be specified in the `restore` command.

```
$ rman target '"john@orcl AS SYSBACKUP"'

target database Password: ******
connected to target database: ORCL (DBID=1345659572, not open)

RMAN> backup tablespace example
```

```
      format '/home/oracle/backup/example001.bck'
      tag 'transparent';
2> 3>

Starting backup at 18-JUN-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=12 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00002
name=/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: starting piece 1 at 18-JUN-13
RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
RMAN-00571:
===========================================================
RMAN-03009: failure of backup command on ORA_DISK_1 channel at
06/18/2013 07:56:40
ORA-19914: unable to encrypt backup
ORA-28365: wallet is not open

RMAN> EXIT
$
```

d. Open the keystore.

```
$ sqlplus / as SYSKM

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                                 IDENTIFIED BY secret;
  2
keystore altered.

SQL> EXIT
$
```

e. Perform the backup

```
$ rman target '"john@orcl AS SYSBACKUP"'

target database Password: ******
connected to target database: ORCL (DBID=1345659572, not open)
```

```
RMAN> backup tablespace example
      format '/home/oracle/backup/example001.bck'
      tag 'transparent';
2>  3>

Starting backup at 18-JUN-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=11 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00002
name=/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: starting piece 1 at 18-JUN-13
channel ORA_DISK_1: finished piece 1 at 18-JUN-13
piece handle=/home/oracle/backup/example001.bck tag=TRANSPARENT
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 18-JUN-13

RMAN>
```

f. List the encrypted backups.

```
RMAN> SELECT tag, encrypted FROM v$backup_piece;

TAG                              ENC
-------------------------------- ---
TRANSPARENT                      YES
```

```
RMAN>
```

3. Back up the `EXAMPLE` tablespace using dual-mode encryption to `/home/oracle/backup/example002.bck`. Set `tag = dual` so that it can be specified in the `restore` command. To set encryption mode and password, use the following command:

   `SET ENCRYPTION ON IDENTIFIED BY "oracle1";`

   a. Set encryption mode and password.

```
RMAN> SET ENCRYPTION ON IDENTIFIED BY "oracle1";

executing command: SET encryption

RMAN>
```

   b. Use the RMAN `BACKUP` command to make a backup to `/home/oracle/backup/example002.bck`.

```
RMAN> backup tablespace example
      format '/home/oracle/backup/example002.bck'
      tag 'dual';
2> 3>
Starting backup at 18-JUN-13
using channel ORA_DISK_1

channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00002
name=/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: starting piece 1 at 18-JUN-13
channel ORA_DISK_1: finished piece 1 at 18-JUN-13
piece handle=/home/oracle/backup/example002.bck tag=DUAL
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 18-JUN-13
```

```
RMAN> SELECT tag, encrypted FROM v$backup_piece;

TAG                              ENC
-------------------------------- ---
TRANSPARENT                      YES
DUAL                             YES
```

```
RMAN>
```

4. Back up the `EXAMPLE` tablespace using password encryption to `/home/oracle/backup/example003.bck`. Set `tag = password` so that it can be

specified in the `restore` command. To set encryption mode and password, use the following command:

```
SET ENCRYPTION ON IDENTIFIED BY "password1" only;
```

a. Set the password for encryption.

```
RMAN> set encryption on identified by "password1" only;

executing command: SET encryption

RMAN>
```

b. Use the RMAN `BACKUP` command to make a backup to `/home/oracle/backup/example003.bck`.

```
RMAN> backup tablespace example
      format '/home/oracle/backup/example003.bck'
      tag 'password';
2> 3>
Starting backup at 18-JUN-13
using channel ORA_DISK_1

channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00002
name=/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: starting piece 1 at 18-JUN-13
channel ORA_DISK_1: finished piece 1 at 18-JUN-13
piece handle=/home/oracle/backup/example003.bck tag=PASSWORD
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 18-JUN-13
```

```
RMAN> SELECT tag, encrypted FROM v$backup_piece;

TAG                              ENC
-------------------------------- ---
TRANSPARENT                      YES
DUAL                             YES
PASSWORD                         YES
```

```
RMAN> EXIT
$
```

5. Close the keystore.

```
$ sqlplus / as SYSKM
```

```
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE
                               IDENTIFIED BY secret;
  2
keystore altered.

SQL> EXIT
$
```

6. In another terminal session, remove the `EXAMPLE` tablespace file.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / AS SYSDBA

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> SELECT name FROM v$datafile;

NAME
-----------------------------------------------------------------
/u01/app/oracle/oradata/orcl/system01.dbf
/u01/app/oracle/oradata/orcl/example01.dbf
/u01/app/oracle/oradata/orcl/sysaux01.dbf
/u01/app/oracle/oradata/orcl/undotbs01.dbf
/u01/app/oracle/oradata/orcl/enctbs01.dbf
/u01/app/oracle/oradata/orcl/users01.dbf

6 rows selected.

SQL> EXIT
$ rm /u01/app/oracle/oradata/orcl/example01.dbf
$
```

7. Attempt to restore the example tablespace by using the backup made with transparent encryption. Why does it fail?

Attempt to restore the backup with the `transparent` tag. The keystore is closed. As a result, the encryption key is not available.

```
$ rman target '"john@orcl AS SYSBACKUP"'

target database Password:
connected to target database: ORCL (DBID=1345659572, not open)

RMAN> restore tablespace example from tag transparent;

Starting restore at 18-JUN-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=12 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00002 to
/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/example001.bck
RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
RMAN-00571:
===========================================================
RMAN-03002: failure of restore command at 06/18/2013 08:01:28
ORA-19870: error while restoring backup piece
/home/oracle/backup/example001.bck
ORA-19913: unable to decrypt backup
ORA-28365: wallet is not open

RMAN>
```

8. Restore the example tablespace by using password encryption.

   The restore from the password-only backup succeeds because the password is provided and the keystore is not needed.

```
RMAN> SET DECRYPTION IDENTIFIED BY "password1";

executing command: SET decryption

RMAN> restore tablespace example from tag "password";

Starting restore at 18-JUN-13
using channel ORA_DISK_1
```

```
channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00002 to
/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/example003.bck
channel ORA_DISK_1: piece
handle=/home/oracle/backup/example003.bck tag=PASSWORD
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:16
Finished restore at 18-JUN-13

RMAN>
```

9. In your second terminal session, once again remove the `EXAMPLE` tablespace datafile.

```
$ rm /u01/app/oracle/oradata/orcl/example01.dbf
$
```

10. Attempt to restore the example tablespace by using dual-mode encryption. Why does it fail?
The restore fails because the keystore is not open and the password is not set.

```
RMAN> restore tablespace example from tag dual;

Starting restore at 18-JUN-13
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00002 to
/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/example002.bck
RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
===============
RMAN-00571:
===========================================================
RMAN-03002: failure of restore command at 06/18/2013 08:19:27
ORA-19870: error while restoring backup piece
/home/oracle/backup/example002.bck
ORA-19913: unable to decrypt backup
ORA-28365: wallet is not open
```

```
RMAN>
```

11. Set the password for dual-mode backup and restore.

    To restore from dual-mode backup, either the password must be provided or the keystore must be open.

```
RMAN> SET DECRYPTION IDENTIFIED BY "oracle1";

executing command: SET decryption

RMAN> restore tablespace example from tag dual;

Starting restore at 18-JUN-13
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00002 to
/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/example002.bck
channel ORA_DISK_1: piece
handle=/home/oracle/backup/example002.bck tag=DUAL
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15
Finished restore at 18-JUN-13

RMAN> exit
$
```

12. In your second terminal session, again remove the datafile.

```
$ rm /u01/app/oracle/oradata/orcl/example01.dbf
$
```

13. Open the encryption keystore.

```
$ sqlplus / as SYSKM

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                               IDENTIFIED BY secret;
  2
```

```
keystore altered.

SQL> EXIT
$
```

14. Restore the example tablespace by using transparent encryption.
    Transparent mode encryption requires the keystore to be open.

```
$ rman target '"john@orcl AS SYSBACKUP"'

target database Password:
connected to target database: ORCL (DBID=1345659572, not open)
```

```
RMAN> restore tablespace example from tag transparent;

Starting restore at 18-JUN-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=11 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00002 to
/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/example001.bck
channel ORA_DISK_1: piece
handle=/home/oracle/backup/example001.bck tag=TRANSPARENT
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15
Finished restore at 18-JUN-13

RMAN>
```

15. In your second terminal session, again remove the datafile and close the terminal window.

```
$ rm /u01/app/oracle/oradata/orcl/example01.dbf
$ exit
```

16. Attempt to restore the example tablespace by using password-encrypted backup without supplying the password.
    The password-encrypted backup must have a password set in the session.

```
RMAN> restore tablespace example from tag "password";

Starting restore at 18-JUN-13
using channel ORA_DISK_1
```

```
channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00002 to
/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/example003.bck
RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
RMAN-00571:
===========================================================
RMAN-03002: failure of restore command at 06/18/2013 08:22:19
ORA-19870: error while restoring backup piece
/home/oracle/backup/example003.bck
ORA-19913: unable to decrypt backup

RMAN>
```

17. Restore dual-mode backup without a password.
    Dual-mode encrypted backup uses either the keystore or the password.

```
RMAN> restore tablespace example from tag dual;

Starting restore at 18-JUN-13
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00002 to
/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/example002.bck
channel ORA_DISK_1: piece
handle=/home/oracle/backup/example002.bck tag=DUAL
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15
Finished restore at 18-JUN-13

RMAN>
```

18. Recover the example tablespace, open the database, and then exit Recovery Manager.

```
RMAN> recover tablespace example;

Starting recover at 18-JUN-13
```

```
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:00

Finished recover at 18-JUN-13

RMAN> ALTER DATABASE OPEN;

Statement processed

RMAN> EXIT
$
```

Practice 20-2: Exporting Encrypted Data

Overview

In this practice, you will perform various Data Pump export operations using the different parameters for encryption. This will help you understand that you may export data in an unsecure manner.

Assumptions

The practice 19-1 successfully completed the creation of the password-based keystore in the `cdb1` and the generation of master keys for each PDB in `cdb1`.

Tasks

1. Execute the `$HOME/labs/ENC/create_tables_pdb1_1.sql` script to create a table with an encrypted column in `pdb1_1` pluggable database.

```
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> @$HOME/labs/ENC/create_tables_pdb1_1.sql
```

```
SQL> SET ECHO ON
SQL>
SQL> connect system/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL> ALTER USER oe IDENTIFIED BY oracle_4U ACCOUNT UNLOCK;

User altered.

SQL> grant create any directory to oe;

Grant succeeded.

SQL>
SQL> connect system/oracle_4U@localhost:1521/pdb1_2
Connected.
SQL> ALTER USER oe IDENTIFIED BY oracle_4U ACCOUNT UNLOCK ;
```

```
User altered.

SQL> grant create any directory to oe;

Grant succeeded.

SQL>
SQL> connect oe/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL> create directory dp as '/tmp';

Directory created.

SQL> connect oe/oracle_4U@localhost:1521/pdb1_2
Connected.
SQL> create directory dp as '/tmp';

Directory created.

SQL> connect oe/oracle_4U@localhost:1521/pdb1_1
Connected.
SQL> drop table cust_payment_info;
drop table cust_payment_info
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> create table cust_payment_info
  2    (first_name varchar2(11),
  3    last_name varchar2(10),
  4    order_number number(5),
  5    credit_card_number varchar2(20) ENCRYPT,
  6    active_card varchar2(3));

Table created.

SQL>
SQL> insert into cust_payment_info values
  2   ('Jon', 'Oldfield', 10001, 5105105105105100,'YES');
```

```
1 row created.

SQL> insert into cust_payment_info values
  2   ('Chris', 'White', 10002, 6011111111111117,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Alan', 'Squire', 10003, 378282246310005,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Mike', 'Anderson', 10004, 6011000000000004,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Annie', 'Schmidt', 10005, 4111111111111111,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Elliott', 'Meyer', 10006, 4222222222222,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Celine', 'Smith', 10007, 343434343434343,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Steve', 'Haslam', 10008, 6011000990139424,'YES');

1 row created.

SQL> insert into cust_payment_info values
  2   ('Albert', 'Einstein', 10009, 5111111111111118,'YES');

1 row created.
```

```
SQL>
SQL>  COMMIT;

Commit complete.

SQL> exit
$
```

2. Export the `OE.CUST_PAYMENT_INFO` table that holds one encrypted column.

```
$ expdp oe@pdb1_1 tables=cust_payment_info  directory=dp
REUSE_DUMPFILES=YES

Password: ******

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
Starting "OE"."SYS_EXPORT_TABLE_01":
oe/********@localhost:1521/pdb1_1 tables=cust_payment_info
directory=dp  REUSE_DUMPFILES=YES
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 64 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
. . exported "OE"."CUST_PAYMENT_INFO"                    7.179
KB       9 rows
ORA-39173: Encrypted data has been stored unencrypted in dump
file set.
Master table "OE"."SYS_EXPORT_TABLE_01" successfully
loaded/unloaded
******************************************************************
Dump file set for OE.SYS_EXPORT_TABLE_01 is:
  /tmp/expdat.dmp
Job "OE"."SYS_EXPORT_TABLE_01" completed with 1 error(s) at Thu
May 30 06:15:10 2013 elapsed 0 00:00:16

$
```

Notice the warning message: `ORA-39173: Encrypted data has been stored unencrypted in dump file set.`

This clearly warns you that the data exported from the `OE.CUST_PAYMENT_INFO` table is stored in clear text in the export dumpfile. The Data Pump export operation decrypted the data to export it into the dumpfile.

3. Use the dual encryption mode.

```
$ expdp oe@pdb1_1 tables=cust_payment_info encryption_mode=dual
directory=dp REUSE_DUMPFILES=YES

Password: ******

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
ORA-39002: invalid operation
ORA-39050: parameter ENCRYPTION is incompatible with parameter
ENCRYPTION_MODE

$
```

By default, the `ENCRYPTION` parameter, when not explicitly defined, sets the scope of encryption to columns only. This encryption scope is incompatible with dual mode encryption export.

4. Set the `ENCRYPTION` parameter explicitly to a compatible value.

```
$ expdp oe@pdb1_1 tables=cust_payment_info encryption_mode=dual
encryption=data_only directory=dp REUSE_DUMPFILES=YES

Password: ******

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
ORA-39002: invalid operation
ORA-39174: Encryption password must be supplied.

$
```

The `ENCRYPTION` parameter sets the scope of encryption to a value compatible with the encryption scope, but the dual mode requires the keystore to be opened and a password explicitly defined. The operation will export data only.

```
$ expdp oe@pdb1_1 tables=cust_payment_info encryption_mode=dual
encryption=data_only encryption_password="welcome1"
directory=dp dumpfile=reuse

Password: ******
```

```
Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
Starting "OE"."SYS_EXPORT_TABLE_01":
oe/********@localhost:1521/pdb1_1 tables=cust_payment_info
encryption_mode=dual encryption=data_only
encryption_password=******** directory=dp REUSE_DUMPFILES=YES
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 64 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
. . exported "OE"."CUST_PAYMENT_INFO"                    7.187
KB       9 rows
Master table "OE"."SYS_EXPORT_TABLE_01" successfully
loaded/unloaded
******************************************************************
Dump file set for OE.SYS_EXPORT_TABLE_01 is:
  /tmp/expdat.dmp
Job "OE"."SYS_EXPORT_TABLE_01" successfully completed at Thu May
30 06:39:29 2013 elapsed 0 00:00:08


$
```

5. Use the same parameters to export metadata only.

```
$ expdp oe@pdb1_1 tables=cust_payment_info encryption_mode=dual
encryption=metadata_only encryption_password="welcome1"
directory=dp REUSE_DUMPFILES=YES


Password: ******


Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
Starting "OE"."SYS_EXPORT_TABLE_01":
oe/********@localhost:1521/pdb1_1 tables=cust_payment_info
encryption_mode=dual encryption=metadata_only
encryption_password=******** directory=dp REUSE_DUMPFILES=YES
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 64 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
```

```
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
. . exported "OE"."CUST_PAYMENT_INFO"                    7.179
KB       9 rows
ORA-39173: Encrypted data has been stored unencrypted in dump
file set.
Master table "OE"."SYS_EXPORT_TABLE_01" successfully
loaded/unloaded
******************************************************************
Dump file set for OE.SYS_EXPORT_TABLE_01 is:
  /tmp/expdat.dmp
Job "OE"."SYS_EXPORT_TABLE_01" completed with 1 error(s) at Thu
May 30 06:48:04 2013 elapsed 0 00:00:06

$
```

Notice the warning message: `ORA-39173: Encrypted data has been stored unencrypted in dump file set.`

This clearly warns you that the data exported from the `OE.CUST_PAYMENT_INFO` table is stored in clear text in the export dumpfile. The Data Pump export operation kept encrypted the metadata only as requested in the command.

6. The `SYSKM` administrator decides to temporarily close the keystore for an administrative keystore maintenance task.

```
$ sqlplus / as SYSKM

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE
                               IDENTIFIED BY secret_cdb1
                               CONTAINER=ALL;
  2   3
keystore altered.

SQL> exit
$
```

7. Export in dual mode.

```
$ expdp oe@pdb1_1 tables=cust_payment_info encryption_mode=dual
encryption=data_only encryption_password="welcome1" directory=dp
REUSE_DUMPFILES=YES
Password: ******
```

```
Export: Release 12.1.0.1.0 - Production on Thu May 30 06:54:18
2013


Copyright (c) 1982, 2013, Oracle and/or its affiliates.  All
rights reserved.


Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
ORA-39002: invalid operation
ORA-39188: unable to encrypt dump file set
ORA-28365: wallet is not open
$
```

The dual mode requires that the keystore be opened.

8. The keystore is still closed but you need to export in a secure mode.
   a. Use the `PASSWORD` mode.

```
$ expdp oe@pdb1_1 tables=cust_payment_info
encryption_mode=password encryption_password="welcome1"
encryption_pwd_prompt=YES directory=dp REUSE_DUMPFILES=YES
Password: ******


Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
UDE-00011: parameter encryption_password is incompatible with
parameter encryption_pwd_prompt
$
```

`encryption_password` and `encryption_pwd_prompt=YES` are incompatible.

   b. Restart the operation without the password. Enter "`welcome1`" when prompted for the password.

```
$ expdp oe@pdb1_1 tables=cust_payment_info
encryption_mode=password ENCRYPTION_PWD_PROMPT=YES directory=dp
REUSE_DUMPFILES=YES
Password: ******


Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options


Encryption Password: ******
```

```
Starting "OE"."SYS_EXPORT_TABLE_01":
oe/********@localhost:1521/pdb1_1 tables=cust_payment_info
encryption_mode=password ENCRYPTION_PWD_PROMPT=YES directory=dp
REUSE_DUMPFILES=YES
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 64 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
ORA-31693: Table data object "OE"."CUST_PAYMENT_INFO" failed to
load/unload and is being skipped due to error:
ORA-29913: error in executing ODCIEXTTABLEPOPULATE callout
ORA-28365: wallet is not open
Master table "OE"."SYS_EXPORT_TABLE_01" successfully
loaded/unloaded
******************************************************************
Dump file set for OE.SYS_EXPORT_TABLE_01 is:
```

```
  /tmp/expdat.dmp
Job "OE"."SYS_EXPORT_TABLE_01" completed with 1 error(s) at Thu
May 30 06:59:29 2013 elapsed 0 00:00:10


$
```

`ENCRYPTION_PASSWORD` specifies a key for re-encrypting encrypted table columns so that they are not written as clear text in the dump file set.
Notice that the data has not been exported. The data needs to be decrypted during export using the keystore before being reencrypted into the dumpfile using the password. This requires the keystore to be opened.

c. Open the keystore and retry.

```
$ sqlplus / as SYSKM


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                               IDENTIFIED BY secret_cdb1
                               CONTAINER=ALL;
  2   3
keystore altered.

SQL> exit
```

```
$
```

Enter "`welcome1`" when prompted for the password.

```
$ expdp oe@pdb1_1 tables=cust_payment_info
encryption_mode=password ENCRYPTION_PWD_PROMPT=YES directory=dp
REUSE_DUMPFILES=YES


Password: ******


Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
```

```


Encryption Password: ******
Starting "OE"."SYS_EXPORT_TABLE_01":
oe/********@localhost:1521/pdb1_1 tables=cust_payment_info
encryption_mode=password ENCRYPTION_PWD_PROMPT=YES directory=dp
REUSE_DUMPFILES=YES
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 64 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
. . exported "OE"."CUST_PAYMENT_INFO"                    7.187
KB       9 rows
Master table "OE"."SYS_EXPORT_TABLE_01" successfully
loaded/unloaded
******************************************************************
Dump file set for OE.SYS_EXPORT_TABLE_01 is:
  /tmp/expdat.dmp
Job "OE"."SYS_EXPORT_TABLE_01" successfully completed at Thu May
30 07:18:10 2013 elapsed 0 00:00:09


$
```

Practice 20-3: Importing Encrypted Data

Overview

In this practice, you will import the `OE.CUST_PAYMENT_INFO` table that holds one encrypted column into another PDB of `cdb1`.

Assumptions

The last export operation successfully completed in the practice 20-2.

Tasks

1. The `SYSKM` administrator decides to temporarily close the keystore for an administrative keystore maintenance task.

```
$ sqlplus / as SYSKM

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE
                               IDENTIFIED BY secret_cdb1
                               CONTAINER=ALL;
  2   3
keystore altered.

SQL> exit
$
```

2. Import the `OE.CUST_PAYMENT_INFO` table into `pdb1_2` of `cdb1`. The `OE.CUST_PAYMENT_INFO` table does not exist in `pdb1_2`.
   If it exists, drop the table.

```
$ sqlplus system@pdb1_2

Enter password: ******
SQL> drop table oe.cust_payment_info;

Table dropped.

SQL> EXIT
$
```

   a. Use the `impdp` command.

```
$ impdp oe@pdb1_2 tables=cust_payment_info directory=dp
Password: ******
```

```
Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
ORA-39002: invalid operation
ORA-39174: Encryption password must be supplied.
$
```

b. The export operation used a password to encrypt data in the dumpfile. The import operation requires the same password to decrypt the data.

```
$ impdp oe@pdb1_2 tables=cust_payment_info
ENCRYPTION_PWD_PROMPT=YES directory=dp

Password: ******

Import: Release 12.1.0.1.0 - Production on Thu May 30 07:46:26
2013

Copyright (c) 1982, 2013, Oracle and/or its affiliates.  All
rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

Encryption Password:
ORA-39002: invalid operation
ORA-39176: Encryption password is incorrect.
$
```

c. Enter the same password ("`welcome1`") used by the export operation. If you use the wrong password, the import fails.

```
$ impdp oe@pdb1_2 tables=cust_payment_info
ENCRYPTION_PWD_PROMPT=YES directory=dp

Password: ******

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

Encryption Password: ******
Master table "OE"."SYS_IMPORT_TABLE_01" successfully
loaded/unloaded
```

```
Starting "OE"."SYS_IMPORT_TABLE_01":
oe/********@localhost:1521/pdb1_2 tables=cust_payment_info
ENCRYPTION_PWD_PROMPT=YES directory=dp
Processing object type TABLE_EXPORT/TABLE/TABLE
ORA-39083: Object type TABLE:"OE"."CUST_PAYMENT_INFO" failed to
create with error:
ORA-28365: wallet is not open
Failing sql is:
CREATE TABLE "OE"."CUST_PAYMENT_INFO" ("FIRST_NAME" VARCHAR2(11
BYTE), "LAST_NAME" VARCHAR2(10 BYTE), "ORDER_NUMBER"
NUMBER(5,0), "CREDIT_CARD_NUMBER" VARCHAR2(20 BYTE) ENCRYPT
USING 'AES192' 'SHA-1', "ACTIVE_CARD" VARCHAR2(3 BYTE)) SEGMENT
CREATION IMMEDIATE PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS NOLOGGING STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTEN
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "OE"."SYS_IMPORT_TABLE_01" completed with 1 error(s) at Thu
May 30 07:54:47 2013 elapsed 0 00:00:04

$
```

The table is created with a `CREDIT_CARD_NUMBER` column, which holds the `ENCRYPT` attribute. The password is required to decrypt the values of the `CREDIT_CARD_NUMBER` column stored in the dumpfile and require the keystore to be opened to re-encrypt the values in the data file where the table segment is stored.

d. Ask the `SYSKM` administrator to open the keystore.

```
$ sqlplus / as SYSKM

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                               IDENTIFIED BY secret_cdb1
                               CONTAINER=ALL;
  2   3
keystore altered.

SQL> exit
$
```

e. Reattempt the import operation.

```
$ impdp oe@pdb1_2 tables=cust_payment_info
ENCRYPTION_PWD_PROMPT=YES directory=dp

Password: ******
Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

Encryption Password: ******
Master table "OE"."SYS_IMPORT_TABLE_01" successfully
loaded/unloaded
Starting "OE"."SYS_IMPORT_TABLE_01":
oe/********@localhost:1521/pdb1_2 tables=cust_payment_info
ENCRYPTION_PWD_PROMPT=YES directory=dp
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "OE"."CUST_PAYMENT_INFO"                    7.187
KB       9 rows
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "OE"."SYS_IMPORT_TABLE_01" successfully completed at Thu May
30 07:50:23 2013 elapsed 0 00:00:10

$
```

3. Consider that the `OE.CUST_PAYMENT_INFO` table already existed into `pdb1_2` of `cdb1` without the `ENCRYPT` attribute.
   a. Drop and re-create the table without the `ENCRYPT` attribute.

```
$ sqlplus oe@pdb1_2
Enter password : ******
```

```
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL> DROP TABLE OE.CUST_PAYMENT_INFO PURGE;

Table dropped.

SQL> create table cust_payment_info
     ( first_name varchar2(11),
       last_name varchar2(10),
       order_number number(5),
       credit_card_number varchar2(20),
       active_card varchar2(3));
  2    3    4    5    6
Table created.

SQL>
```

b. The `SYSKM` administrator closes the keystore for maintenance operation.

```
SQL> CONNECT / as SYSKM
Connected.
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE
                               IDENTIFIED BY secret_cdb1
                               CONTAINER=ALL;
  2   3
keystore altered.

SQL> exit
$
```

c. Use the `impdp` command to import the `OE.CUST_PAYMENT_INFO` table.

```
$ impdp oe@pdb1_2 tables=cust_payment_info directory=dp

Password: ******

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options
ORA-39002: invalid operation
ORA-39174: Encryption password must be supplied.
```

```
$
```

d. The export operation used a password to encrypt data in the dumpfile. The import operation requires the same password to decrypt the data.

```
$ impdp oe@pdb1_2 tables=cust_payment_info
ENCRYPTION_PWD_PROMPT=YES directory=dp
TABLE_EXISTS_ACTION=truncate
Password: ******

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

Encryption Password: ******
Master table "OE"."SYS_IMPORT_TABLE_01" successfully
loaded/unloaded
Starting "OE"."SYS_IMPORT_TABLE_01":
oe/********@localhost:1521/pdb1_2 tables=cust_payment_info
ENCRYPTION_PWD_PROMPT=YES directory=dp
TABLE_EXISTS_ACTION=truncate
Processing object type TABLE_EXPORT/TABLE/TABLE
Table "OE"."CUST_PAYMENT_INFO" exists and has been truncated.
Data will be loaded but all dependent metadata will be skipped
due to table_exists_action of truncate
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "OE"."CUST_PAYMENT_INFO"                    7.187
KB       9 rows
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "OE"."SYS_IMPORT_TABLE_01" successfully completed at Thu May
30 08:27:12 2013 elapsed 0 00:00:06
$
```

Notice that even if the keystore is closed, the import operation does not need it. The password is sufficient to decrypt the data in the dumpfile. The decrypted data is not re-encrypted because the table does not hold any `ENCRYPT` column.

Practices for Lesson 21: Using Unified Auditing

Chapter 21

Practices for Lesson 21: Overview

Practices Overview

In the practices for this lesson, you enable unified audit, configure for Data Pump export auditing, and audit export and RMAN operations. You then view the audited data in the `UNIFIED_AUDIT_TRAIL` view.

Practice 21-1: Enabling Unified Auditing

Overview

In this practice, you enable unified auditing.

Tasks

1. Shut down all Oracle processes of all instances.
   a. Shut down the listener.

```
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base remains unchanged with value /u01/app/oracle
$
```

```
$ lsnrctl stop

LSNRCTL for Linux: Version 12.1.0.1.0 - Production on 30-MAY-
2013 15:18:54

Copyright (c) 1991, 2013, Oracle.  All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1521)))
The command completed successfully
$
```

   b. Shut down all instances.

```
$ pgrep -lf pmon
13266 ora_pmon_cdb1
20655 ora_pmon_em12rep
32139 ora_pmon_orcl
$
```

      1) Shut down the `orcl` instance.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
```

```
SQL> EXIT
$
```

2) Shut down the `cdb2` instance.

```
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base remains unchanged with value /u01/app/oracle
$
```

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> EXIT
$
```

3) Shut down the `em12rep` instance.

   a) Stop the OMS.

```
$ cd /u01/app/oracle/product/middleware/oms
$ export OMS_HOME=/u01/app/oracle/product/middleware/oms
$ $OMS_HOME/bin/emctl stop oms
Oracle Enterprise Manager Cloud Control 12c Release 2
Copyright (c) 1996, 2012 Oracle Corporation.  All rights
reserved.
Stopping WebTier...
WebTier Successfully Stopped
Stopping Oracle Management Server...
Oracle Management Server Successfully Stopped
Oracle Management Server is Down
$
```

   b) Shut down the repository database instance `em12rep`.

```
$ . oraenv
ORACLE_SID = [cdb1] ? em12rep
The Oracle base remains unchanged with value /u01/app/oracle
$
```

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> EXIT
$
```

4) Verify that all instances are down.

```
$ pgrep -lf pmon
$
```

2. Enable the Unified Audit option. Be cautious to copy the whole `make` command with the **`ORACLE_HOME=$ORACLE_HOME`** argument.

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk uniaud_on ioracle
ORACLE_HOME=$ORACLE_HOME
/usr/bin/ar d
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/libknlopt.a
kzanang.o
/usr/bin/ar cr
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/libknlopt.a
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/kzaiang.o
chmod 755 /u01/app/oracle/product/12.1.0/dbhome_1/bin


 - Linking Oracle
rm -f /u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/oracle
/u01/app/oracle/product/12.1.0/dbhome_1/bin/orald  -o
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/oracle -m64 -z
noexecstack -L/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/
-L/u01/app/oracle/product/12.1.0/dbhome_1/lib/ -
L/u01/app/oracle/product/12.1.0/dbhome_1/lib/stubs/   -Wl,-E
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/opimai.o
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/ssoraed.o
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/ttcsoi.o -Wl,-
-whole-archive -lperfsrv12 -Wl,--no-whole-archive
/u01/app/oracle/product/12.1.0/dbhome_1/lib/nautab.o
/u01/app/oracle/product/12.1.0/dbhome_1/lib/naeet.o
/u01/app/oracle/product/12.1.0/dbhome_1/lib/naect.o
/u01/app/oracle/product/12.1.0/dbhome_1/lib/naedhs.o
```

```
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/config.o  -
lserver12 -lodm12 -lcell12 -lnnet12 -lskgxp12 -lsnls12 -lnls12
-lcore12 -lsnls12 -lnls12 -lcore12 -lsnls12 -lnls12 -lxml12 -
lcore12 -lunls12 -lsnls12 -lnls12 -lcore12 -lnls12 -lclient12  -
lvsn12 -lcommon12 -lgeneric12 -lknlopt `if /usr/bin/ar tv
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/libknlopt.a |
grep xsyeolap.o > /dev/null 2>&1 ; then echo "-loraolap12" ; fi`
-lskjcx12 -lslax12 -lpls12  -lrt -lplp12 -lserver12 -lclient12
-lvsn12 -lcommon12 -lgeneric12 `if [ -f
/u01/app/oracle/product/12.1.0/dbhome_1/lib/libavserver12.a ] ;
then echo "-lavserver12" ; else echo "-lavstub12"; fi` `if [ -f
/u01/app/oracle/product/12.1.0/dbhome_1/lib/libavclient12.a ] ;
then echo "-lavclient12" ; fi` -lknlopt -lslax12 -lpls12  -lrt -
lplp12 -ljavavm12 -lserver12  -lwwg  `cat
/u01/app/oracle/product/12.1.0/dbhome_1/lib/ldflags`    -
lncrypt12 -lnsgr12 -lnzjs12 -ln12 -lnl12 -lnro12 `cat
/u01/app/oracle/product/12.1.0/dbhome_1/lib/ldflags`    -
lncrypt12 -lnsgr12 -lnzjs12 -ln12 -lnl12 -lnnz12 -lzt12 -lztkg12
-lmm -lsnls12 -lnls12  -lcore12 -lsnls12 -lnls12 -lcore12 -
lsnls12 -lnls12 -lxml12 -lcore12 -lunls12 -lsnls12 -lnls12 -
lcore12 -lnls12 -lztkg12 `cat
/u01/app/oracle/product/12.1.0/dbhome_1/lib/ldflags`    -
lncrypt12 -lnsgr12 -lnzjs12 -ln12 -lnl12 -lnro12 `cat
/u01/app/oracle/product/12.1.0/dbhome_1/lib/ldflags`    -
lncrypt12 -lnsgr12 -lnzjs12 -ln12 -lnl12 -lnnz12 -lzt12 -lztkg12
-lsnls12 -lnls12  -lcore12 -lsnls12 -lnls12 -lcore12 -lsnls12 -
lnls12 -lxml12 -lcore12 -lunls12 -lsnls12 -lnls12 -lcore12 -
lnls12 `if /usr/bin/ar tv
/u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/libknlopt.a |
grep "kxmnsd.o" > /dev/null 2>&1 ; then echo " " ; else echo "-
lordsdo12"; fi` -
L/u01/app/oracle/product/12.1.0/dbhome_1/ctx/lib/ -lctxc12 -
lctx12 -lzx12 -lgx12 -lctx12 -lzx12 -lgx12 -lordimt12 -lclsra12
-ldbcfg12 -lhasgen12 -lskgxn2 -lnnz12 -lzt12 -lxml12 -locr12 -
locrb12 -locrutl12 -lhasgen12 -lskgxn2 -lnnz12 -lzt12 -lxml12  -
lgeneric12 -loraz -llzopro -lorabz2 -lipp_z -lipp_bz2 -
lippdcemerged -lippsemerged -lippdcmerged  -lippsmerged -
lippcore  -lippcpemerged -lippcpmerged  -lsnls12 -lnls12  -
lcore12 -lsnls12 -lnls12 -lcore12 -lsnls12 -lnls12 -lxml12 -
lcore12 -lunls12 -lsnls12 -lnls12 -lcore12 -lnls12 -lsnls12 -
lunls12  -lsnls12 -lnls12  -lcore12 -lsnls12 -lnls12 -lcore12 -
lsnls12 -lnls12 -lxml12 -lcore12 -lunls12 -lsnls12 -lnls12 -
lcore12 -lnls12 -lasmclnt12 -lcommon12 -lcore12  -laio -lons
`cat /u01/app/oracle/product/12.1.0/dbhome_1/lib/sysliblist` -
Wl,-rpath,/u01/app/oracle/product/12.1.0/dbhome_1/lib -lm
`cat /u01/app/oracle/product/12.1.0/dbhome_1/lib/sysliblist` -
ldl -lm   -L/u01/app/oracle/product/12.1.0/dbhome_1/lib
test ! -f /u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle ||\
           mv -f
/u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle
/u01/app/oracle/product/12.1.0/dbhome_1/bin/oracleO
```

```
mv /u01/app/oracle/product/12.1.0/dbhome_1/rdbms/lib/oracle
/u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle
chmod 6751 /u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle
$
```

3. Restart the processes.
   a. Restart the database `orcl` only. A later practice requires the database to be in `ARCHIVELOG` mode: set the `ARCHIVELOG` mode now.

```
$ . oraenv
ORACLE_SID = [em12rep] ? orcl
The Oracle base remains unchanged with value /u01/app/oracle
$
```

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Thu May 30 15:27:15
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup mount
ORACLE instance started.

Total System Global Area 501059584 bytes
Fixed Size               2290024 bytes
Variable Size          264244888 bytes
Database Buffers       226492416 bytes
Redo Buffers             8032256 bytes
Database mounted.

SQL> ALTER DATABASE ARCHIVELOG;

Database altered.

SQL> ALTER DATABASE OPEN;

Database altered.

SQL> EXIT
$
```

   b. Verify that unified auditing is enabled. You can see that the Unified Auditing option is enabled in the SQL*Plus banner.

```
$ sqlplus / as sysdba
```

```
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing and Unified Auditing options

SQL> COL  parameter FORMAT A20
SQL> COL  value     FORMAT A20
SQL> select parameter , value
     from v$option
     where PARAMETER = 'Unified Auditing';

PARAMETER            VALUE
-------------------- --------------------
Unified Auditing     TRUE

SQL>
```

4. Check the existence of the predefined `ORA_SECURECONFIG` audit policy.

```
SQL> COL  POLICY_NAME FORMAT A20
SQL> COL  AUDIT_OPTION FORMAT A40
SQL> set PAGES 100
SQL> select POLICY_NAME, AUDIT_OPTION
     from   AUDIT_UNIFIED_POLICIES
     where  policy_name =  'ORA_SECURECONFIG'  order by 2 ;
  2    3
POLICY_NAME          AUDIT_OPTION
-------------------- ----------------------------------------
ORA_SECURECONFIG     ADMINISTER KEY MANAGEMENT
ORA_SECURECONFIG     ALTER ANY PROCEDURE
ORA_SECURECONFIG     ALTER ANY SQL TRANSLATION PROFILE
ORA_SECURECONFIG     ALTER ANY TABLE
ORA_SECURECONFIG     ALTER DATABASE
ORA_SECURECONFIG     ALTER DATABASE LINK
ORA_SECURECONFIG     ALTER PLUGGABLE DATABASE
ORA_SECURECONFIG     ALTER PROFILE
ORA_SECURECONFIG     ALTER ROLE
ORA_SECURECONFIG     ALTER SYSTEM
ORA_SECURECONFIG     ALTER USER
ORA_SECURECONFIG     AUDIT SYSTEM
ORA_SECURECONFIG     CREATE ANY JOB
ORA_SECURECONFIG     CREATE ANY LIBRARY
```

```
ORA_SECURECONFIG     CREATE ANY PROCEDURE
ORA_SECURECONFIG     CREATE ANY SQL TRANSLATION PROFILE
ORA_SECURECONFIG     CREATE ANY TABLE
ORA_SECURECONFIG     CREATE DATABASE LINK
ORA_SECURECONFIG     CREATE DIRECTORY
ORA_SECURECONFIG     CREATE EXTERNAL JOB
ORA_SECURECONFIG     CREATE PLUGGABLE DATABASE
ORA_SECURECONFIG     CREATE PROFILE
ORA_SECURECONFIG     CREATE PUBLIC SYNONYM
ORA_SECURECONFIG     CREATE ROLE
ORA_SECURECONFIG     CREATE SQL TRANSLATION PROFILE
ORA_SECURECONFIG     CREATE USER
ORA_SECURECONFIG     DROP ANY PROCEDURE
ORA_SECURECONFIG     DROP ANY SQL TRANSLATION PROFILE
ORA_SECURECONFIG     DROP ANY TABLE
ORA_SECURECONFIG     DROP DATABASE LINK
ORA_SECURECONFIG     DROP DIRECTORY
ORA_SECURECONFIG     DROP PLUGGABLE DATABASE
ORA_SECURECONFIG     DROP PROFILE
ORA_SECURECONFIG     DROP PUBLIC SYNONYM
ORA_SECURECONFIG     DROP ROLE
ORA_SECURECONFIG     DROP USER
ORA_SECURECONFIG     EXEMPT ACCESS POLICY
ORA_SECURECONFIG     EXEMPT REDACTION POLICY
ORA_SECURECONFIG     GRANT ANY OBJECT PRIVILEGE
ORA_SECURECONFIG     GRANT ANY PRIVILEGE
ORA_SECURECONFIG     GRANT ANY ROLE
ORA_SECURECONFIG     LOGMINING
ORA_SECURECONFIG     LOGOFF
ORA_SECURECONFIG     LOGON
ORA_SECURECONFIG     PURGE DBA_RECYCLEBIN
ORA_SECURECONFIG     SET ROLE
ORA_SECURECONFIG     TRANSLATE ANY SQL

47 rows selected.

SQL>
```

5. Verify that the predefined `ORA_SECURECONFIG` audit policy is enabled by default.

```
SQL> select POLICY_NAME
     from   AUDIT_UNIFIED_ENABLED_POLICIES
     where  policy_name = 'ORA_SECURECONFIG';
 2    3
```

```
POLICY_NAME
--------------------
ORA_SECURECONFIG

SQL>
```

6. Are users connections still audited?

```
SQL> connect hr
Enter password: ******
Connected.
SQL> connect hr
Enter password: ******
Connected.
SQL> connect / as sysdba
Connected.
SQL> col dbusername format A20
SQL> col action_name format A20
SQL> select action_name, dbusername
     from   unified_audit_trail
     where  dbusername='HR';
  2    3

ACTION_NAME          DBUSERNAME
-------------------- --------------------
LOGON                HR
LOGON                HR
… rows deleted.
LOGOFF               HR
LOGOFF               HR
… rows deleted.

SQL> exit
$
```

7. Restart the listener.

```
$ lsnrctl start

Starting /u01/app/oracle/product/12.1.0/dbhome_1/bin/tnslsnr:
please wait...

TNSLSNR for Linux: Version 12.1.0.1.0 - Production
System parameter file is
/u01/app/oracle/product/12.1.0/dbhome_1/network/admin/listener.o
ra
```

```
Log messages written to
/u01/app/oracle/diag/tnslsnr/<YourServer>/listener/alert/log.xml
Listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
Listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<YourServer>)(PORT=152
1)))

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 12.1.0.1.0
- Production
Start Date                30-MAY-2013 15:28:59
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/app/oracle/product/12.1.0/dbhome_1/network/admin/listener.o
ra
Listener Log File
/u01/app/oracle/diag/tnslsnr/<YourServer>/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<YourServer>)(PORT=152
1)))
The listener supports no services
The command completed successfully
$
```

Practice 21-2: Creating and Enabling Audit Policies

Overview

In this practice, the security officer, the SEC user, will create audit policies to audit privileges, actions and roles under defined conditions.

Tasks

1. Grant the SEC user the AUDIT_ADMIN role to allow him to manage audit policies.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing and Unified Auditing options

SQL> GRANT audit_admin to SEC;

Grant succeeded.

SQL>
```

2. Connect as SEC to create an audit policy that will audit the OE user using the SELECT ANY TABLE or CREATE LIBRARY system privileges and this for each statement executed.
   a. Grant the SELECT ANY TABLE to the OE and HR users. In case the users are already granted the SELECT object privilege on the SH.SALES or HR.EMPLOYEES tables, revoke the object privileges.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> REVOKE select ON sh.sales FROM hr;

REVOKE select ON sh.sales FROM hr
*
ERROR at line 1:
ORA-01927: cannot REVOKE privileges you did not grant

SQL> REVOKE select ON hr.employees FROM oe;

REVOKE select ON hr.employees FROM oe
*
ERROR at line 1:
ORA-01927: cannot REVOKE privileges you did not grant
```

```
SQL> GRANT select any table TO oe, hr;

Grant succeeded.

SQL>
```

b. Create the audit policy.

```
SQL> CREATE AUDIT POLICY aud_syspriv_pol
            PRIVILEGES select any table, create library
     WHEN 'SYS_CONTEXT(''USERENV'',''SESSION_USER'')=''OE'''
     EVALUATE PER STATEMENT;
2    3    4
Audit policy created.

SQL>
```

c. Enable the audit policy.

```
SQL> AUDIT POLICY aud_syspriv_pol;

Audit succeeded.

SQL>
```

d. View the audit policy options.

```
SQL> col audit_option format A17
SQL> col policy_name format A16
SQL> col audit_condition format A42
SQL> SELECT POLICY_NAME, AUDIT_OPTION, AUDIT_CONDITION
     FROM   AUDIT_UNIFIED_POLICIES
     WHERE  POLICY_NAME ='AUD_SYSPRIV_POL';
  2    3
POLICY_NAME       AUDIT_OPTION
---------------- ---------------
AUDIT_CONDITION
------------------------------------------
AUD_SYSPRIV_POL  CREATE LIBRARY
SYS_CONTEXT('USERENV','SESSION_USER')='OE'

AUD_SYSPRIV_POL  SELECT ANY TABLE
SYS_CONTEXT('USERENV','SESSION_USER')='OE'

SQL>
```

e. Verify that the audit policy is enabled.

```
SQL> col user_name format A10
SQL> SELECT POLICY_NAME, ENABLED_OPT, USER_NAME, SUCCESS,
FAILURE
```

```
     FROM   AUDIT_UNIFIED_ENABLED_POLICIES
     WHERE  POLICY_NAME ='AUD_SYSPRIV_POL';
 2    3
POLICY_NAME         ENABLED_ USER_NAME  SUC FAI
------------------ -------- ---------- --- ---
AUD_SYSPRIV_POL    BY       ALL USERS  YES YES

SQL>
```

f. Connect as `HR` and then as `OE` and perform actions that require the `SELECT ANY TABLE` or `CREATE LIBRARY` system privileges.

```
SQL> connect hr
Enter password: ******
Connected.
SQL> select count(*) from sh.sales;

  COUNT(*)
----------
    918843

SQL> connect oe
Enter password: ******
Connected.
SQL> select last_name from hr.employees;

LAST_NAME
-------------------------
… rows deleted

Urman
Vargas
Vishney
Vollman
Walsh
Weiss
Whalen
Zlotkey

83 rows selected.

SQL>
```

g. View the resulting audit data.

```
SQL> connect sec
Enter password: ******
```

```
Connected.
SQL> col action_name format A16
SQL> col policy_name format A18
SQL> col system_privilege_used format A20
SQL> select DBUSERNAME, ACTION_NAME, SYSTEM_PRIVILEGE_USED
     from   unified_audit_trail
     where  DBUSERNAME in ('HR','OE')
     and    action_name not in ('LOGON','LOGOFF');
  2    3    4

DBUSERNAME                      ACTION_NAME    SYSTEM_PRIVILEGE_USE
------------------------------ -------------- --------------------
OE                              SELECT         SELECT ANY TABLE

… rows deleted

SQL>
```

Notice that the action executed by the `HR` user has not been audited. Only `OE` was configured in the audit policy.

3. Create an audit policy that will audit any user performing any `SELECT` or `UPDATE` operation on any object using an object or system privilege, or deleting rows from the `HR.CODE` table.
   a. Create the audit policy.

```
SQL> CREATE AUDIT POLICY aud_action_pol
             ACTIONS select, update, delete ON hr.code;
   2
Audit policy created.

SQL>
```

   b. Enable the audit policy for all users except `OE`.

```
SQL> AUDIT POLICY aud_action_pol EXCEPT oe;

Audit succeeded.

SQL>
```

   c. View the audit policy options.

```
SQL> col audit_option format A17
SQL> col policy_name format A16
SQL> col audit_condition format A42
SQL> SELECT POLICY_NAME, AUDIT_OPTION, AUDIT_CONDITION
     FROM   AUDIT_UNIFIED_POLICIES
     WHERE  POLICY_NAME ='AUD_ACTION_POL';
  2    3
POLICY_NAME        AUDIT_OPTION      AUDIT_CONDITION
```

```
---------------- ---------------- -------------------------
AUD_ACTION_POL    SELECT           NONE
AUD_ACTION_POL    UPDATE           NONE
AUD_ACTION_POL    DELETE           NONE

SQL>
```

d. Verify that the audit policy is enabled.

```
SQL> col user_name format A10
SQL> SELECT POLICY_NAME, ENABLED_OPT, USER_NAME, SUCCESS,
FAILURE
     FROM   AUDIT_UNIFIED_ENABLED_POLICIES
     WHERE  POLICY_NAME ='AUD_ACTION_POL';
  2    3
POLICY_NAME       ENABLED_ USER_NAME  SUC FAI
----------------- -------- ---------- --- ---
AUD_ACTION_POL    EXCEPT   OE         YES YES

SQL>
```

e. Perform an audit operation. First create a new user `DEV` and grant appropriate privileges to `DEV` to execute operations.

```
SQL> CREATE USER dev IDENTIFIED BY oracle_4U;

User created.

SQL> GRANT create session TO dev;

Grant succeeded.

SQL> CONNECT hr
Enter password: ******
Connected.
SQL> GRANT delete on hr.code TO dev;

Grant succeeded.

SQL> CONNECT dev
Enter password: ******
Connected.
SQL> DELETE hr.code WHERE rownum=1;

1 row deleted.

SQL> COMMIT;
```

```
Commit complete.

SQL> CONNECT oe
Enter password: ******
Connected.
SQL> SELECT count(*) FROM hr.employees;

  COUNT(*)
----------
        83

SQL>
```

f. View the resulting audit data.

```
SQL> connect sec
Enter password: ******
Connected.
SQL> set pages 100
SQL> col dbusername format A8
SQL> col action_name format A8
SQL> col unified_audit_policies format a40
SQL> SELECT UNIFIED_AUDIT_POLICIES, DBUSERNAME, ACTION_NAME
     FROM   unified_audit_trail
     WHERE  dbusername in ('DEV','OE')
     AND    action_name not in ('LOGON', 'LOGOFF')
     AND    unified_audit_policies like '%ACTION%';
  2    3    4    5
UNIFIED_AUDIT_POLICIES                 DBUSERNA ACTION_N
---------------------------------------- -------- --------
… rows deleted

AUD_ACTION_POL                         DEV    SELECT
AUD_ACTION_POL, AUD_ACTION_POL         DEV    SELECT
AUD_ACTION_POL, AUD_ACTION_POL         DEV    SELECT
AUD_ACTION_POL                         DEV    DELETE

13 rows selected.

SQL>
```

Notice that OE was excluded from the auditing process.

4. Create an audit policy that will audit all users while using the `MGR_ROLE` role.

a. Create a new user if the user does not exist yet. Create a new role and grant the new role to `jim`.

```
SQL> DROP USER jim;

User dropped.

SQL> CREATE USER jim IDENTIFIED BY oracle_4U;

User created.
```

```
SQL> CREATE ROLE mgr_role;

Role created.

SQL> GRANT create tablespace TO mgr_role;

Grant succeeded.

SQL> GRANT mgr_role, create session TO jim;

Grant succeeded.

SQL>
```

b. Create the audit policy.

```
SQL> CREATE AUDIT POLICY aud_role_pol ROLES mgr_role;

Audit policy created.

SQL>
```

c. Enable the audit policy.

```
SQL> AUDIT POLICY aud_role_pol WHENEVER SUCCESSFUL;

Audit succeeded.

SQL>
```

d. Create another audit policy that will audit all users as soon as these users use the `DBA` role.

1) Create a `DBA_JUNIOR` user granted the `DBA` role.

```
SQL> CREATE USER dba_junior IDENTIFIED BY oracle_4U;

User created.

SQL> GRANT dba TO dba_junior;

Grant succeeded.

SQL>
```

2) Create the audit policy.

```
SQL> CREATE AUDIT POLICY aud_dba_pol ROLES dba;

Audit policy created.
```

```
SQL>
```

3) Enable the audit policy.

```
SQL> AUDIT POLICY aud_dba_pol WHENEVER SUCCESSFUL;

Audit succeeded.

SQL>
```

e. View the audit policy options.

```
SQL> col audit_option format A20
SQL> col policy_name format A18
SQL> SELECT POLICY_NAME, AUDIT_OPTION, CONDITION_EVAL_OPT
     FROM   AUDIT_UNIFIED_POLICIES
     WHERE  POLICY_NAME in ('AUD_ROLE_POL','AUD_DBA_POL');
  2    3
POLICY_NAME        AUDIT_OPTION         CONDITION
------------------ -------------------- ---------
AUD_ROLE_POL       MGR_ROLE             NONE
AUD_DBA_POL        DBA                  NONE

SQL>
```

f. Verify that the audit policy is enabled.

```
SQL> col user_name format A10
SQL> SELECT POLICY_NAME, ENABLED_OPT, USER_NAME, SUCCESS,
FAILURE
     FROM   AUDIT_UNIFIED_ENABLED_POLICIES
     WHERE  POLICY_NAME in ('AUD_ROLE_POL','AUD_DBA_POL');
  2    3
POLICY_NAME        ENABLED_  USER_NAME  SUC FAI
------------------ -------- ---------- --- ---
AUD_ROLE_POL       BY        ALL USERS  YES NO
AUD_DBA_POL        BY        ALL USERS  YES NO

SQL>
```

g. Perform an audit operation for both role type audited policies.

```
SQL> CONNECT jim
Enter password: ******
Connected.
SQL> CREATE TABLESPACE test DATAFILE '/tmp/test01.dbf' size 10m;

Tablespace created.

SQL> CONNECT dba_junior
```

```
Enter password: ******
Connected.
SQL> ALTER SYSTEM SET job_queue_processes=200;

System altered.

SQL> ALTER SYSTEM SET job_queue_processes=100;

System altered.

SQL>
```

h. View the resulting audit data.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> col dbusername format A10
SQL> col action_name format A18
SQL> col unified_audit_policies format a30
SQL> SELECT UNIFIED_AUDIT_POLICIES, DBUSERNAME,
            ACTION_NAME, SYSTEM_PRIVILEGE_USED
     FROM   unified_audit_trail
     WHERE  DBUSERNAME in ('JIM','DBA_JUNIOR')
     AND    ACTION_NAME not in ('LOGON', 'LOGOFF')
     AND    (UNIFIED_AUDIT_POLICIES like '%AUD_ROLE_POL%'
     OR     UNIFIED_AUDIT_POLICIES like '%AUD_DBA_POL%');
  2    3   4   5   6   7
UNIFIED_AUDIT_POLICIES         DBUSERNAME ACTION_NAME
SYSTEM_PRIVILEGE_U
------------------------------ ---------- ------------------
------------------
rows deleted …

AUD_ROLE_POL, AUD_DBA_POL      JIM         CREATE TABLESPACE
CREATE TABLESPACE

ORA_SECURECONFIG, AUD_DBA_POL  DBA_JUNIOR  ALTER SYSTEM
ALTER SYSTEM

ORA_SECURECONFIG, AUD_DBA_POL  DBA_JUNIOR  ALTER SYSTEM
ALTER SYSTEM

SQL>
```

The first row displays `AUD_ROLE_POL`, `AUD_DBA_POL` in the `UNIFIED_AUDIT_POLICIES`. Both policies track the `CREATE TABLESPACE` system privilege.

5. *(Optional: if you skip this task, then go to practice 21-3)* Create an audit policy that will audit all users while using the `STORAGE_ROLE` role or performing any action related to tables.
   a. Create a new role and grant this new role to `DEV` and grant `DROP ANY TABLE` to `JIM`.

```
SQL> CREATE ROLE storage_role;

Role created.

SQL> GRANT drop tablespace TO storage_role;

Grant succeeded.

SQL> GRANT storage_role TO dev;

Grant succeeded.

SQL> GRANT drop any table TO jim;

Grant succeeded.

SQL>
```

   b. Create and enable the audit policy.

```
SQL> CREATE AUDIT POLICY aud_mixed_pol
          ACTIONS create table, drop table, truncate table
          ROLES   storage_role;
  2    3
Audit policy created.

SQL> AUDIT POLICY aud_mixed_pol;

Audit succeeded.

SQL>
```

   c. Verify that the audit policy is enabled.

```
SQL> SELECT * FROM AUDIT_UNIFIED_ENABLED_POLICIES
     WHERE  POLICY_NAME like '%MIXED%';
  2
USER_NAME  POLICY_NAME        ENABLED_ SUC FAI
---------- ------------------ -------- --- ---
ALL USERS  AUD_MIXED_POL      BY       YES YES

SQL>
```

   d. Perform operations.

```
SQL> CONNECT dev
Enter password: ******
Connected.
SQL> DROP TABLESPACE test including contents and datafiles;

Tablespace dropped.

SQL> CONNECT jim
Enter password: ******
Connected.
SQL> DROP TABLE hr.code purge;

Table dropped.

SQL>
```

e. View the resulting audit data.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> col unified_audit_policies format A44
SQL> SELECT UNIFIED_AUDIT_POLICIES, DBUSERNAME,
            ACTION_NAME, SYSTEM_PRIVILEGE_USED
     FROM   unified_audit_trail
     WHERE  DBUSERNAME in ('JIM','DEV')
     AND    UNIFIED_AUDIT_POLICIES like '%AUD_MIXED_POL%'
     AND    ACTION_NAME not in ('LOGON', 'LOGOFF');
  2    3    4    5    6

UNIFIED_AUDIT_POLICIES                      DBUSERNAME
ACTION_NAME        SYSTEM_PRIVILEGE_U
-------------------------------------------- ----------
----------------- -----------------
AUD_MIXED_POL, ORA_SECURECONFIG, AUD_DBA_POL JIM
DROP TABLE         DROP ANY TABLE

AUD_DBA_POL, AUD_MIXED_POL                  DEV
DROP TABLESPACE    DROP TABLESPACE

SQL>
```

Practice 21-3: Cleaning Up Audit Policies and Data

Overview

In this practice, you will drop the audit policies and the audit trail data.

Tasks

1. Drop the audit polices created in the previous practices.
   a. Display the list of audit policies.

```
SQL> col policy_name FORMAT A30
SQL> SELECT distinct policy_name FROM AUDIT_UNIFIED_POLICIES;

POLICY_NAME
------------------------------
AUD_ROLE_POL
AUD_SYSPRIV_POL
ORA_RAS_POLICY_MGMT
ORA_DATABASE_PARAMETER
ORA_RAS_SESSION_MGMT
ORA_ACCOUNT_MGMT
AUD_ACTION_POL
AUD_MIXED_POL
AUD_DBA_POL
ORA_SECURECONFIG

10 rows selected.

SQL>
```

   b. Drop each audit policy.

```
SQL> DROP AUDIT POLICY aud_role_pol;
DROP AUDIT POLICY aud_role_pol
*
ERROR at line 1:
ORA-46361: Audit policy cannot be dropped as it is currently
enabled.

SQL>
```

   Notice that an enabled audit policy cannot be dropped. First disable it.

```
SQL> NOAUDIT POLICY aud_role_pol;

Noaudit succeeded.

SQL> DROP AUDIT POLICY aud_role_pol;
```

```
Audit Policy dropped.

SQL> NOAUDIT POLICY aud_syspriv_pol;

Noaudit succeeded.

SQL> DROP AUDIT POLICY aud_syspriv_pol;

Audit Policy dropped.

SQL> NOAUDIT POLICY aud_action_pol;

Noaudit succeeded.

SQL> DROP AUDIT POLICY aud_action_pol;

Audit Policy dropped.

SQL> NOAUDIT POLICY aud_mixed_pol;

Noaudit succeeded.

SQL> DROP AUDIT POLICY aud_mixed_pol;

Audit Policy dropped.

SQL> NOAUDIT POLICY aud_dba_pol;

Noaudit succeeded.

SQL> DROP AUDIT POLICY aud_dba_pol;

Audit Policy dropped.

SQL>
```

2. Clean up the audit trail data.
   a. You can perform the cleanup manually.

```
SQL> SELECT count(*) FROM unified_audit_trail;

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

```
     21163

SQL> exec DBMS_AUDIT_MGMT.SET_LAST_ARCHIVE_TIMESTAMP ( -
    AUDIT_TRAIL_TYPE  => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED, -
    LAST_ARCHIVE_TIME => sysdate)
> >
PL/SQL procedure successfully completed.

SQL> exec DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL( -
    AUDIT_TRAIL_TYPE  => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED, -
    USE_LAST_ARCH_TIMESTAMP => TRUE)
> >
```

The cleanup may last several minutes. If you attempt to connect as the `SEC` user or even `AS SYSDBA` in another session, the session will be hanging until the cleanup process completes. The audit is locked during the cleanup.

```
PL/SQL procedure successfully completed.

SQL> SELECT count(*) FROM unified_audit_trail;

  COUNT(*)
----------
       358

SQL>
```

b. You can also schedule the cleanup as follows.

```
SQL> exec DBMS_AUDIT_MGMT.CREATE_PURGE_JOB (-
       AUDIT_TRAIL_TYPE=>DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED, -
       AUDIT_TRAIL_PURGE_INTERVAL => 1, -
       AUDIT_TRAIL_PURGE_NAME => 'Audit_Trail_PJ', -
       USE_LAST_ARCH_TIMESTAMP => TRUE)
> > > >
PL/SQL procedure successfully completed.

SQL>
```

c. View the cleanup job executions.

```
SQL> col JOB_NAME format A14
SQL> col STATUS format A12
SQL> col ACTUAL_START_DATE format A40

SQL> SELECT  JOB_NAME, STATUS, ACTUAL_START_DATE
     FROM    dba_scheduler_job_run_details
```

```
     WHERE   JOB_NAME='AUDIT_TRAIL_PJ'
     ORDER BY ACTUAL_START_DATE;
  2    3    4
JOB_NAME       STATUS            ACTUAL_START_DATE
-------------- ------------ ----------------------------------
AUDIT_TRAIL_PJ SUCCEEDED    31-MAY-13 05.18.39.227394 AM ETC/UTC

SQL>
```

Practice 21-4: Auditing SYS User *(Optional)*

Overview

In this practice, you will audit actions performed by the `SYS` user in `orcl`, which are not audited by the `ORA_SECURECONFIG` predefined audit policy.

Tasks

1. Create and enable an audit policy that audits any update or select action on the `HR.EMPLOYEES` table.
   a. Still connected as the security officer, create, enable and display the audit policy that audits any update action on the `HR.EMPLOYEES` table.

```
SQL> CREATE AUDIT POLICY aud_sys_pol
            ACTIONS update ON hr.employees,
                    select ON hr.employees;
2   3
Audit policy created.

SQL> AUDIT POLICY aud_sys_pol BY sys WHENEVER SUCCESSFUL;

Audit succeeded.

SQL> SELECT * FROM AUDIT_UNIFIED_ENABLED_POLICIES
     WHERE  POLICY_NAME like '%SYS%';
  2
USER_NAME   POLICY_NAME         ENABLED_ SUC FAI
----------- ------------------- -------- --- ---
SYS         AUD_SYS_POL         BY       YES NO

SQL>
```

   b. Connect as `SYS` and execute an update command and a select command on the `HR.EMPLOYEES` table.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> UPDATE hr.employees SET salary=salary+100
     WHERE last_name='Me';
  2
0 rows updated.

SQL> SELECT max(salary) FROM hr.employees;

MAX(SALARY)
-----------
      24000
```

```
SQL>
```

c. Connect as the security officer to display the audited actions.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> col dbusername format A10
SQL> col action_name format A12
SQL> col system_privilege_used FORMAT A30
SQL> col object_name format A10
SQL> SELECT dbusername, action_name, object_name,
            system_privilege_used, unified_audit_policies
     FROM   unified_audit_trail
     WHERE  UNIFIED_AUDIT_POLICIES like '%AUD_SYS_POL%'
     AND    ACTION_NAME not in ('LOGON', 'LOGOFF');
  2    3    4    5
DBUSERNAME ACTION_NAME  OBJECT_NAM SYSTEM_PRIVILEGE_USED
---------- ------------ ---------- -----------------------
UNIFIED_AUDIT_POLICIES
------------------------------
SYS     SELECT  EMPLOYEES  SYSDBA
AUD_SYS_POL

SYS    SELECT EMPLOYEES   SYSDBA, SELECT ANY TABLE
AUD_SYS_POL

SYS     UPDATE  EMPLOYEES  SYSDBA
AUD_SYS_POL

SQL>
```

2. Drop the audit policy.

```
SQL> NOAUDIT POLICY aud_sys_pol BY sys;

Noaudit succeeded.

SQL> DROP AUDIT POLICY aud_sys_pol;

Audit Policy dropped.

SQL> EXIT
$
```

Practice 21-5: Auditing Data Pump Export *(Optional)*

Overview

In this practice, you create audit policies to audit Data Pump export operations in `pdb1_1` and import operations in `pdb1_2`. Then you will view the audited data after the exports and imports completed.

Assumptions

Practice 21-1 successfully enabled unified audit.

Tasks

1. In `pdb1_1`, create a `DP_PDB1_1_POL` for the component Data Pump, and more specifically for export operations.
   a. Connect as the security officer in `PDB1_1`. If the security officer, the `C##SEC` user does not exist in the pluggable databases, create the user. Use the following commands.

```
$ cd $HOME/labs/USERS
$ . oraenv
ORACLE_SID = [orcl] ? cdb1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to an idle instance.

SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2290024 bytes
Variable Size             268439192 bytes
Database Buffers          222298112 bytes
Redo Buffers                8032256 bytes
Database mounted.
Database opened.
SQL>
SQL> EXIT
$
$ ./create_sec_cdb.sh

SQL*Plus: Release 12.1.0.1.0 Production on Tue Jun 18 11:00:11
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.
```

```
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics,
Real Application Testing and Unified Auditing options

SQL> DROP USER c##sec CASCADE;
DROP USER c##sec CASCADE
          *
ERROR at line 1:
ORA-01918: user 'C##SEC' does not exist


SQL> CREATE USER c##sec IDENTIFIED BY oracle_4sec
  2  DEFAULT TABLESPACE USERS
  3  QUOTA UNLIMITED ON  USERS CONTAINER=ALL;

User created.

SQL>
SQL> GRANT create session
  2    TO c##sec
  3    WITH ADMIN OPTION CONTAINER=ALL;

Grant succeeded.

SQL>
SQL> GRANT select_catalog_role, select any table,
  2        create any context, drop any context,
  3        create user, alter user, drop user,
  4        create role, alter any role, drop any role,
  5        create table, create procedure,
  6        create any trigger, administer database trigger,
  7        create any directory, alter profile, create profile,
  8        drop profile, audit system, alter system,
  9        grant any object privilege, grant any privilege,
 10        grant any role
 11     TO c##sec
 12       CONTAINER=ALL;
```

```
Grant succeeded.

SQL>
SQL> GRANT execute on DBMS_SESSION to c##sec CONTAINER=ALL;

Grant succeeded.

SQL> GRANT execute on UTL_FILE to c##sec CONTAINER=ALL;

Grant succeeded.

SQL> GRANT audit_admin TO c##sec CONTAINER=ALL;

Grant succeeded.

SQL> EXIT
$
```

b. Create the `DP_PDB1_1_POL` audit policy in `pdb1_1`, to audit export operations for the component Data Pump .

```
$ sqlplus c##sec@pdb1_1

Enter password: ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing and Unified Auditing options

SQL> create audit policy DP_PDB1_1_POL actions
COMPONENT=datapump export;

Audit policy created.

SQL>
```

c. Enable the export audit policy.

```
SQL> audit policy DP_PDB1_1_POL;

Audit succeeded.

SQL>
```

d. Verify that the policy exists.

```
SQL> col user_name format A10
```

```
SQL> col policy_name format A20
SQL> SELECT * FROM   AUDIT_UNIFIED_ENABLED_POLICIES
     where POLICY_NAME like '%DP%';

USER_NAME  POLICY_NAME           ENABLED_ SUC FAI
---------- -------------------- -------- --- ---
ALL USERS  DP_PDB1_1_POL        BY       YES YES

SQL>
```

e. Create a directory for export operations.

```
SQL> CREATE DIRECTORY exp_dir AS
'/u01/app/oracle/admin/cdb1/dpdump';

Directory created.

SQL> GRANT read, write ON DIRECTORY exp_dir TO system;

Grant succeeded.

SQL> exit
$
```

f. Perform an export operation. Before exporting, ensure that the dump file does not exist; else, the export command will fail.

```
$ expdp system@pdb1_1 dumpfile=HR_tables tables=HR.EMPLOYEES
DIRECTORY=exp_dir REUSE_DUMPFILES=YES

Password: ******

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing
and Unified Auditing options
Starting "SYSTEM"."SYS_EXPORT_TABLE_01":
system/********@localhost:1521/pdb1_1 dumpfile=HR_tables
tables=HR.EMPLOYEES DIRECTORY=exp_dir REUSE_DUMPFILES=YES
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 64 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
Processing object type
TABLE_EXPORT/TABLE/POST_INSTANCE/PROCACT_INSTANCE
. . exported "HR"."EMPLOYEES"                            17.06
KB     107 rows
Master table "SYSTEM"."SYS_EXPORT_TABLE_01" successfully
loaded/unloaded
******************************************************************
**************
Dump file set for SYSTEM.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/admin/cdb1/dpdump/HR_tables.dmp
Job "SYSTEM"."SYS_EXPORT_TABLE_01" successfully completed at Sat
Jun 1 23:19:14 2013 elapsed 0 00:00:25


$
```

g. View the resulting audit data.

```
$ sqlplus c##sec@pdb1_1

Enter password : ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing and Unified Auditing options

SQL> set pages 100
SQL> select DBUSERNAME, DP_TEXT_PARAMETERS1,
            DP_BOOLEAN_PARAMETERS1, UNIFIED_AUDIT_POLICIES
     from   UNIFIED_AUDIT_TRAIL
     where  DP_TEXT_PARAMETERS1 is not null;

DBUSERNAME
------------------------------
DP_TEXT_PARAMETERS1
------------------------------------------------------------------
```

```
DP_BOOLEAN_PARAMETERS1
---------------------------------------------------------------
UNIFIED_AUDIT_POLICIES
---------------------------------------------------------------
SYSTEM
MASTER TABLE:  "SYSTEM"."SYS_EXPORT_TABLE_01" , JOB_TYPE:
EXPORT, METADATA_JOB_M
ODE: TABLE_EXPORT, JOB VERSION: 12.1.0.0.0, ACCESS METHOD:
AUTOMATIC, DATA OPTIO
NS: 0, DUMPER DIRECTORY: NULL  REMOTE LINK: NULL, TABLE EXISTS:
NULL, PARTITION
OPTIONS: NONE
MASTER_ONLY: FALSE, DATA_ONLY: FALSE, METADATA_ONLY: FALSE,
DUMPFILE_PRESENT: TR
UE, JOB_RESTARTED: FALSE



SQL> exit
$
```

2. In `pdb1_2`, create a `DP_PDB1_2_POL` for the component Data Pump, and more specifically for import operations.
   a. Connect as the security officer in `PDB1_2` to create the `DP_PDB1_2_POL` audit policy to audit import operations for the component Data Pump .

```
$ sqlplus c##sec@pdb1_2

Enter password : ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing
and Unified Auditing options

SQL> create audit policy DP_PDB1_2_POL actions
COMPONENT=datapump import;

Audit policy created.

SQL>
```

   b. Enable the export audit policy.

```
SQL> audit policy DP_PDB1_2_POL;

Audit succeeded.
```

```
SQL>
```

c. Verify that the policy exists.

```
SQL> col user_name format A10
SQL> col policy_name format A20
SQL> SELECT * FROM   AUDIT_UNIFIED_ENABLED_POLICIES
     where POLICY_NAME like '%DP%';
  2
USER_NAME  POLICY_NAME          ENABLED_ SUC FAI
---------- -------------------- -------- --- ---
ALL USERS  DP_PDB1_2_POL        BY       YES YES

SQL>
```

d. Create a directory for import operations.

```
SQL> CREATE DIRECTORY imp_dir AS
'/u01/app/oracle/admin/cdb1/dpdump';

Directory created.

SQL> GRANT read, write ON DIRECTORY imp_dir TO system;

Grant succeeded.

SQL> exit
$
```

e. Perform an import operation. Before importing, ensure that the dump file does not exist; else, the import command will fail.

```
$ impdp system@pdb1_2 dumpfile=HR_tables tables=HR.EMPLOYEES
DIRECTORY=imp_dir

Password: ******

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing
and Unified Auditing options
Master table "SYSTEM"."SYS_IMPORT_TABLE_01" successfully
loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_TABLE_01":
system/********@localhost:1521/pdb1_2 dumpfile=HR_tables
tables=HR.EMPLOYEES DIRECTORY=imp_dir
Processing object type TABLE_EXPORT/TABLE/TABLE
```

```
ORA-39151: Table "HR"."EMPLOYEES" exists. All dependent metadata
and data will be skipped due to table_exists_action of skip
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
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
Job "SYSTEM"."SYS_IMPORT_TABLE_01" completed with 1 error(s) at
Sat Jun 1 23:30:55 2013 elapsed 0 00:00:06


$
```

f. View the resulting audit data.

```
$ sqlplus c##sec@pdb1_2
Enter password : ******
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing and Unified Auditing options

SQL> set pages 100
SQL> select DBUSERNAME, DP_TEXT_PARAMETERS1,
            DP_BOOLEAN_PARAMETERS1, UNIFIED_AUDIT_POLICIES
     from   UNIFIED_AUDIT_TRAIL
     where  DP_TEXT_PARAMETERS1 is not null;
  2    3    4
DBUSERNAME
------------------------------
DP_TEXT_PARAMETERS1
----------------------------------------------------------------
DP_BOOLEAN_PARAMETERS1
----------------------------------------------------------------
UNIFIED_AUDIT_POLICIES
----------------------------------------------------------------
SYSTEM
```

```
MASTER TABLE:  "SYSTEM"."SYS_IMPORT_TABLE_01" , JOB_TYPE:
IMPORT, METADATA_JOB_M
ODE: TABLE_EXPORT, JOB VERSION: 12.1.0.0.0, ACCESS METHOD:
AUTOMATIC, DATA OPTIO
NS: 0, DUMPER DIRECTORY: NULL  REMOTE LINK: NULL, TABLE EXISTS:
SKIP, PARTITION
OPTIONS: NONE
MASTER_ONLY: FALSE, DATA_ONLY: FALSE, METADATA_ONLY: FALSE,
DUMPFILE_PRESENT: TR
UE, JOB_RESTARTED: FALSE



SQL>
```

3. View all export and import audited operations performed in `cdb1`.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> set pages 100
SQL> select DBUSERNAME, DP_TEXT_PARAMETERS1,
            DP_BOOLEAN_PARAMETERS1, UNIFIED_AUDIT_POLICIES
     from   UNIFIED_AUDIT_TRAIL
     where  DP_TEXT_PARAMETERS1 is not null;
  2    3    4

no rows selected

SQL>
```

Notice that there are no rows displayed. The `UNIFIED_AUDIT_TRAIL` displays the audited data for the current container, the `root` container in this case.

```
SQL> select DBUSERNAME, DP_TEXT_PARAMETERS1,
            DP_BOOLEAN_PARAMETERS1,
            UNIFIED_AUDIT_POLICIES, CON_ID
     from   CDB_UNIFIED_AUDIT_TRAIL
     where  DP_TEXT_PARAMETERS1 is not null;
  2    3    4    5
DBUSERNAME
------------------------------
DP_TEXT_PARAMETERS1
--------------------------------------------------------------------
DP_BOOLEAN_PARAMETERS1
--------------------------------------------------------------------
UNIFIED_AUDIT_POLICIES
--------------------------------------------------------------------
    CON_ID
```

```
----------
SYSTEM
MASTER TABLE:  "SYSTEM"."SYS_IMPORT_TABLE_01" , JOB_TYPE:
IMPORT, METADATA_JOB_M
ODE: TABLE_EXPORT, JOB VERSION: 12.1.0.0.0, ACCESS METHOD:
AUTOMATIC, DATA OPTIO
NS: 0, DUMPER DIRECTORY: NULL  REMOTE LINK: NULL, TABLE EXISTS:
SKIP, PARTITION
OPTIONS: NONE
MASTER_ONLY: FALSE, DATA_ONLY: FALSE, METADATA_ONLY: FALSE,
DUMPFILE_PRESENT: TR
UE, JOB_RESTARTED: FALSE

      4


SYSTEM
MASTER TABLE:  "SYSTEM"."SYS_EXPORT_TABLE_01" , JOB_TYPE:
EXPORT, METADATA_JOB_M
ODE: TABLE_EXPORT, JOB VERSION: 12.1.0.0.0, ACCESS METHOD:
AUTOMATIC, DATA OPTIO
NS: 0, DUMPER DIRECTORY: NULL  REMOTE LINK: NULL, TABLE EXISTS:
NULL, PARTITION
OPTIONS: NONE
MASTER_ONLY: FALSE, DATA_ONLY: FALSE, METADATA_ONLY: FALSE,
DUMPFILE_PRESENT: TR
UE, JOB_RESTARTED: FALSE

      3



SQL> EXIT
$
```

Practice 21-6: Auditing RMAN Backups

In this practice, you perform RMAN backups for the `orcl` database. Then you will view the audited data after RMAN backups are completed. You do not have to create any audit policy for RMAN operations. RMAN is by default audited.

Assumptions

Practice 21-1 successfully enabled unified audit.

Tasks

1. Perform a RMAN backup of the `USERS` tablespace.
   a. If the keystore is not opened, you will encounter the following errors `ORA-19914: unable to encrypt backup` and `ORA-28365: wallet is not open`. In this case, first open the keystore.

```
$ . oraenv
ORACLE_SID = [cdb1] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as syskm

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                               IDENTIFIED BY secret;
  2
keystore altered.

SQL> exit
$
```

   b. Perform the backup.

```
$ rman target /

connected to target database: ORCL (DBID=1315477536)

RMAN> backup tablespace USERS;

Starting backup at 02-JUN-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
```

```
channel ORA_DISK_1: SID=366 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00006
name=/u01/app/oracle/oradata/orcl/users01.dbf
channel ORA_DISK_1: starting piece 1 at 02-JUN-13
channel ORA_DISK_1: finished piece 1 at 02-JUN-13
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/backupset/2013_06
_02/o1_mf_nnndf_TAG20130602T083703_8tp11jsx_.bkp
tag=TAG20130602T083703 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:04
Finished backup at 02-JUN-13
RMAN> exit

Recovery Manager complete.
$
```

2. Perform a restore and recover after removing the `USERS` tablespace file.
   a. Find the data file name of the `USERS` tablespace and remove the file.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing and Unified Auditing options

SQL> select name from v$datafile;

NAME
-----------------------------------------------------------------
/u01/app/oracle/oradata/orcl/system01.dbf
/u01/app/oracle/oradata/orcl/example01.dbf
/u01/app/oracle/oradata/orcl/sysaux01.dbf
/u01/app/oracle/oradata/orcl/undotbs01.dbf
/u01/app/oracle/oradata/orcl/enctbs01.dbf
/u01/app/oracle/oradata/orcl/users01.dbf

6 rows selected.

SQL> !rm /u01/app/oracle/oradata/orcl/users01.dbf

SQL>
```

b. Put the tablespace OFFLINE.

```
SQL> alter tablespace users offline immediate;

Tablespace altered.

SQL> exit
$
```

c. Restore and recover the data file.

```
$ rman target /

connected to target database: ORCL (DBID=1315477536)

RMAN> restore tablespace USERS;

Starting restore at 02-JUN-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=130 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00006 to
/u01/app/oracle/oradata/orcl/users01.dbf
channel ORA_DISK_1: reading from backup piece
/u01/app/oracle/fast_recovery_area/ORCL/backupset/2013_06_02/o1_
mf_nnndf_TAG20130602T083703_8tp11jsx_.bkp
channel ORA_DISK_1: piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/backupset/2013_06
_02/o1_mf_nnndf_TAG20130602T083703_8tp11jsx_.bkp
tag=TAG20130602T083703
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
Finished restore at 02-JUN-13

RMAN> recover tablespace USERS;

Starting recover at 02-JUN-13
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:01
```

```
Finished recover at 02-JUN-13

RMAN> exit
$
```

d. Put the tablespace `USERS` back online.

```
$ sqlplus system
Enter password: ******

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics, Real Application Testing and Unified Auditing options

SQL> alter tablespace USERS online;

Tablespace altered.

SQL>
```

3. View the resulting audit data.

```
SQL> select DBUSERNAME, RMAN_OPERATION
     from   UNIFIED_AUDIT_TRAIL
     where  RMAN_OPERATION is not null;
   2    3
DBUSERNAME                     RMAN_OPERATION
------------------------------ --------------------
SYS                    Recover
SYS                    Restore
SYS                    Backup

SQL>
```

Practice 21-7: Auditing Database Vault Violations *(Optional)*

Overview

In this practice, you will create an audit policy auditing actions using a Database Vault realm that protects the `HR.EMPLOYEES` and `HR.DEPARTMENTS` tables from system privileged users.

Assumptions

Database Vault has been successfully configured in practice 3-7.

Tasks

1. Reenable Database Vault and create the Database Vault "`HR Application`" realm, to protect the `HR.EMPLOYEES` and `HR.DEPARTMENTS` tables from system privileged users.
   a. Use the `$HOME/labs/AUDIT/audit_dv.sql` script.

```
SQL> @$HOME/labs/AUDIT/audit_dv.sql
SQL> connect sec_admin/oracle_4U
Connected.
SQL> exec DVSYS.DBMS_MACADM.ENABLE_DV

PL/SQL procedure successfully completed.

SQL> set echo on
SQL> EXEC  DVSYS.DBMS_MACADM.DELETE_REALM(realm_name  => 'HR
Application')
BEGIN DVSYS.DBMS_MACADM.DELETE_REALM(realm_name    => 'HR
Application'); END;

*
ERROR at line 1:
ORA-47241: Realm HR Application not found
ORA-06512: at "DVSYS.DBMS_MACADM", line 1847
ORA-06512: at line 1

SQL> BEGIN
  2   DVSYS.DBMS_MACADM.CREATE_REALM(
  3    realm_name    => 'HR Application',
  4    description   => 'Realm to protect the HR application',
  5    enabled       => DBMS_MACUTL.G_YES,
  6    audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL +
DBMS_MACUTL.G_REALM_AUDIT_SUCCESS,
  7    realm_type    => 1);
  8  END;
  9  /
```

```
PL/SQL procedure successfully completed.

SQL>
SQL> BEGIN
  2   DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM(
  3    realm_name   => 'HR Application',
  4    object_owner => 'HR',
  5    object_name  => 'EMPLOYEES',
  6    object_type  => 'TABLE');
  7  END;
  8  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2   DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM(
  3    realm_name   => 'HR Application',
  4    object_owner => 'HR',
  5    object_name  => 'DEPARTMENTS',
  6    object_type  => 'TABLE');
  7  END;
  8  /

PL/SQL procedure successfully completed.

SQL>
```

b. Restart the database instance.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289400 bytes
Variable Size             297795848 bytes
Database Buffers          192937984 bytes
Redo Buffers                8036352 bytes
Database mounted.
```

```
Database opened.
SQL>
```

2. Create and enable an audit policy that audits any violation against the Database Vault "`HR Application`" realm created by the `$HOME/labs/AUDIT/audit_dv.sql` script.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> CREATE AUDIT POLICY aud_DV_pol ACTIONS
            COMPONENT = dv realm violation on "HR Application";
  2
Audit policy created.

SQL> AUDIT POLICY aud_DV_pol;

Audit succeeded.

SQL>
```

3. Connect as a privileged user attempting to update rows in the `HR.EMPLOYEES` table.

```
SQL> CONNECT system
Enter password: ******
Connected.
SQL> UPDATE hr.employees SET salary=salary+100;
UPDATE hr.employees SET salary=salary+100
          *
ERROR at line 1:
ORA-01031: insufficient privileges
SQL>
```

4. View the audited data related to the realm violation.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> COL dbusername FORMAT A8
SQL> COL DV_ACTION_NAME FORMAT A22
SQL> COL DV_ACTION_OBJECT_NAME FORMAT A16
SQL> select DBUSERNAME, DV_ACTION_NAME, DV_RETURN_CODE,
            DV_ACTION_OBJECT_NAME
     from   UNIFIED_AUDIT_TRAIL
     where  DV_ACTION_NAME is not null;
DBUSERNA DV_ACTION_NAME        DV_RETURN_CODE DV_ACTION_OBJECT
-------- ---------------------- -------------- ----------------
… rows deleted
```

```
SYSTEM   Realm Violation Audit            1031 HR Application

… rows deleted

SQL>
```

5. Disable and drop the audit policy.

```
SQL> NOAUDIT POLICY aud_DV_pol;

Noaudit succeeded.

SQL> DROP AUDIT POLICY aud_DV_pol;

Audit Policy dropped.

SQL> EXIT
$
```

6. Run the `DV_drop_realm.sh` script to remove the Database Vault protection on the `HR.EMPLOYEES` and `HR.DEPARTMENTS` tables.

```
$ $HOME/labs/DV/DV_drop_realm.sh

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics,
Oracle Database Vault and Real Application Testing options

PL/SQL procedure successfully completed.
$
```

7. Run the `DV_disable.sh` script to disable Database Vault in the database.

```
$ $HOME/labs/DV/DV_disable.sh
SQL*Plus: Release 12.1.0.1.0 Production on Sun Jun 2 12:09:07
2013

Copyright (c) 1982, 2013, Oracle.  All rights reserved.



Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics,
```

```
Oracle Database Vault, Real Application Testing and Unified
Auditing options

Connected.

PL/SQL procedure successfully completed.

Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics,
Oracle Database Vault, Real Application Testing and Unified
Auditing options
$
```

8. Restart the database instance.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options

SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2289400 bytes
Variable Size             264241416 bytes
Database Buffers          226492416 bytes
Redo Buffers                8036352 bytes
Database mounted.
Database opened.
SQL> EXIT
$
```

Practices for Lesson 22: Using Fine-Grained Audit

Chapter 22

Practice 22-1: Implementing Fine-Grained Auditing

Overview

There is a business requirement that a record must be logged whenever employee salary information is accessed. The execution of `INSERT`, `UPDATE`, and `DELETE` commands is recorded in a journal table by the use of triggers. Create a proof of concept solution for `SELECT` accesses. Create a user `PFAY`, and prove that `SELECT` accesses will be recorded. Execute a practice script to create a procedure called `SEC.LOG_EMPS_SALARY`. This procedure inserts a record in the `SEC.TEST_AUDIT_PROC` table to demonstrate that additional audit information can be captured and stored.

Assumptions: This solution depends on step 1 of Practice 4-1. The `SEC` user must exist and the password of the `SEC` user is `oracle_4sec` and the password of the `HR` user is `oracle_4U` by these previous practices.

Task

1. As the `SEC` user, create the `PFAY` user and grant `SELECT` access to the `HR.EMPLOYEES` table to `PFAY`.

   Create the `PFAY` user with the password `oracle_4U`.
   Grant `PFAY` the required access.

   Because `SEC` has been granted `GRANT ANY OBJECT PRIVILEGE`, the `SEC` user may grant `SELECT` on `HR.EMPLOYEES`.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus sec
Enter password: ******


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced
Analytics and Real Application Testing options


SQL> DROP USER pfay;


User dropped.


SQL> GRANT create session TO pfay IDENTIFIED BY oracle_4U;


Grant succeeded.


SQL> GRANT select ON hr.employees TO pfay;
```

```
Grant succeeded.

SQL>
```

2. Ensure that EXAMPLE is the default tablespace for the SEC user and that SEC has the ability to create objects in the EXAMPLE tablespace. Because SEC has been granted ALTER USER, the SEC user may alter his or her own account settings.

   Note: Every user may change his or her password.

```
SQL> ALTER USER sec
      DEFAULT TABLESPACE example
      QUOTA UNLIMITED ON example;
  2    3
User altered.

SQL>
```

3. Enable the SEC user to execute the DBMS_FGA package. The SEC user cannot grant privileges on objects owned by SYS.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> GRANT execute ON dbms_fga TO sec;

Grant succeeded.

SQL>
```

4. As the SEC user, create an FGA policy with the following properties:
   Object: HR.EMPLOYEES
   Name: AUDIT_EMPS_SALARY
   Audits: Any access to the SALARY column
   Policy: Enabled

```
SQL> connect sec
Enter password: ******
Connected.
SQL>
SQL> BEGIN
     dbms_fga.add_policy (
       object_schema         => 'hr',
       object_name           => 'employees',
       policy_name           => 'audit_emps_salary',
       audit_condition       => NULL,
       audit_column          => 'salary',
       enable                => TRUE );
     END;
```

```
   /
  2    3    4    5    6    7    8    9   10
PL/SQL procedure successfully completed.

SQL>
```

Practice 22-2: Viewing the FGA Trail

Overview

In this practice, you view the fine-grained audit results.

Tasks

1. As the `PFAY` user, select `SALARY` from the `HR.EMPLOYEES` table. Save this statement as `list.sql` because you will execute it again.

```
SQL> CONNECT pfay
Enter password: ******
Connected.
SQL>
SQL> SELECT salary FROM hr.employees;

    SALARY
----------
     24000
     17000
     17000
      9000
      6000
      4800
      4800
…   Rows deleted  …
      6500
     10000
     12000
      8300

83 rows selected.

SQL> save /home/oracle/labs/list.sql replace
Wrote file /home/oracle/labs/list.sql
SQL>
```

2. As the `SEC` user, display the audit record from the previous `SELECT` statement. Use `$HOME/labs/FGA/view.sql`.

   Note: The time stamp that is shown is the time when step 1 was executed.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> @$HOME/labs/FGA/view.sql
```

```
SQL> COL timestamp        FORMAT A10
```

```
SQL> COL db_user           FORMAT A7
SQL> COL object_schema         FORMAT A15
SQL> COL object_name      FORMAT A12
SQL> COL policy_name      FORMAT A20
SQL> COL sql_bind         FORMAT A10
SQL> COL sql_text         FORMAT A32
SQL> COL dbusername   FORMAT A10
SQL> COL fga_policy_name FORMAT A18
SQL>
SQL> SET PAGESIZE 40
SQL> SET LINESIZE 56
SQL>
SQL> SELECT    to_char(timestamp, 'YYMMDDHH24MI') AS timestamp,
  2             db_user,
  3             object_schema,
  4             object_name,
  5             policy_name,
  6             sql_bind,
  7             sql_text
  8  FROM dba_fga_audit_trail;

no rows selected

SQL>
SQL> SELECT    to_char(event_timestamp, 'YYMMDDHH24MI')  AS
timestamp,
  2            dbusername,
  3            fga_policy_name,
  4            sql_text
  5  FROM      unified_audit_trail
  6  WHERE      fga_policy_name = 'AUDIT_EMPS_SALARY';


TIMESTAMP  DBUSERNAME FGA_POLICY_NAME
---------- ---------- ------------------
SQL_TEXT
--------------------------------
1306030211 PFAY       AUDIT_EMPS_SALARY
SELECT salary FROM hr.employees
```

```
SQL>
```

Practice 22-3: Using an Event Handler

Overview

In this practice, you will create an FGA policy with an event handler triggered to capture additional information.

Tasks

1. Review and then execute `$HOME/labs/FGA/test_audit_proc.sql`, which creates a table to store audit records and creates a procedure to store audit events in that table. The script creates the `TEST_AUDIT_PROC` table and the `LOG_EMPS_SALARY` procedure. The procedure captures additional information and inserts it into the table. It is important to capture enough information in the table to be able to relate this record back to a single FGA audit record.

```
SQL> @$HOME/labs/FGA/test_audit_proc.sql
SQL> SET ECHO OFF
SQL>
SQL> CONNECT sec/oracle_4sec
Connected.
SQL>
SQL> DROP TABLE sec.test_audit_proc;

DROP TABLE sec.test_audit_proc
               *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
SQL> CREATE TABLE sec.test_audit_proc (
  2    object_schema VARCHAR2(80),
  3    object_name      VARCHAR2(80),
  4    policy_name      VARCHAR2(80),
  5    session_id       NUMBER,
  6    timestamp        DATE,
  7    audit_entry_id NUMBER );

Table created.

SQL>
SQL> DROP PROCEDURE sec.log_emps_salary;

DROP PROCEDURE sec.log_emps_salary
*
ERROR at line 1:
ORA-04043: object LOG_EMPS_SALARY does not exist
```

```
SQL>
SQL> CREATE PROCEDURE sec.log_emps_salary (
  2    p_object_schema VARCHAR2,
  3    p_object_name  VARCHAR2,
  4    p_policy_name  VARCHAR2 )
  5  AS
  6  BEGIN
  7    INSERT
  8      INTO sec.test_audit_proc
  9        (object_schema, object_name, policy_name, session_id,
 10          timestamp)
 11      VALUES (p_object_schema,
 12           p_object_name,
 13           p_policy_name,
 14           SYS_CONTEXT('userenv', 'SESSIONID'),
 15           systimestamp);
 16  END;
 17  /

Procedure created.

SQL>
```

2. Drop the FGA policy and re-create it so that it calls the procedure created in the previous step.

```
SQL> BEGIN
       dbms_fga.drop_policy (
         object_schema => 'hr',
         object_name   => 'employees',
         policy_name   => 'audit_emps_salary' );

       dbms_fga.add_policy (
         object_schema  => 'hr',
         object_name  => 'employees',
         policy_name  => 'audit_emps_salary',
         audit_condition => NULL,
         audit_column  => 'salary',
         handler_schema => 'sec',
         handler_module => 'log_emps_salary',
         enable  => TRUE );
     END;
   /
```

```
  2    3    4    5    6    7    8    9   10   11   12   13   14
15   16   17
PL/SQL procedure successfully completed.

SQL>
```

3. As the PFAY user, select SALARY from the HR.EMPLOYEES table.

```
SQL> CONNECT pfay
Enter password: ******
Connected.
SQL> SELECT salary FROM hr.employees;

    SALARY
----------
     24000
     17000
     17000
      9000
… Rows deleted …
      2600
      4400
     13000
      6000
      6500
     10000
     12000
      8300

83 rows selected.

SQL>
```

4. As the SEC user, display the audit record from the previous SELECT statement. Use the same script that you used in practice 22-2 step 2, $HOME/labs/FGA/view.sql.

```
SQL> CONNECT sec
Enter password: ******
Connected.
SQL> @$HOME/labs/FGA/view.sql
SQL>
SQL> COL timestamp        FORMAT A10
SQL> COL db_user            FORMAT A7
SQL> COL object_schema          FORMAT A15
SQL> COL object_name      FORMAT A12
SQL> COL policy_name      FORMAT A20
```

```
SQL> COL sql_bind         FORMAT A10
SQL> COL sql_text         FORMAT A32
SQL> COL dbusername   FORMAT A10
SQL> COL fga_policy_name FORMAT A18
SQL>
SQL> SET PAGESIZE 40
SQL> SET LINESIZE 56
SQL>
SQL> SELECT    to_char(timestamp, 'YYMMDDHH24MI') AS timestamp,
  2             db_user,
  3             object_schema,
  4             object_name,
  5             policy_name,
  6             sql_bind,
  7             sql_text
  8  FROM dba_fga_audit_trail;

no rows selected

SQL>
SQL> SELECT    to_char(event_timestamp, 'YYMMDDHH24MI')  AS
timestamp,
  2            dbusername,
  3            fga_policy_name,
  4            sql_text
  5  FROM      unified_audit_trail
  6  WHERE      fga_policy_name = 'AUDIT_EMPS_SALARY';

TIMESTAMP  DBUSERNAME FGA_POLICY_NAME
---------- ---------- ------------------
SQL_TEXT
--------------------------------
1306030225 PFAY       AUDIT_EMPS_SALARY
SELECT salary FROM hr.employees

1306030211 PFAY       AUDIT_EMPS_SALARY
SELECT salary FROM hr.employees

SQL>
```

5. Verify that the audit handler created a row in the `TEST_AUDIT_PROC` table.

```
SQL> SELECT  object_schema,
             object_name,
```

```
             policy_name
     FROM    test_audit_proc;

OBJECT_SCHEM OBJECT_NAM POLICY_NAME
------------ ---------- --------------------
HR           EMPLOYEES  AUDIT_EMPS_SALARY

SQL>
```

6. As SEC, display the audit policy information from the data dictionary.

```
SQL> COLUMN pf_schema FORMAT A10
SQL> COLUMN pf_package FORMAT A12
SQL> COLUMN pf_function FORMAT A20
SQL>
SQL> SELECT * FROM dba_audit_policies;

OBJECT_SCHEMA  OBJECT_NAM POLICY_OWNER
-------------- ---------- ------------------------------
POLICY_NAME          POLICY_TEXT          POLICY_COLUMN
-------------------- -------------------- --------------
PF_SCHEMA  PF_PACKAGE   PF_FUNCTION          ENA SEL INS
---------- ------------ -------------------- --- --- ---
UPD DEL AUDIT_TRAIL  POLICY_COLU
--- --- ------------ -----------
HR             EMPLOYEES  SEC
AUDIT_EMPS_SALARY                           SALARY
SEC                     LOG_EMPS_SALARY      YES YES NO
NO  NO  DB+EXTENDED  ANY_COLUMNS


SQL>
```

7. Drop the FGA policy.

```
SQL> EXEC dbms_fga.drop_policy (-
            object_schema => 'hr', -
            object_name   => 'employees', -
            policy_name   => 'audit_emps_salary' )
> > >
PL/SQL procedure successfully completed.

SQL> exit
$
```

Appendix D
Source Code

Source for PROXY_USER

```
#include <oci.h>
#include <stdio.h>

#define MAXTHREAD 10

static OCIError   *errhp;
static OCIEnv     *envhp;
static OCICPool   *poolhp;

static int employeeNum[MAXTHREAD];

static OraText *poolName;
static sb4 poolNameLen;
static text *database;
static text *username;
static text *password;
static text *nullpassword =(text *)"";
static text *appusername =(text *)"HRAPP";
static text *apppassword =(text *)"HRAPP";

static ub4 conMin = 2;
static ub4 conMax = 5;
static ub4 conIncr = 1;

static void checkerr (OCIError *errhp, sword status);
static void threadFunction (dvoid *arg);

int main (ac, av)
  unsigned ac;
  char *av[];
{
  unsigned ai;
  int i = 0;
  database = av[1];
  username = av[2];

  if (ac>3)
    password = av[3];
  else password = nullpassword;

  printf("Database: %s\nUsername: %s\nPassword: %s\n", database,
          username, password);

  OCIEnvCreate (&envhp, OCI_THREADED, (dvoid *)0,
               (dvoid *(*)()) 0,(dvoid * (*)()) 0,
               (dvoid (*)()) 0, 0, (dvoid *)0);
...
```

Source for PROXY_USER (continued)

```
...
   (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp,
                OCI_HTYPE_ERROR,(size_t) 0, (dvoid **) 0);


  (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **) &poolhp,
                OCI_HTYPE_CPOOL,(size_t) 0, (dvoid **) 0);

  /* CREATE THE CONNECTION POOL */
  checkerr (errhp, OCIConnectionPoolCreate(envhp,
                   errhp,poolhp, &poolName, &poolNameLen,
                   database,strlen(database),
                   conMin, conMax, conIncr,
                   appusername,strlen(appusername),
                   apppassword,
                   strlen(apppassword),OCI_DEFAULT));

  printf("Successful connection: Username: %s\n", appusername);
/* Multiple threads using the connection pool */
  {
    OCIThreadId     *thrid[MAXTHREAD];
    OCIThreadHandle *thrhp[MAXTHREAD];

    OCIThreadProcessInit ();
    checkerr (errhp, OCIThreadInit (envhp, errhp));
    for (i = 0; i < MAXTHREAD; ++i)
    {
     checkerr (errhp, OCIThreadIdInit(envhp, errhp,
               &thrid[i]));
     checkerr (errhp, OCIThreadHndInit(envhp, errhp,
               &thrhp[i]));
    }
    for (i = 0; i < MAXTHREAD; ++i)
    {
      employeeNum[i]=i;
      checkerr (errhp, OCIThreadCreate (envhp, errhp,
          threadFunction, (dvoid *) &employeeNum[i], thrid[i],
          thrhp[i]));
    }
    for (i = 0; i < MAXTHREAD; ++i)
...
```

Source for PROXY_USER (continued)

```
...
    {
      checkerr (errhp, OCIThreadJoin (envhp, errhp, thrhp[i]));
      checkerr (errhp, OCIThreadClose (envhp, errhp, thrhp[i]));
      checkerr (errhp, OCIThreadIdDestroy (envhp, errhp,
               &(thrid[i])));
      checkerr (errhp, OCIThreadHndDestroy (envhp, errhp,
               &(thrhp[i])));
    }
    checkerr (errhp, OCIThreadTerm (envhp, errhp));
  } /* ALL THE THREADS ARE COMPLETE */
checkerr(errhp, OCIConnectionPoolDestroy(poolhp, errhp,OCI_DEFAULT));
checkerr(errhp, OCIHandleFree((dvoid *)poolhp,
          OCI_HTYPE_CPOOL));
checkerr(errhp, OCIHandleFree((dvoid *)errhp,
          OCI_HTYPE_ERROR));
}
/* end of main () */

static void threadFunction (dvoid *arg)
{
  int i, c;
  int empno = *(int *)arg;
  OCISvcCtx *svchp = (OCISvcCtx *) arg;
  text insertst1[256];
  OCIStmt *stmthp = (OCIStmt *)0;

  sword status;

  status = OCILogon2(envhp, errhp, &svchp,
             (CONST OraText *)username, strlen(username),
             (CONST OraText *)password, strlen(password),
             (CONST OraText *)poolName, poolNameLen, OCI_CPOOL);

  if (status == 0)
    printf("Successful connection: Username: %s\n", username);
  else checkerr(errhp,status);

...
```

Source for PROXY_USER (continued)

```
...
  for (i=0; i < 1000; ++i) i=i;
/*
  printf("Press any key to continue:");
  c = getchar();

  checkerr(errhp,OCILogon2(envhp, errhp, &svchp,
            (CONST OraText *)username, strlen(username),
            (CONST OraText *)password, strlen(password),
            (CONST OraText *)poolName, poolNameLen, OCI_CPOOL));

  sprintf(insertst1,"SET ROLE hr_emp_clerk");

  OCIHandleAlloc(envhp, (dvoid **)&stmthp, OCI_HTYPE_STMT,
                 (size_t)0,(dvoid **)0);

  checkerr(errhp, OCIStmtPrepare (stmthp, errhp,
           (text *)insertst1,(ub4)strlen(insertst1),
            OCI_NTV_SYNTAX, OCI_DEFAULT));

  checkerr(errhp, OCIStmtExecute (svchp, stmthp, errhp, (ub4)1,
           (ub4)0,(OCISnapshot *)0, (OCISnapshot *)0,
            OCI_DEFAULT ));

  checkerr(errhp, OCITransCommit(svchp,errhp,(ub4)0));

  checkerr(errhp, OCIHandleFree((dvoid *) stmthp,
            OCI_HTYPE_STMT));
*/
  checkerr(errhp, OCILogoff((dvoid *) svchp, errhp));
} /* end of threadFunction (dvoid *) */

void checkerr(errhp, status)
OCIError *errhp;
sword status;
{
  text errbuf[512];
  sb4 errcode = 0;

  switch (status)
  {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    (void) printf("Error - OCI_SUCCESS_WITH_INFO\n");
    break;

...
```

Source for PROXY_USER (continued)

```
...
  case OCI_NEED_DATA:
    (void) printf("Error - OCI_NEED_DATA\n");
    break;
  case OCI_NO_DATA:
    (void) printf("Error - OCI_NODATA\n");
    break;
  case OCI_ERROR:
    (void) OCIErrorGet((dvoid *)errhp, (ub4) 1, (text *) NULL,
          &errcode, errbuf,(ub4) sizeof(errbuf),
          OCI_HTYPE_ERROR);
    (void) printf("Error - %.*s\n", 512, errbuf);
    break;
  case OCI_INVALID_HANDLE:
    (void) printf("Error - OCI_INVALID_HANDLE\n");
    break;

case OCI_STILL_EXECUTING:
    (void) printf("Error - OCI_STILL_EXECUTE\n");
    break;
  case OCI_CONTINUE:
    (void) printf("Error - OCI_CONTINUE\n");
    break;
  default:
    break;
  }
}
```

Source for PROXY_ROLE

```
#include <oci.h>
#include <stdio.h>

#define MAXTHREAD 10

static OCIError   *errhp;
static OCIEnv     *envhp;
static OCICPool   *poolhp;

static int employeeNum[MAXTHREAD];

static OraText *poolName;
static sb4 poolNameLen;
static text *database;
static text *username;
static text *password;
static text *role;
static text *nullpassword =(text *)"";
static text *appusername =(text *)"HRAPP";
static text *apppassword =(text *)"HRAPP";

static ub4 conMin = 2;
static ub4 conMax = 5;
static ub4 conIncr = 1;

static void checkerr (OCIError *errhp, sword status);
static void threadFunction (dvoid *arg);

int main (ac, av)
  unsigned ac;
  char *av[];
{
  unsigned ai;

  int i = 0;
  database = av[1];
  role = av[2];
  username = av[3];

  if (ac>4)
    password = av[4];
  else password = nullpassword;

printf("Database: %s\nRole:     %s\nUsername: %s\nPassword:
%s\n", database, role, username, password);

...
```

Source for PROXY_ROLE (continued)

```
...
  OCIEnvCreate (&envhp, OCI_THREADED, (dvoid *)0,
                (dvoid * (*)()) 0, (dvoid * (*)()) 0,
                (dvoid (*)()) 0, 0, (dvoid *)0);

  (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp,
              OCI_HTYPE_ERROR,(size_t) 0, (dvoid **) 0);


  (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **) &poolhp,
              OCI_HTYPE_CPOOL, (size_t) 0, (dvoid **) 0);

  /* CREATE THE CONNECTION POOL */
  checkerr (errhp, OCIConnectionPoolCreate(envhp,
                 errhp,poolhp, &poolName, &poolNameLen,
                 database,strlen(database),
                 conMin, conMax, conIncr,
                 appusername,strlen(appusername),
                 apppassword,strlen(apppassword),OCI_DEFAULT));

  printf("Successful connection: Username: %s\n", appusername);

  /* Multiple threads using the connection pool */
  {
    OCIThreadId     *thrid[MAXTHREAD];
    OCIThreadHandle *thrhp[MAXTHREAD];

    OCIThreadProcessInit ();
    checkerr (errhp, OCIThreadInit (envhp, errhp));
    for (i = 0; i < MAXTHREAD; ++i)
    {
      checkerr (errhp, OCIThreadIdInit (envhp, errhp,
              &thrid[i]));
      checkerr (errhp, OCIThreadHndInit (envhp, errhp,
              &thrhp[i]));
    }
    for (i = 0; i < MAXTHREAD; ++i)
    {
      checkerr (errhp, OCIThreadJoin (envhp, errhp, thrhp[i]));
      checkerr (errhp, OCIThreadClose (envhp, errhp, thrhp[i]));
      checkerr (errhp, OCIThreadIdDestroy (envhp, errhp,
              &(thrid[i])));
      checkerr (errhp, OCIThreadHndDestroy (envhp, errhp,
              &(thrhp[i])));
    }
    checkerr (errhp, OCIThreadTerm (envhp, errhp));
  } /* ALL THE THREADS ARE COMPLETE */
...
```

Source for PROXY_ROLE (continued)

```
...
   checkerr(errhp, OCIConnectionPoolDestroy(poolhp, errhp,
                OCI_DEFAULT));
  checkerr(errhp, OCIHandleFree((dvoid *)poolhp,
                OCI_HTYPE_CPOOL));
  checkerr(errhp, OCIHandleFree((dvoid *)errhp,
                OCI_HTYPE_ERROR));
} /* end of main () */

static void threadFunction (dvoid *arg)
{
  int i, c;
  int empno = *(int *)arg;
  OCISvcCtx *svchp = (OCISvcCtx *) arg;
  text insertst1[256];
  OCIStmt *stmthp = (OCIStmt *)0;

  sword status;
  status = OCILogon2(envhp, errhp, &svchp,
              (CONST OraText *)username, strlen(username),
              (CONST OraText *)password, strlen(password),
              (CONST OraText *)poolName, poolNameLen,OCI_CPOOL);

  if (status == 0)
    printf("Successful connection: Username: %s\n", username);
  else checkerr(errhp,status);

  sprintf(insertst1,"SET ROLE %s", role);

  OCIHandleAlloc(envhp, (dvoid **)&stmthp, OCI_HTYPE_STMT,
                    (size_t)0, (dvoid **)0);

  checkerr(errhp, OCIStmtPrepare (stmthp, errhp,
          (text *)insertst1, (ub4)strlen(insertst1),
           OCI_NTV_SYNTAX, OCI_DEFAULT));

  status = OCIStmtExecute (svchp, stmthp, errhp, (ub4)1, (ub4)0,
           (OCISnapshot *)0, (OCISnapshot *)0, OCI_DEFAULT );

  if (status == 0)
    printf("Role successfully enabled: %s\n", role);
  else checkerr(errhp,status);

  checkerr(errhp, OCITransCommit(svchp,errhp,(ub4)0));

  checkerr(errhp, OCIHandleFree((dvoid *) stmthp,
                OCI_HTYPE_STMT));
...
```

Source for PROXY_ROLE (continued)

```
...
  checkerr(errhp, OCILogoff((dvoid *) svchp, errhp));
} /* end of threadFunction (dvoid *) */

void checkerr(errhp, status)
OCIError *errhp;
sword status;
{
  text errbuf[512];
  sb4 errcode = 0;

  switch (status)
  {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    (void) printf("Error - OCI_SUCCESS_WITH_INFO\n");
    break;
  case OCI_NEED_DATA:
    (void) printf("Error - OCI_NEED_DATA\n");
    break;
  case OCI_NO_DATA:
    (void) printf("Error - OCI_NODATA\n");
    break;
  case OCI_ERROR:
    (void) OCIErrorGet((dvoid *)errhp, (ub4) 1, (text *) NULL,
               &errcode,errbuf, (ub4) sizeof(errbuf),
               OCI_HTYPE_ERROR);
    (void) printf("Error - %.*s\n", 512, errbuf);
    break;
  case OCI_INVALID_HANDLE:
    (void) printf("Error - OCI_INVALID_HANDLE\n");
    break;
  case OCI_STILL_EXECUTING:
    (void) printf("Error - OCI_STILL_EXECUTE\n");
    break;
  case OCI_CONTINUE:
    (void) printf("Error - OCI_CONTINUE\n");
    break;
  default:
    break;
  }
}
```

Appendix E
USERENV and SYS_SESSION_ROLES Contexts

Predefined Parameters of the USERENV Namespace (continued)

Predefined Parameters of the USERENV Namespace

Parameter	Return Value
ACTION	Identifies the position in the module (application name) and is set through the DBMS_APPLICATION_INFO package or OCI
AUDITED_CURSORID	Returns the cursor ID of the SQL that triggered the audit. This parameter is not valid in a fine-grained auditing environment. If you specify it in such an environment, the Oracle database always returns NULL.
AUTHENTICATED_IDENTITY	Returns the identity used in authentication. In the list that follows, the type of user is followed by the value returned: • Kerberos-authenticated enterprise user: Kerberos principal name • Kerberos-authenticated external user: Kerberos principal name; same as the schema name • SSL-authenticated enterprise user: The DN in the user's PKI certificate • SSL-authenticated external user: The DN in the user's PKI certificate • Password-authenticated enterprise user: Nickname; same as the login name • Password-authenticated database user: The database username; same as the schema name • OS-authenticated external user: The external operating system username • RADIUS/DCE-authenticated external user: The schema proxy with DN: Oracle Internet Directory DN of the client • Proxy with certificate: Certificate DN of the client • Proxy with username: Database username if client is a local database user; nickname if client is an enterprise user • SYSDBA/SYSOPER using Password File: Login name • SYSDBA/SYSOPER using OS authentication: Operating system username
AUTHENTICATION_DATA	Is the data being used to authenticate the login user. For X.503 certificate–authenticated sessions, this field returns the context of the certificate in HEX2 format. **Note:** You can change the return value of the AUTHENTICATION_DATA attribute by using the length parameter of the syntax. Values up to 4,000 are accepted. This is the only attribute of USERENV for which the Oracle

Predefined Parameters of the USERENV Namespace (continued)

	database implements such a change.
AUTHENTICATION_METHOD	Returns the method of authentication. In the list that follows, the type of user is followed by the method returned: • Password-authenticated enterprise user, local database user, or SYSDBA/SYSOPER using Password file; proxy with username using password: PASSWORD • Kerberos-authenticated enterprise or external user: KERBEROS • SSL-authenticated enterprise or external user: SSL • RADIUS-authenticated external user: RADIUS • OS-authenticated external user or SYSDBA/SYSOPER: OS • DCE-authenticated external user: DCE • Proxy with certificate, DN, or username without using password: NONE • Background process (job queue slave process): JOB You can use IDENTIFICATION_TYPE to distinguish between external and enterprise users when the authentication method is Password, Kerberos, or SSL.
BG_JOB_ID	Is the Job ID of the current session if it is established by an Oracle database background process; null, if the session is not established by a background process
CDB_NAME	If queried while connected to a multitenant container database (CDB), returns the name of the CDB. Otherwise, returns NULL.
CLIENT_IDENTIFIER	Returns an identifier that is set by the application through the DBMS_SESSION.SET_IDENTIFIER procedure, the OCI_ATTR_CLIENT_IDENTIFIER OCI attribute, or the Oracle.jdbc.OracleConnection.setClientIdentifier Java class. This attribute is used by various database components to identify lightweight application users who authenticate as the same database user.
CLIENT_INFO	Returns up to 64 bytes of user session information that can be stored by an application by using the DBMS_APPLICATION_INFO package
C LIENT_PROGRAM_NAME	The name of the program used for the database session
CON_ID	If queried while connected to a CDB, returns the current container ID. Otherwise, returns 0.
CON_NAME	If queried while connected to a CDB, returns the current container name. Otherwise, returns the name of the database as specified in the DB_NAME initialization parameter.

Predefined Parameters of the USERENV Namespace (continued)

CURRENT_BIND	The bind variables for fine-grained auditing
CURRENT_EDITION_ID	The identifier of the current edition
CURRENT_EDITION_NAME	The name of the current edition
CURRENT_SCHEMA	Name of the default schema being used in the current schema. This value can be changed during the session with an ALTER SESSION SET CURRENT_SCHEMA statement.
CURRENT_SCHEMAID	Is the identifier of the default schema being used in the current session
CURRENT_SQL CURRENT_SQL*n*	CURRENT_SQL returns the first 4 KB of the current SQL that triggered the fine-grained auditing event. The CURRENT_SQL*n* attributes return subsequent 4 KB increments, where *n* can be an integer from 1 through 7, inclusive. CURRENT_SQL1 returns bytes 4 KB to 8 KB; CURRENT_SQL2 returns bytes 8 KB to 12 KB, and so on. You can specify these attributes only inside the event handler for the fine-grained auditing feature.
CURRENT_SQL_LENGTH	Is the length of the current SQL statement that triggers fine-grained audit or row-level security (RLS) policy functions or event handlers; valid only inside the function or event handler
CURRENT_USER	The name of the database user whose privileges are currently active. This may change during the duration of a session to reflect the owner of any active definer's rights object. When no definer's rights object is active, CURRENT_USER returns the same value as SESSION_USER. When used directly in the body of a view definition, this returns the user that is executing the cursor that is using the view; it does not respect views used in the cursor as being definer's rights.
CURRENT_USERID	The identifier of the database user whose privileges are currently active
DATABASE_ROLE	The database role using the SYS_CONTEXT function with the USERENV namespace. The role is one of the following: PRIMARY, PHYSICAL STANDBY, LOGICAL STANDBY, SNAPSHOT STANDBY.
DB_DOMAIN	Is the domain of the database as specified in the DB_DOMAIN initialization parameter
DB_NAME	Is the name of the database as specified in the DB_NAME initialization parameter
DB_SUPPLEMENTAL_LOG_LEVEL	If supplemental logging is enabled, returns a string containing the list of enabled supplemental logging levels. Possible values are: ALL_COLUMN, FOREIGN_KEY, MINIMAL, PRIMARY_KEY, PROCEDURAL, and UNIQUE_INDEX. If supplemental logging is not enabled, returns NULL.

Predefined Parameters of the USERENV Namespace (continued)

DB_UNIQUE_NAME	Is the name of the database as specified in the DB_UNIQUE_NAME initialization parameter
DBLINK_INFO	Returns the source of a database link session. Specifically, it returns a string of the form: SOURCE_GLOBAL_NAME=dblink_src_global_name, DBLINK_NAME=dblink_name, SOURCE_AUDIT_SESSIONID=dblink_src_audit_sessionid where: dblink_src_global_name is the unique global name of the source database dblink_name is the name of the database link on the source database dblink_src_audit_sessionid is the audit session ID of the session on the source database that initiated the connection to the remote database using dblink_name
ENTRYID	Is the current audit entry number. The audit entryid sequence is shared between fine-grained audit records and regular audit records. You cannot use this attribute in distributed SQL statements.
ENTERPRISE_IDENTITY	Returns the user's enterprisewide identity: • For enterprise users: The Oracle Internet Directory DN • For external users: The external identity (Kerberos principal name, RADIUS and DCE schema names, OS username, Certificate DN) • For local users and SYSDBA/SYSOPER logins: NULL The value of the attribute differs by proxy method: • For a proxy with DN: The Oracle Internet Directory DN of the client • For a proxy with certificate: The certificate DN of the client for external users; the Oracle Internet Directory DN for global users • For a proxy with username: The Oracle Internet Directory DN if the client is an enterprise user; NULL if the client is a local database user
FG_JOB_ID	Is the job ID of the current session if it is established by a client foreground process; Null, if the session is not established by a foreground process.
GLOBAL_CONTEXT_MEMORY	Returns the number being used in the system global area by the globally accessed context

Predefined Parameters of the USERENV Namespace (continued)

GLOBAL_UID	Returns the global user ID from Oracle Internet Directory for Enterprise User Security (EUS) logins; returns null for all other logins
HOST	Is the name of the host machine from which the client has connected
IDENTIFICATION_TYPE	Returns the way the user's schema was created in the database. Specifically, it reflects the IDENTIFIED clause in the CREATE/ALTER USER syntax. In the list that follows, the syntax used during schema creation is followed by the identification type returned: • IDENTIFIED BY password: LOCAL • IDENTIFIED EXTERNALLY: EXTERNAL • IDENTIFIED GLOBALLY: GLOBAL SHARED • IDENTIFIED GLOBALLY AS DN: GLOBAL PRIVATE
INSTANCE	Is the instance identification number of the current instance
INSTANCE_NAME	Is the name of the instance
IP_ADDRESS	Is the IP address of the machine from which the client is connected. If the user does not connect through the listener, this attribute is NULL.
IS_APPLY_SERVER	Returns TRUE if queried from within a SQL Apply server in a logical standby database. Otherwise, returns FALSE.
IS_DG_ROLLING_UPGRADE	Returns TRUE if a rolling upgrade of the database software in a Data Guard configuration, initiated by way of the DBMS_ROLLING package, is active. Otherwise, returns FALSE.
ISDBA	Returns TRUE if the user has been authenticated as having DBA privileges either through the operating system or through a password file
LANG	Is the ISO abbreviation for the language name; a shorter form than the existing LANGUAGE parameter
LANGUAGE	Is the language and territory currently used by your session, along with the database character set. In this form: language_territory.characterset
MODULE	Is the application name (module) set through the DBMS_APPLICATION_INFO package or OCI
NETWORK_PROTOCOL	Is the network protocol being used for communication, as specified in the 'PROTOCOL=*protocol*' portion of the connect string
NLS_CALENDAR	Is the current calendar of the current session
NLS_CURRENCY	Is the currency of the current session
NLS_DATE_FORMAT	Is the date format for the session

Predefined Parameters of the USERENV Namespace (continued)

NLS_DATE_LANGUAGE	Is the language used for expressing dates
NLS_SORT	Is BINARY or the linguistic sort basis
NLS_TERRITORY	Is the territory of the current session
ORACLE_HOME	The full path name for the Oracle home directory.
OS_USER	Is the operating system username of the client process that initiated the database session
PLATFORM_SLASH	The slash character that is used as the file path delimiter for your platform.
POLICY_INVOKER	Is the invoker of row-level security (RLS) policy functions
PROXY_ENTERPRISE_IDENTITY	Returns the Oracle Internet Directory DN when the proxy user is an enterprise user
PROXY_USER	Is the name of the database user who opened the current session on behalf of SESSION_USER
PROXY_USERID	Is the identifier of the database user who opened the current session on behalf of SESSION_USER
SCHEDULER_JOB	Returns Y if the current session belongs to a foreground job or background job. Otherwise, returns N.
SERVER_HOST	Is the host name of the machine on which the instance is running
SERVICE_NAME	Is the name of the service to which a given session is connected
SESSION_EDITION_ID	The identifier of the session edition
SESSION_EDITION_NAME	The name of the session edition
SESSION_USER	Is the database username by which the current user is authenticated. This value remains the same throughout the duration of the session.
SESSION_USERID	Is the identifier of the database username by which the current user is authenticated
SESSIONID	Is the auditing session identifier. You cannot use this attribute in distributed SQL statements.
SID	Is the session number (different from the session ID)
STATEMENTID	Is the auditing statement identifier. STATEMENTID represents the number of SQL statements audited in a given session.
TERMINAL	Is the operating system identifier for the client of the current session. In distributed SQL statements, this attribute returns the identifier for your local session. In a distributed environment, this is supported only for remote SELECT statements, not for remote INSERT, UPDATE, or DELETE operations. (The return length of this parameter

Predefined Parameters of the USERENV Namespace (continued)

	may vary by operating system.)

Predefined Parameters of the `USERENV` Namespace (continued)

Predefined Parameters of the `SYS_SESSION_ROLES` Namespace

Parameter	Return Value
`role_name`	Shows whether the role is currently enabled for the session or not. Returns FALSE when the role is not enabled.

Predefined Parameters of the USERENV Namespace (continued)