I am creating a container database (CDB) by name cdb1 and one pluggable database pdb1. These can be done by two ways.

i) By using Oracle Universal Installer (OUI)

ii) Manual creation By using SQL commands

6.1. Creating a Container Database

The Oracle Universal Installer (OUI) allows you to create a container database (CDB) during the software installation. The "Typical Install Configuration" screen has a checkbox to indicate the database is a container database. You can optionally create a single pluggable database (PDB) in this screen also.

Figure 6.1 Screen shot Oracle Database Typical Installation Configuration
The advanced configuration options provide the same ability on the "Database Identifiers" screen.

![Figure 6.2 Screen shot specifying Database Identifiers for accessing Database Uniquely](image)

In both cases the creation of a pluggable database (PDB) is optional, so you can create an empty container database and create the pluggable database later if you wish.

### 6.1.1. Database Configuration Assistant (DBCA)

The Database Configuration Assistant (DBCA) gives similar options to the Oracle Universal Installer (OUI). The "Creation Mode" page allows you to enter the default installation configuration details directly.
Figure 6.3 Screen shot Creating Database with Default configuration
If you chose the "Advanced Mode" option, you can create a CDB and multiple PBDs in one go.

**Figure 6.4** Screen shot creating a Container Database with one or more PDBs

### 6.2. Manual Creation

Using the DBCA, either during or after the software installation is the recommended approach to creating CDBs, but it is possible to create them manually.

We will place all datafiles, logfiles, controlfile of CDB in `/u02/oracle/cdb1` directory and place seed database pdbseed database datafiles in `/u02/oracle/cdb1/pdbseed` directory and pluggable database PDB1 files `/u02/oracle/cdb1/pdb1` directory

**Step 1: Make directories**

```
$mkdir /u02/oracle/cdb1
$cd /u02/oracle/cdb1
```
$mkdir pdb1
$mkdir fast_recovery_area
$mkdir pdbseed

Create directory for storing audit files

$mkdir $ORACLE_BASE/admin/cdb1
$cd $ORACLE_BASE/admin/cdb1
$mkdir adump

Create directory for diagnostic destination

$cd /u02/oracle/cdb1
$mkdir diag

Step 2: Create Parameter File by copying Template

$cd $ORACLE_HOME/dbs
$cp init.ora initcdb1.ora

Now open the PFile in vi editor and edit the require parameters. After editing it
Should look like as shown below
  db_name='cdb1'
  memory_target=500M
  processes = 150
  audit_file_dest=/opt/oracle/product/admin/cdb1/adump
  audit_trail='db'
  db_block_size=8192
  db_domain=
  db_recovery_file_dest=/u02/oracle/cdb1/fast_recovery_area'
  db_recovery_file_dest_size=2G
  diagnostic_dest=/u02/oracle/cdb1/diag
  dispatchers=(PROTOCOL=TCP) (SERVICE=CDB1XDB)
  open_cursors=300
  remote_login_passwordfile='EXCLUSIVE'
  undo_tablespace='UNDOTBS1'
  # You may want to ensure that control files are created on separate physical
  # devices
  control_files = '/u02/oracle/cdb1/control01.ora'
  compatible ='12.1.0'
  enable_pluggable_database=TRUE

Step 3- Set the ORACLE_SID environment variable and start the instance
$export ORACLE_SID=cdb1
$sqlplus
Enter User: / as sysdba
SQL>

**Step 4:- Give Create Database statement as follows**

```sql
CREATE DATABASE cdb1
   LOGFILE GROUP 1 '/u02/oracle/cdb1/log1.ora' size 10M,
   GROUP 2 '/u02/oracle/cdb1/log2.ora' size 10M
   EXTENT MANAGEMENT LOCAL
   DATAFILE '/u02/oracle/cdb1/system01.dbf'
   SIZE 700M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
   SYSAUX DATAFILE '/u02/oracle/cdb1/sysaux01.dbf'
   SIZE 550M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
   DEFAULT TABLESPACE deftbs
   DATAFILE '/u02/oracle/cdb1/deftbs01.dbf'
   SIZE 500M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED
   DEFAULT TEMPORARY TABLESPACE tempts1
   TEMPFILE '/u02/oracle/cdb1/temp01.dbf'
   SIZE 20M REUSE AUTOEXTEND ON NEXT 640K MAXSIZE UNLIMITED
   UNDO TABLESPACE undotbs1
   DATAFILE '/u02/oracle/cdb1/undotbs01.dbf'
   SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED
   ENABLE PLUGGABLE DATABASE
   SEED
   FILE_NAME_CONVERT = ('/u02/oracle/cdb1/',
                        '/u02/oracle/cdb1/pdbseed/')
   SYSTEM DATAFILES SIZE 125M AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
   SYSAUX DATAFILES SIZE 100M
   USER_DATA TABLESPACE usertbs
   DATAFILE '/u02/oracle/cdb1/pdbseed/usertbs01.dbf'
   SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

**Step 5:-**

Run the scripts. In Oracle 12c it is recommended to run catalog and catproc scripts through Perl program catcon.pl

Goto ORACLE_HOME/rdbms/admin directory. This is where all the scripts are available.

$cd $ORACLE_HOME/rdbms/admin
Give the following command at O/s prompt

```
$perl catcon.pl -d $ORACLE_HOME/rdbms/admin -b catalog_output catalog.sql
$perl catcon.pl -d $ORACLE_HOME/rdbms/admin -b catproc_output catproc.sql
$perl catcon.pl -d $ORACLE_HOME/rdbms/admin -b catblock_output catblock.sql
$perl catcon.pl -d $ORACLE_HOME/rdbms/admin -b catoctk_output catoctk.sql
$perl catcon.pl -d $ORACLE_HOME/rdbms/admin -b owminst_output owminst.plb
$perl catcon.pl -d $ORACLE_HOME/sqlplus/admin -b pupbld_output pupbld.sql
```

**Step 6:-**

See the status of Pluggable databases

SQL> show pdbs

<table>
<thead>
<tr>
<th>CON_ID</th>
<th>CON_NAME</th>
<th>OPEN MODE</th>
<th>RESTRICTED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PDB$SEED</td>
<td>READ ONLY</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Step 7:- Let’s us create the pluggable database PDB1**

To create a pluggable database give the following command

```
SQL>CREATE PLUGGABLE DATABASE pdb1 ADMIN USER pdb1adm IDENTIFIED BY tiger
STORAGE (MAXSIZE 2G MAX_SHARED_TEMP_SIZE 100M)
DEFAULT TABLESPACE users
DATAFILE '/u02/oracle/cdb1/pdb1/users01.dbf'
SIZE 250M AUTOEXTEND ON
FILE_NAME_CONVERT = ('/u02/oracle/cdb1/pdbseed/',
'/u02/oracle/cdb1/pdb1/');
```

To view list of PDB’s

SQL> show pdbs

<table>
<thead>
<tr>
<th>CON_ID</th>
<th>CON_NAME</th>
<th>OPEN MODE</th>
<th>RESTRICTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>PDB$SEED</td>
<td>READ ONLY</td>
<td>NO</td>
</tr>
<tr>
<td>3</td>
<td>PDB1</td>
<td>MOUNTED</td>
<td></td>
</tr>
</tbody>
</table>

Now open the pluggable database
SQL> alter pluggable database pdb1 open;

See the status
SQL> show pdb

<table>
<thead>
<tr>
<th>CON_ID</th>
<th>CON_NAME</th>
<th>OPEN MODE</th>
<th>RESTRICTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>PDB$SEED</td>
<td>READ ONLY</td>
<td>NO</td>
</tr>
<tr>
<td>3</td>
<td>PDB1</td>
<td>READ WRITE</td>
<td></td>
</tr>
</tbody>
</table>

To switch to Pluggable database give the following command
SQL> alter session set container=pdb1

To view current container, give the following command
SQL> show con_name

6.3. Creating USER Accounts and Connecting to CDB/PDB

You can create 2 types of users in Multitenant databases

1. Common User
2. Local User

Common User:- A common user is created in root CDB. Common user can connect to root CDB and all PDB’s including future PDB’s which you may plug.

You should not create any objects in Common User account as it will cause problems while connecting and disconnecting PDB’s

Local User:- A local user is created in a PDB database and he can connect has privileges in that PDB only.

Creating a common user account.

SQL> create user c##admin identified by adminpwd container=all;

SQL> grant connect, resource to c##admin;
Design and Development of an Advanced Database System with Multitenant Architecture,
Advanced Security Using Transparent Data Encryption & Data Redaction

CHAPTER 6 IMPLEMENTATIONS & RESULTS

SQL>conn c##admin/adminpwd

Creating a Local User

SQL>alter session set container=pdb1;
SQL>create user scott identified by tiger quota 50M on users;
SQL>grant connect,resource to scott;

6.4. Creating a pluggable database

The DBCA includes a new option on the opening "Database Operation" screen that allows you to manage the pluggable databases of an existing container database. Select the "Manage Pluggable Databases" option and click the "Next" button.

Figure 6.5 Screen shot managing a pluggable Database
6.4.1. Creating a Pluggable Database (PDB) using the DBCA

On the "Manage Pluggable Databases" screen shown previously, select the "Create a Pluggable Database" option and click the "Next" button. On the resulting screen, select the container database to house the new pluggable database and click the "Next" button.
Figure 6.7 Screen shot selecting a Container database for creating pluggable database

Select the "Create a new Pluggable Database" option and click the "Next" button. If you were plugging in a previously unplugged database, you would select the PDB Archive or PDB File Set options to match the format of the files containing the unplugged PDB.
Figure 6.8 Screen shot Creating a New Pluggable Database

Enter the pluggable database name, database location and admin credentials, then click the "Next" button.
Figure 6.9 Screen shot Creating Default user Tablespace for Pluggable database

If you are happy with the summary information, click the "Finish" button.
Figure 6.10 Screen shot Summary of Created Pluggable Database

Wait while the pluggable database is created. Once complete, click the "OK" button on the message dialog and the "Close" button on the main screen.
6.4.2. Unplug a Pluggable Database (PDB) using the DBCA

On the "Manage Pluggable Databases" screen shown previously, select the "Unplug a Pluggable Database" option and click the "Next" button. On the resulting screen, select the container database that houses the pluggable database to be unplugged and click the "Next" button.
Figure 6.12 Screen shot selecting a Container Database for Unplugging a Pluggable Database

Select the PDB to unplug, decide whether to use a pluggable database archive or a file set and enter the appropriate location details. Click the "Next" button.
Figure 6.13 Screen shot unplugging a Pluggable Database

If you are happy with the summary information, click the "Finish" button.
Figure 6.14 Screen shot Summary of Unplugged Database

Wait while the pluggable database is unplugged. Once complete, click the "OK" button on the message dialog and the "Close" button on the main screen.
6.5. Plug-in a Pluggable Database (PDB) using the DBCA

On the "Manage Pluggable Databases" screen shown previously, select the "Create a Pluggable Database" option and click the "Next" button. On the resulting screen, select the container database to house the new pluggable database and click the "Next" button.
Select the "Create Pluggable Database from PDB Archive" or "Create Pluggable Database using PDB File Set" option and enter the location of the required files. You can browse for the files using the "Browse" button.
Enter the pluggable database name, database location and admin credentials, then click the "Next" button.
Figure 6.18 Screen shot Creating PDB as a Clone

If you are happy with the summary information, click the "Finish" button.
Figure 6.19 Screen shot DCA Summary of Pluggable Database

Wait while the pluggable database is created. Once complete, click the "OK" button on the message dialog and the "Close" button on the main screen.
6.6. Applying Redaction policy using OEM wizard

Oracle Data Redaction provides functionality to mask (redact) data that is returned from user SELECT queries. The masking takes place in real time. The Data Redaction policy applies to the querying user, depending on this user's SYS_CONTEXT values. Oracle Database redacts only the data for the rows specified by the user's query, not the data for the entire column. The redaction takes place immediately before the data is returned to the querying user or application.
Figure 6.21 Screen shot OEM for Data Redaction policy
Creating a policy to redact the data from the salary column of EMPLOYEES table for all users apart from the user named SUPERVISOR:
Figure 6.23 Screen shot Redacting Employee Database
6.6.1. Verifying if the policy works as desired

[oracle@waheed ~]$ sqlplus funcionario/zaheer@localhost:1521/testpdb

SQL*Plus: Release 12.1.0.1.0 Production on Wed Apr 2 10:22:46 2014
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, Automatic Storage Management, OLAP, Advanced Analytics and Real Application Testing options

SQL> select EMPLOYEE_ID, FIRST_NAME, SALARY from hr.Employees WHERE ROWNUM < 5;

EMPLOYEE_ID  FIRST_NAME                      SALARY
---------------------  --------------------           ------------
100   Waheed                   0
101   Shagufta                  0
102   Shaista                    0
103   Ladly                       0

Querying the table again, this time connected as the user SUPERVISOR:

[oracle@waheed ~]$ sqlplus supervisor/zaheer@localhost:1521/testpdb

SQL*Plus: Release 12.1.0.1.0 Production on Wed Apr 2 10:26:04 2014
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, Automatic Storage Management, OLAP, Advanced Analytics and Real Application Testing options

SQL> select EMPLOYEE_ID, FIRST_NAME, SALARY from hr.Employees WHERE ROWNUM < 5;

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>FIRST_NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Waheed</td>
<td>24000</td>
</tr>
<tr>
<td>101</td>
<td>Shagufta</td>
<td>17000</td>
</tr>
<tr>
<td>102</td>
<td>Shaista</td>
<td>17000</td>
</tr>
<tr>
<td>103</td>
<td>Ladly</td>
<td>19000</td>
</tr>
</tbody>
</table>

6.7. Creating the Redaction policy using SQL * Plus

SQL>

BEGIN

 DBMS_REDACT.ADD_POLICY (OBJECT_SCHEMA => 'HR', object_name => 'EMPLOYEES', policy_name => 'POLITICA_TESTE', expression => 'SYS_CONTEXT(''USERENV'', 'SESSION_USER'') != ''SUPERVISOR'');

 DBMS_REDACT.ALTER_POLICY (OBJECT_SCHEMA => 'HR', object_name => 'EMPLOYEES', policy_name => 'POLITICA_TESTE', action => DBMS_REDACT.ADD_COLUMN, column_name => '"SALARY"', function_type => DBMS_REDACT.FULL);

END;

PL/SQL procedure successfully completed.
6.8. Creating the Redaction policy using SQL Developer

Figure 6.24 Screen shot Modifying Data Redaction
Figure 6.25 Screen shot Creating Data Redaction Dialog Box
6.9. SQL Script demonstrating Redaction Policy

To set up a demonstration, uses the setup.sql script, then uses the code shown in below listing to set up a redaction policy on the EMP table.

Setup.sql
create table emp
(   empno number,
    ename varchar2(20),
    salary number,
    hire_date date,
    email_id varchar2(30) ) ;
insert into emp(empno, ename, salary, hire_date, email_id)
values(1,'Zaheer',1013,sysdate,'Zaheer@gmail.com');
insert into emp(empno, ename, salary, hire_date, email_id)
values(2,'Shanu',1578,sysdate-2,'shanu@gmail.com');
insert into emp(empno, ename, salary, hire_date, email_id)
values(3,'Kaneez',1398,sysdate-3,'Kaneez@gmail.com');
commit;

SQL> select * from emp;

EMPNO  ENAME   SALARY  HIRE_DATE      EMAIL_ID
--------  -------  --------  -----------------  ----------------------
      1  Zaheer     1013   06-MAR-2015 02:20:42 PM Zaheer@gmail.com
      2   Shanu     1578   04-MAR-2015 02:20:42 PM shanu@gmail.com
      3 Kaneez     1398   03-MAR-2015 02:20:42 PM Kaneez@gmail.com

3 rows selected.

When the owner of the table "EMP" selects data, the result displays the values intact - but when any other user selects data from the table, the data must be masked.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description of redaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salary</td>
<td>replace with some random numbers and don't show actual values</td>
</tr>
<tr>
<td>Hire_date</td>
<td>show only the day and month year should be replaced with 1990</td>
</tr>
<tr>
<td>Email_id</td>
<td>Keep the domain name and rest others should be masked</td>
</tr>
</tbody>
</table>

Table 6.1 Description of Data redaction applied on Employee Database
SQL> grant select on emp to scott;
Grant succeeded.
SQL> begin
  2  dbms_redact.add_policy
  3     ( object_schema=>user,
  4       object_name=>'EMP',
  5       policy_name=>'EMP_REDACT',
  6       column_name=>'SALARY',
  7       function_type=>dbms_redact.random,
  8       expression=>'q|sys_context('userenv','current_user')!= 'WAHEED'||
  9   );
10  end;
11  /
PL/SQL procedure successfully completed.

SQL>

SQL> begin
  2  dbms_redact.alter_policy
  3     ( object_schema=> user,
  4       object_name=>'EMP',
  5       policy_name=>'EMP_REDACT',
  6       action=>dbms_redact.add_column,
  7       column_name=>'HIRE_DATE',
  8       function_type=>dbms_redact.partial,
  9       function_parameters=>'MDy1990' );
10  end;
11  /
PL/SQL procedure successfully completed.
SQL> begin
2     dbms_redact.alter_policy
3     ( object_schema=> user,
4       object_name=>'EMP',
5       policy_name=>'EMP_REDACT',
6       action=>dbms_redact.add_column,
7       column_name=>'EMAIL_ID',
8       function_type=>dbms_redact.regexp,
9       regexp_pattern=>dbms_redact.RE_PATTERN_EMAIL_ADDRESS,
10      regexp_replace_string=>dbms_redact.RE_REDACT_EMAIL_NAME,
11      regexp_match_parameter=>dbms_redact.RE_CASE_INSENSITIVE );
12  end;
13  /
PL/SQL procedure successfully completed.

SQL>
SQL> select * from emp ;

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>SALARY</th>
<th>HIRE_DATE</th>
<th>EMAIL_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Zaheer</td>
<td>1013</td>
<td>06-MAR-2015 02:20:42 PM</td>
<td><a href="mailto:Zaheer@gmail.com">Zaheer@gmail.com</a></td>
</tr>
<tr>
<td></td>
<td>Shanu</td>
<td>1578</td>
<td>04-MAR-2015 02:20:42 PM</td>
<td><a href="mailto:shanu@gmail.com">shanu@gmail.com</a></td>
</tr>
<tr>
<td></td>
<td>Kaneez</td>
<td>1398</td>
<td>03-MAR-2015 02:20:42 PM</td>
<td><a href="mailto:Kaneez@gmail.com">Kaneez@gmail.com</a></td>
</tr>
</tbody>
</table>

3 rows selected.

SQL>
When connected from Scott schema the redacted data look like this.

SQL> select * from WAHEED.emp;

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>ENAME</th>
<th>SALARY</th>
<th>HIRE_DATE</th>
<th>EMAIL_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zaheer</td>
<td>796</td>
<td>28-DEC-1990 04:10:34 PM</td>
<td><a href="mailto:xxxx@gmail.com">xxxx@gmail.com</a></td>
</tr>
<tr>
<td>2</td>
<td>Shanu</td>
<td>94</td>
<td>26-DEC-1990 04:10:34 PM</td>
<td><a href="mailto:xxxx@gmail.com">xxxx@gmail.com</a></td>
</tr>
<tr>
<td>3</td>
<td>Kaneez</td>
<td>79</td>
<td>25-DEC-1990 04:10:34 PM</td>
<td><a href="mailto:xxxx@gmail.com">xxxx@gmail.com</a></td>
</tr>
</tbody>
</table>

3 rows selected.

SQL>

Information's on redaction policies and policy columns are maintained in REDACTION_POLICIES and REDACTION_COLUMNS data dictionary.

SQL> @printtbl ' select * from redaction_policies '

OBJECT_OWNER : "WAHEED"
OBJECT_NAME : "EMP"
POLICY_NAME : "EMP_REDACT"
EXPRESSION : "sys_context('userenv','current_user')!= 'WAHEED'"
ENABLE : "YES"
POLICY_DESCRIPTION : ""

-----------------

SQL> PL/SQL procedure successfully completed.

SQL> @printtbl ' select * from redaction_columns '

OBJECT_OWNER : "WAHEED"
OBJECT_NAME : "EMP"
COLUMN_NAME : "SALARY"
FUNCTION_TYPE : "RANDOM REDACTION"
FUNCTION_PARAMETERS : ""
REGEXP_PATTERN : ""
REGEXP_REPLACE_STRING : ""
REGEXP_POSITION : "0"
REGEXP_OCCURRENCE : "0"
REGEXP_MATCH_PARAMETER : ""


COLUMN_DESCRIPTION : ""
---------------------
OBJECT_OWNER : "WAHEED"
OBJECT_NAME : "EMP"
COLUMN_NAME : "EMAIL_ID"
FUNCTION_TYPE : "REGEXP REDACTION"
FUNCTION_PARAMETERS : ""
REGEXP_PATTERN : "((A-Za-z0-9._%+-\+)@([A-Za-z0-9.-]+\.[A-Za-z]{2,4})"
REGEXP_REPLACE_STRING : "xxxx@\2"
REGEXP_POSITION : "1"
REGEXP_OCCURRENCE : "0"
REGEXP_MATCH_PARAMETER : "i"
COLUMN_DESCRIPTION : ""
---------------------
OBJECT_OWNER : "WAHEED"
OBJECT_NAME : "EMP"
COLUMN_NAME : "HIRE_DATE"
FUNCTION_TYPE : "PARTIAL REDACTION"
FUNCTION_PARAMETERS : "MDy1990"
REGEXP_PATTERN : ""
REGEXP_REPLACE_STRING : ""
REGEXP_POSITION : "0"
REGEXP_OCCURRENCE : "0"
REGEXP_MATCH_PARAMETER : ""
COLUMN_DESCRIPTION : ""
---------------------
SQL> PL/SQL procedure successfully completed.

However If the querying user has the EXEMPT REDACTION POLICY system privilege, redaction will not be performed.

SQL> connect sys/oracle@pdb1 as sysdba
Connected.

sys@PDB1> grant exempt redaction policy to scott;
Grant succeeded.

sys@PDB1> set timing off
sys@PDB1>

sys@PDB1> grant exempt redaction policy to scott;
Grant succeeded.

sys@PDB1> connect scott/tiger@pdb1
Connected.

SQL>

SQL> select * from session_privs ;

PRIVILEGE
-------------
CREATE SESSION
EXEMPT REDACTION POLICY

2 rows selected.

SQL>

SQL> select * from WAHEED.emp;

EMPNO ENAME SALARY HIRE_DATE EMAIL_ID
---------- -------------------- ---------- -----------------------   -------------------
1 Zaheer 1013 06-MAR-2015 02:20:42 PM Zaheer@gmail.com
2 Shanu 1578 04-MAR-2015 02:20:42 PM shanu@gmail.com
3 Kaneez 1398 03-MAR-2015 02:20:42 PM Kaneez@gmail.com

3 rows selected.