Multitenant : Create and Configure a Pluggable Database (PDB) in Oracle Database 12c Release 1 (12.1)

The multitenant option introduced in Oracle Database 12c allows a single container database (CDB) to host multiple separate pluggable databases (PDB). This article covers the options available to create a new pluggable database in an existing container database.

There are articles specifically about installation of Oracle Database 12c here (articles-12c#database-12cr1-installations).

- Oracle Universal Installer (OUI)
- Database Configuration Assistant (DBCA)
  - Create a Pluggable Database (PDB) using the DBCA
  - Unplug a Pluggable Database (PDB) using the DBCA
  - Plugin a Pluggable Database (PDB) using the DBCA
  - Delete a Pluggable Database (PDB) using the DBCA
  - Configure a Pluggable Database (PDB) using the DBCA
- Manual (SQL*Plus)
  - Create a Pluggable Database (PDB) Manually
  - Unplug a Pluggable Database (PDB) Manually
  - Plugin a Pluggable Database (PDB) Manually
  - Clone a Pluggable Database (PDB) Manually
  - Clone a Pluggable Database (PDB) Manually (Metadata-Only : NO DATA)
  - Delete a Pluggable Database (PDB) Manually
- SQL Developer
- Cloud Control

Related articles.

- Multitenant : Create a Pluggable Database (https://www.youtube.com/watch?v=dPHerZHvUyk)
- Multitenant : Unplug and Plugin a Pluggable Database (https://www.youtube.com/watch?v=yMBcKhWFMwE)

- Multitenant : Migrate a Non-Container Database (CDB) to a Pluggable Database (PDB) in Oracle Database 12c Release 1 (12.1) (multitenant-migrate-non-cdb-to-pdb-12cr1)
- Multitenant : Clone a Remote PDB or Non-CDB in Oracle Database 12c (12.1.0.2) (multitenant-clone-remote-pdb-or-non-cdb-12cr1)
- Multitenant : Configure Instance Parameters and Modify Container Databases (CDB) and Pluggable Databases (PDB) in Oracle Database 12c Release 1 (12.1) (multitenant-configure-instance-parameters-of-cdb-and-pdb-12cr1)
- Multitenant : Metadata Only PDB Clones in Oracle Database 12c Release 1 (12.1.0.2) (multitenant-metadata-only-pdb-clones-12cr1)
- Multitenant : PDB Subset Cloning in Oracle Database 12c Release 1 (12.1.0.2) (multitenant-pdb-subset-cloning-12cr1)

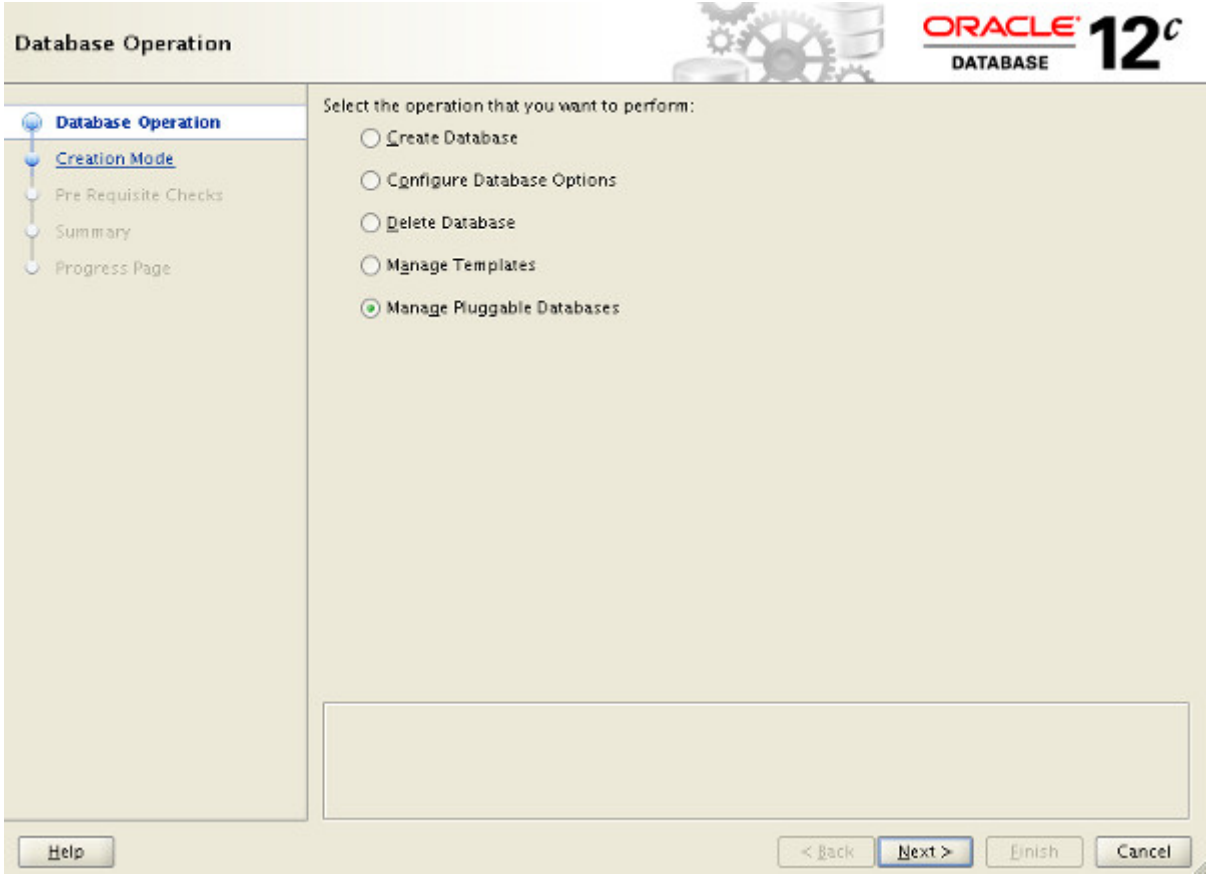
Oracle Universal Installer (OUI)

In a previous article (multitenant-create-and-configure-container-database-12cr1) we discussed the creation of a pluggable database (PDB) during the creation of the container database (CDB) during the installation of the Oracle software using the Oracle Universal Installer (OUI). This topic will not be repeated here, so please refer to the Container Database (multitenant-create-and-configure-container-database-12cr1) article for more information.

Database Configuration Assistant (DBCA)

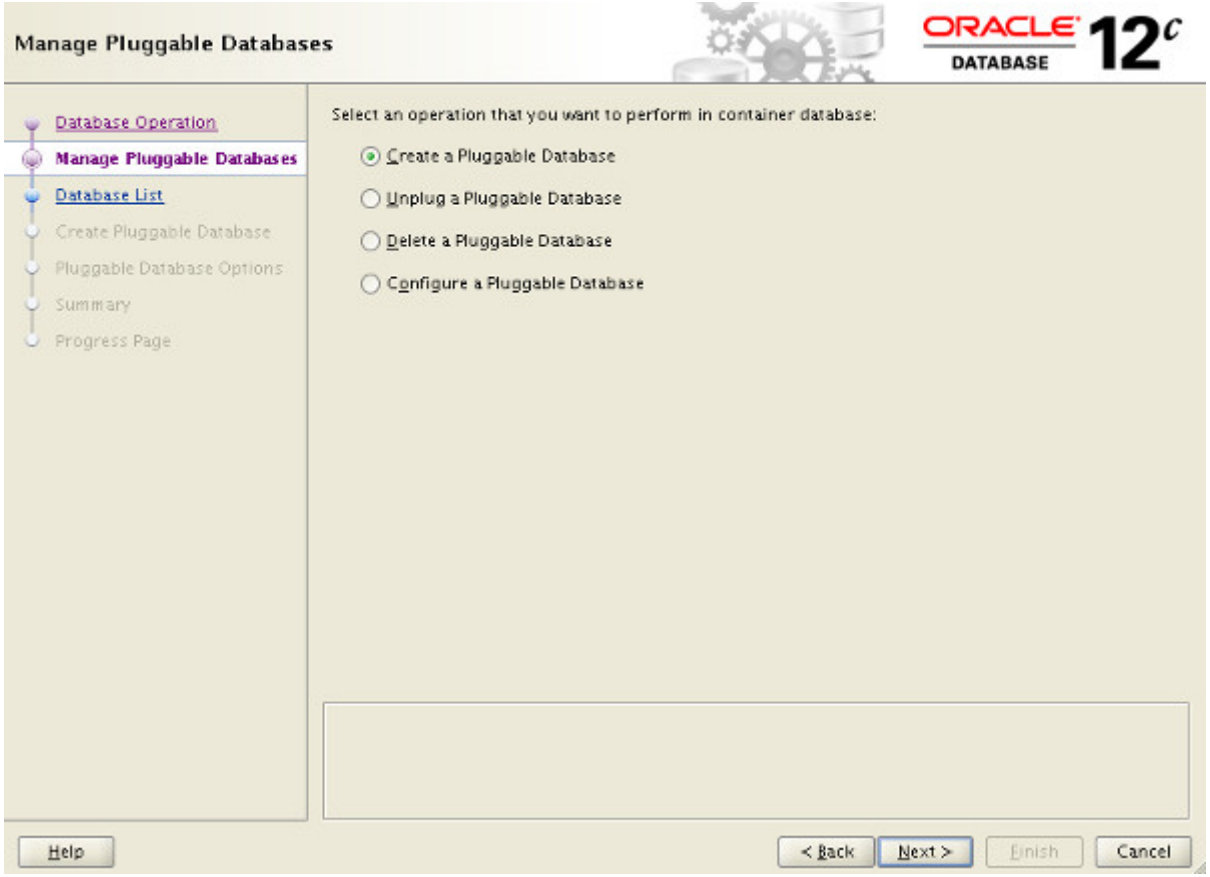
In a previous article (multitenant-create-and-configure-container-database-12cr1) we discussed the creation of a pluggable database (PDB) during the creation of the container database (CDB) using the Database Configuration Assistant (DBCA). This topic will not be repeated here, so please refer to the Container Database (multitenant-create-and-configure-container-database-12cr1) article for more information.

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.



You can see from the resulting screen what operations are possible with pluggable databases.

Translate

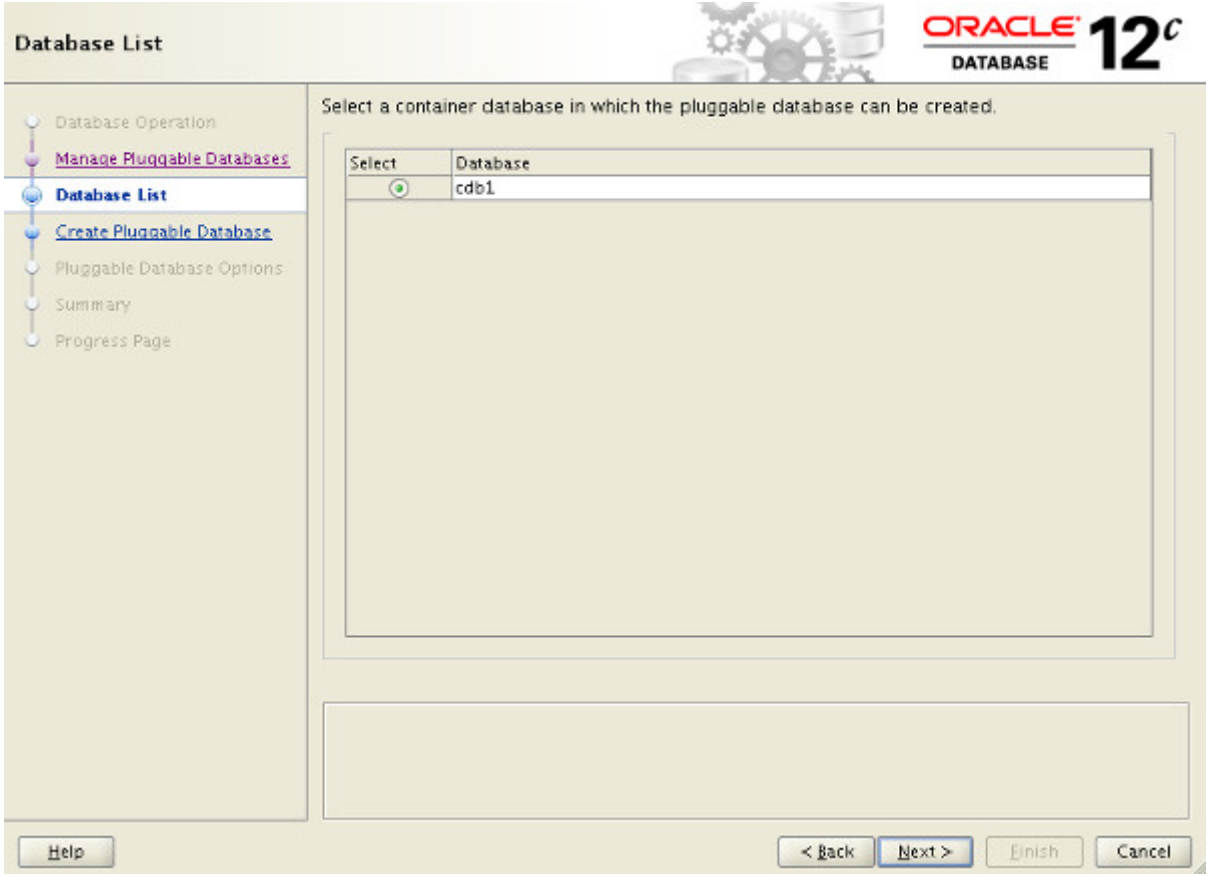


The following sections describe some of these options.

Create a Pluggable Database (PDB) using the DBCA

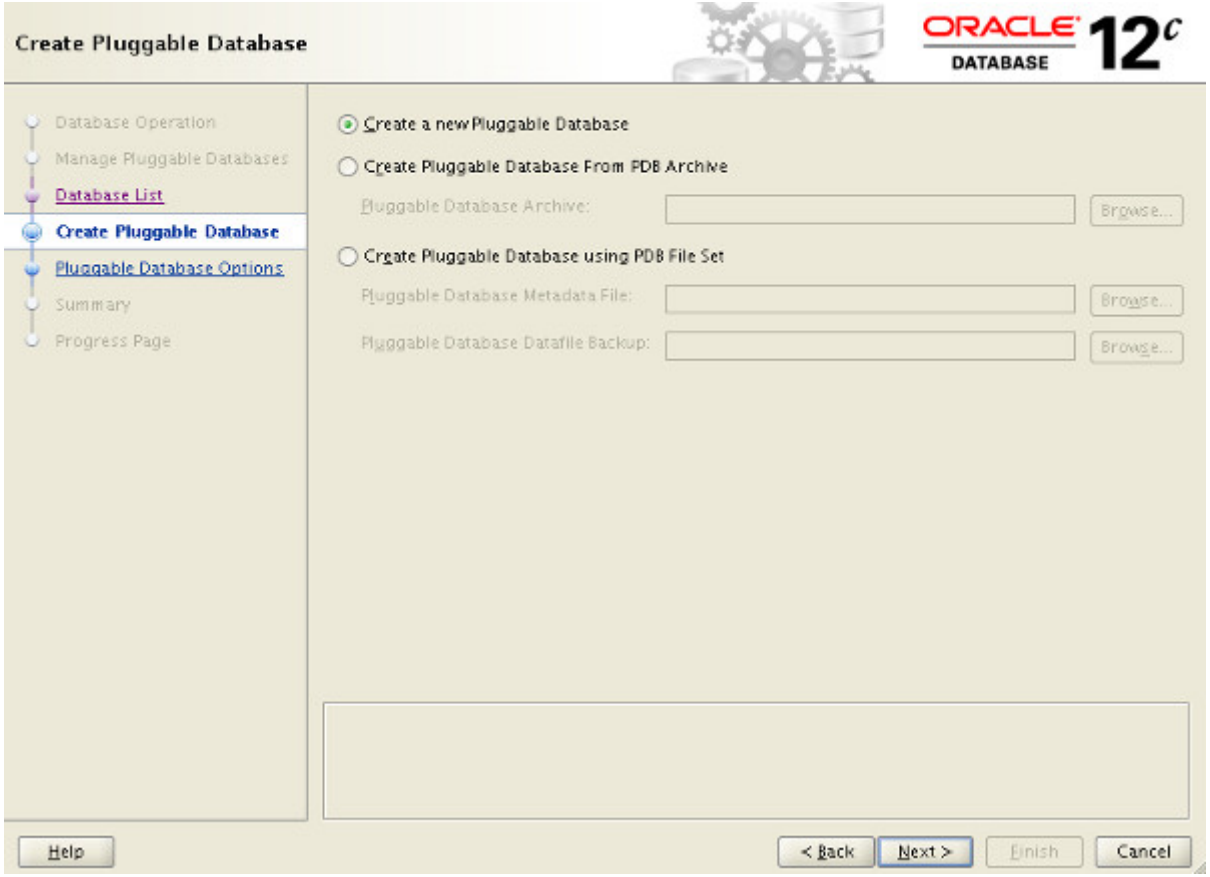
On the "Manage Pluggable Databases" screen shown previously, select the "Create a Pluggable Database" option and click the "Next" button. On the resulting screen, select the container database to house the new pluggable database and click the "Next" button.

Translate



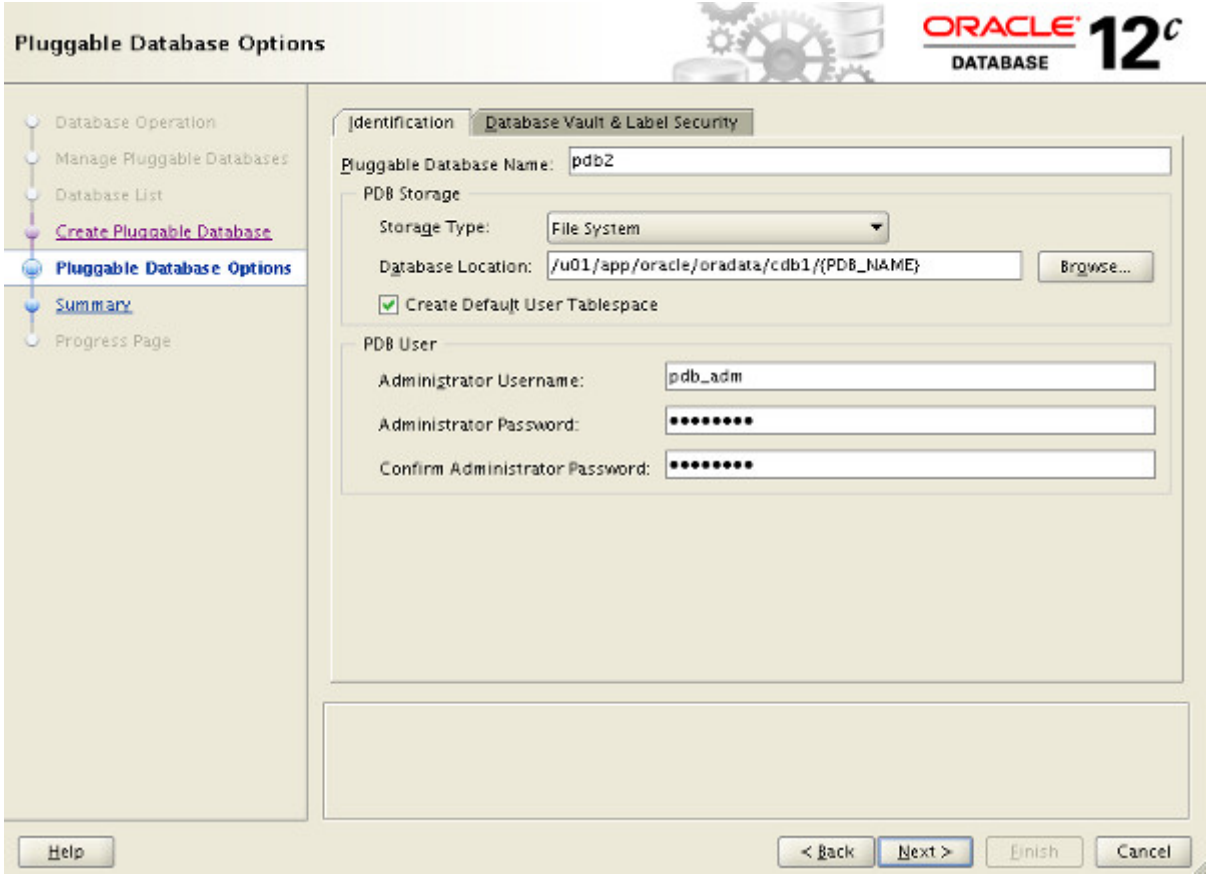
Select the "Create a new Pluggable Database" option and click the "Next" button. If you were plugging in a previously unplugged database, you would select the PDB Archive or PDB File Set options to match the format of the files containing the unplugged PDB.

Translate



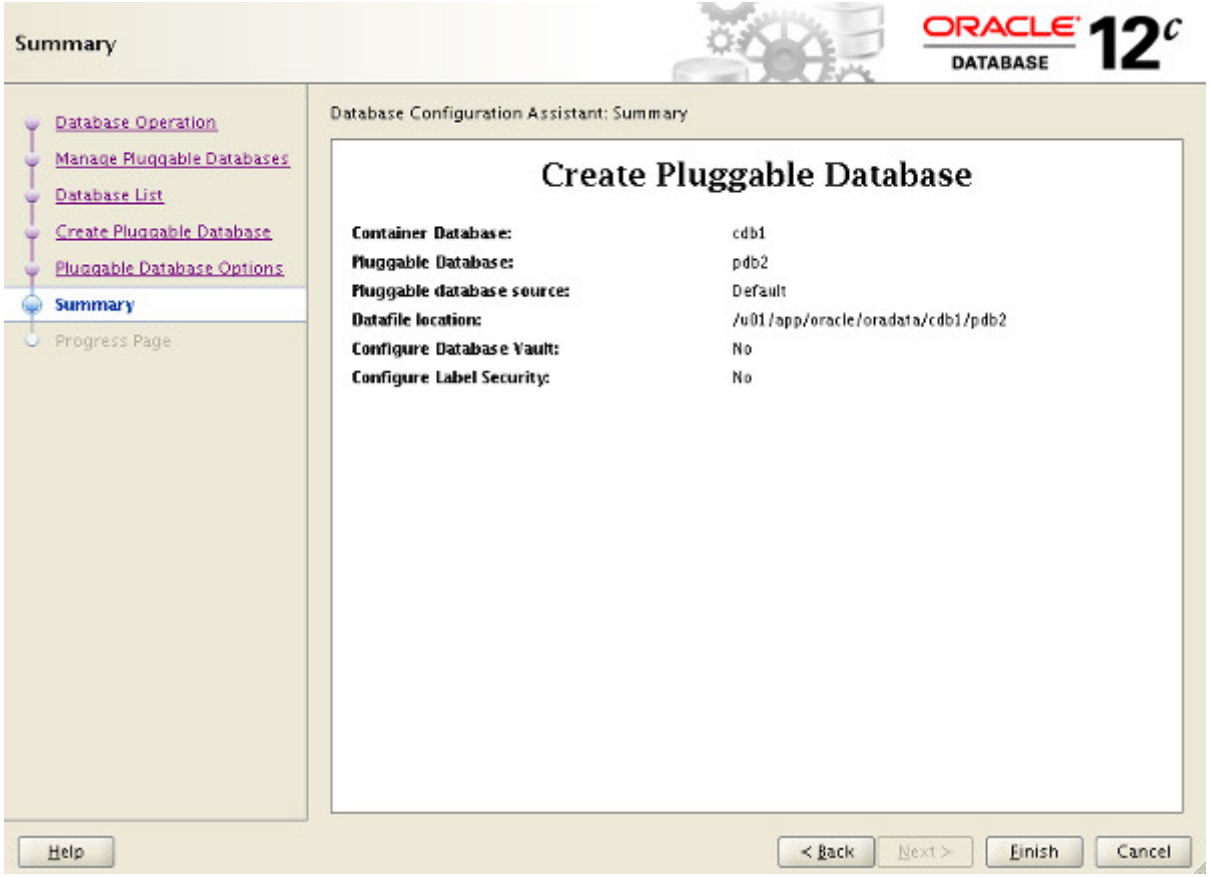
Enter the pluggable database name, database location and admin credentials, then click the "Next" button.

Translate



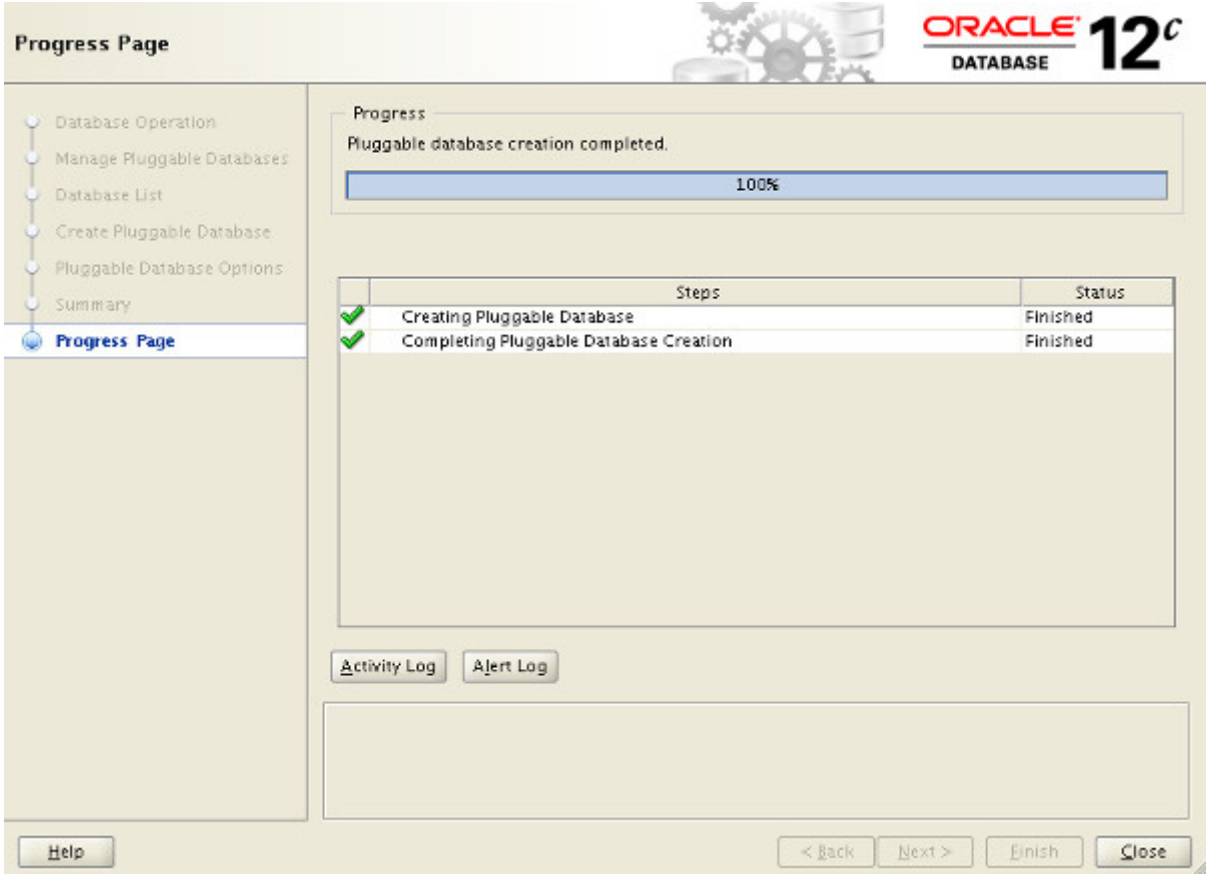
If you are happy with the summary information, click the "Finish" button.

Translate



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.

Translate

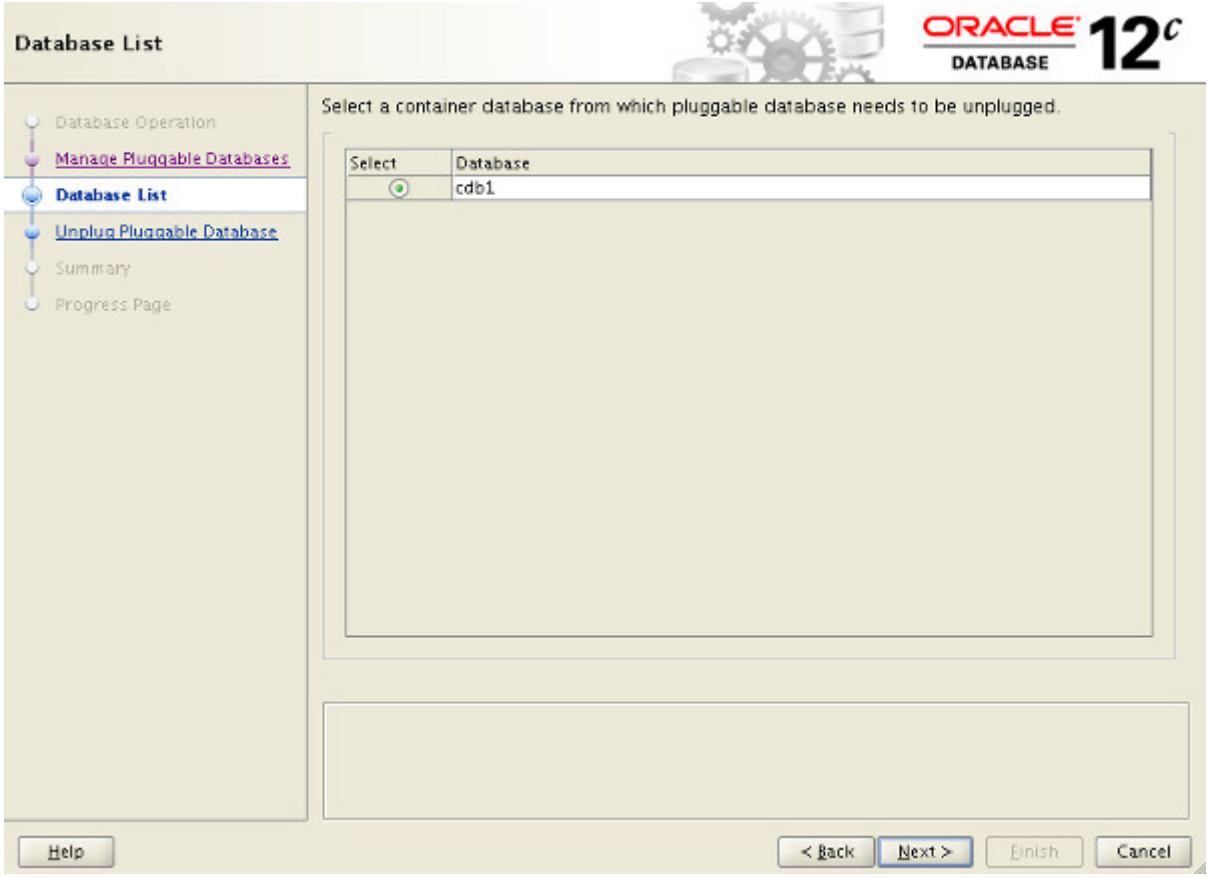


The new pluggable database has been created as a clone of the seed database.

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.

Translate

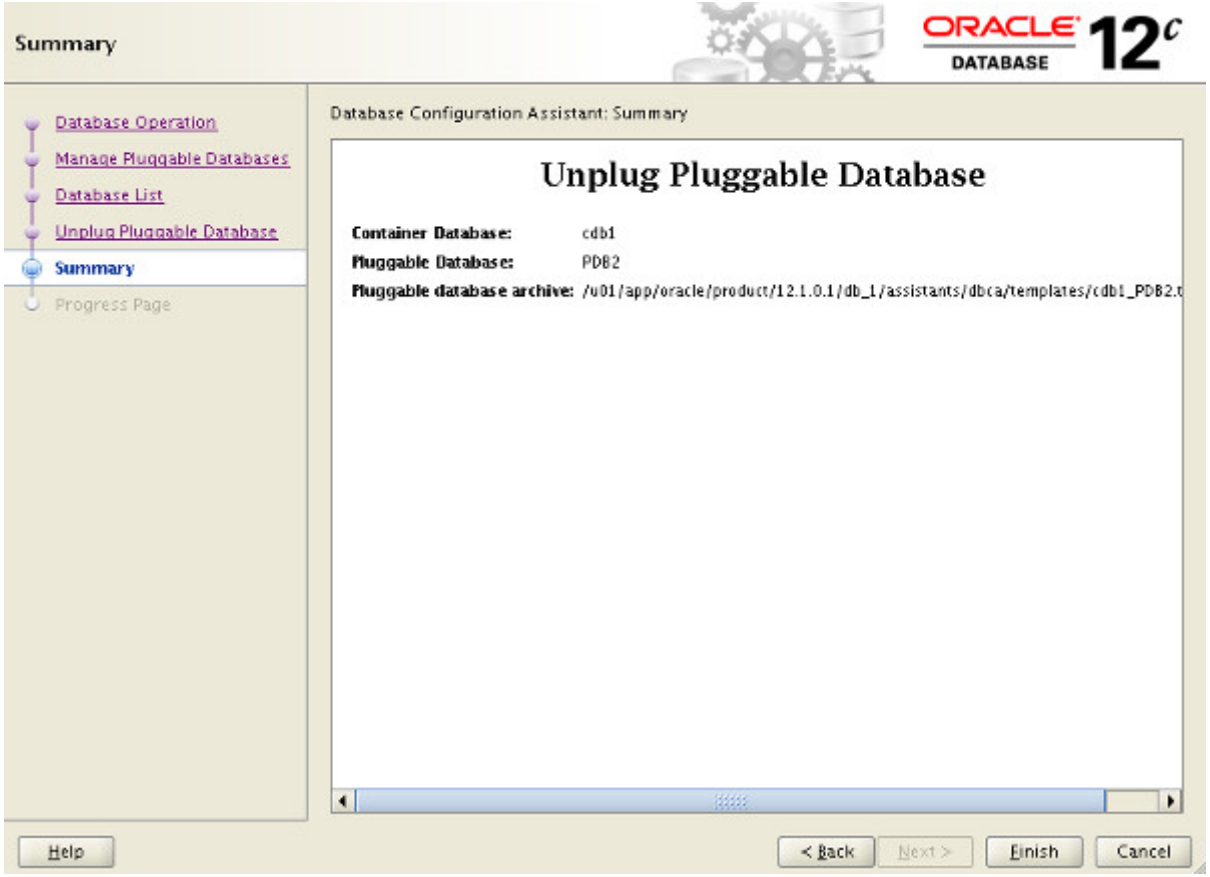


Select the PDB to unplug, decide whether to use a pluggable database archive or a file set and enter the appropriate location details. Click the "Next" button.

Translate

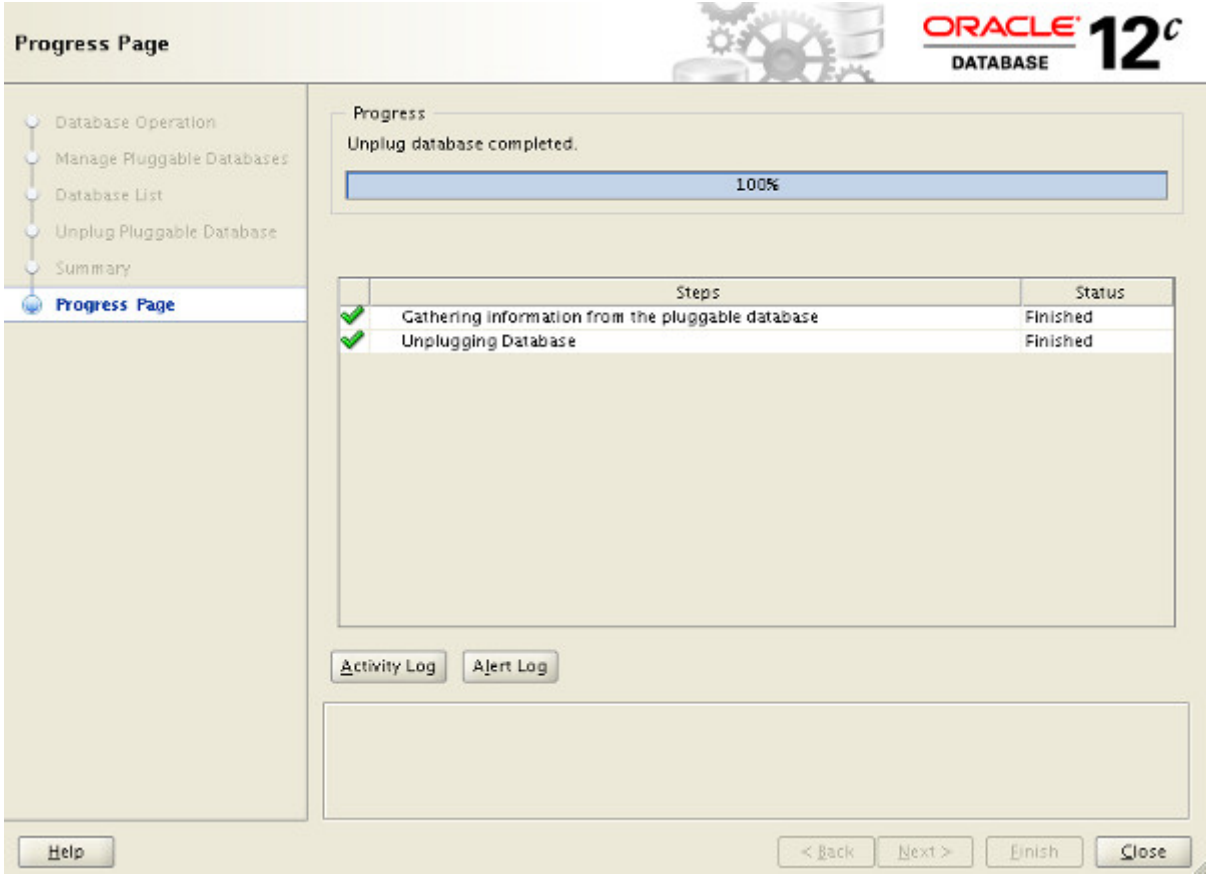
If you are happy with the summary information, click the "Finish" button.

Translate



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.

Translate

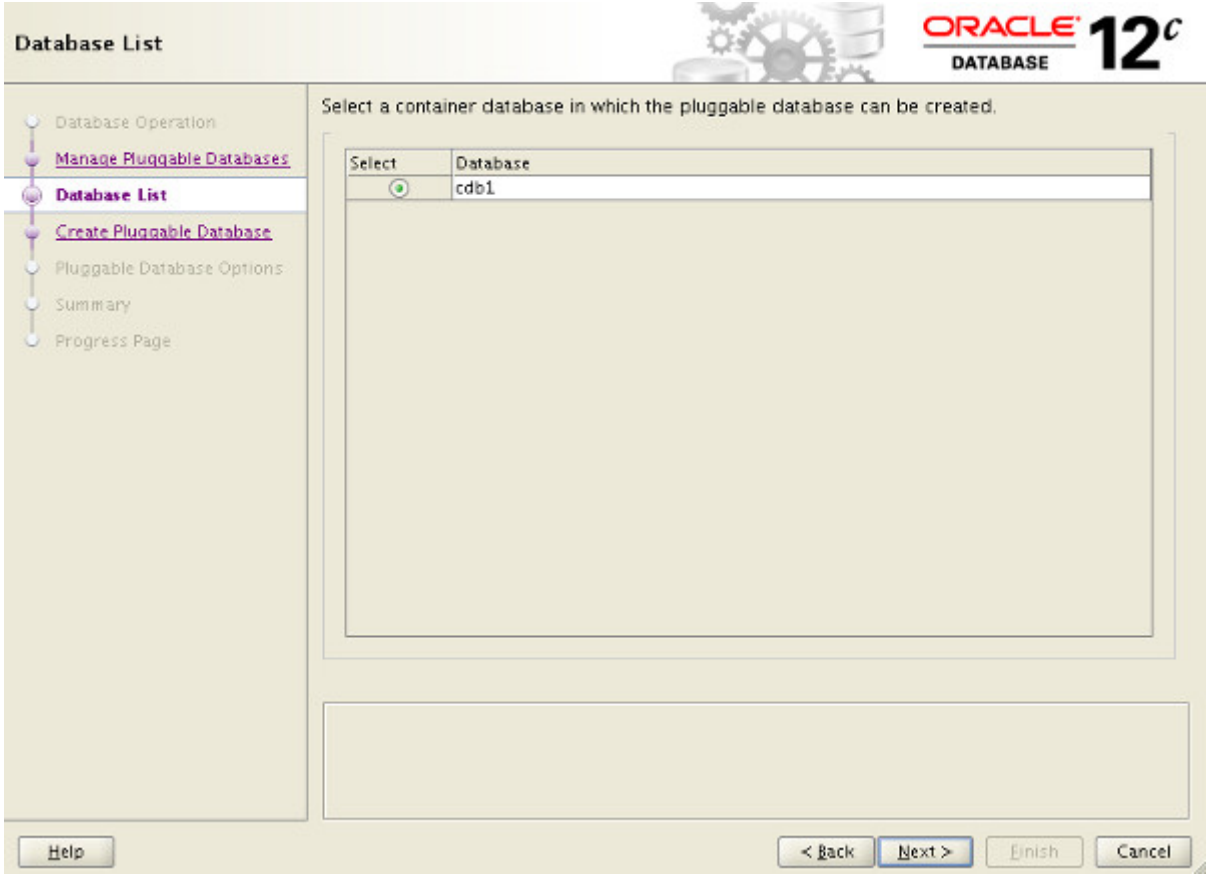


The pluggable database has now been unplugged.

Plugin a Pluggable Database (PDB) using the DBCA

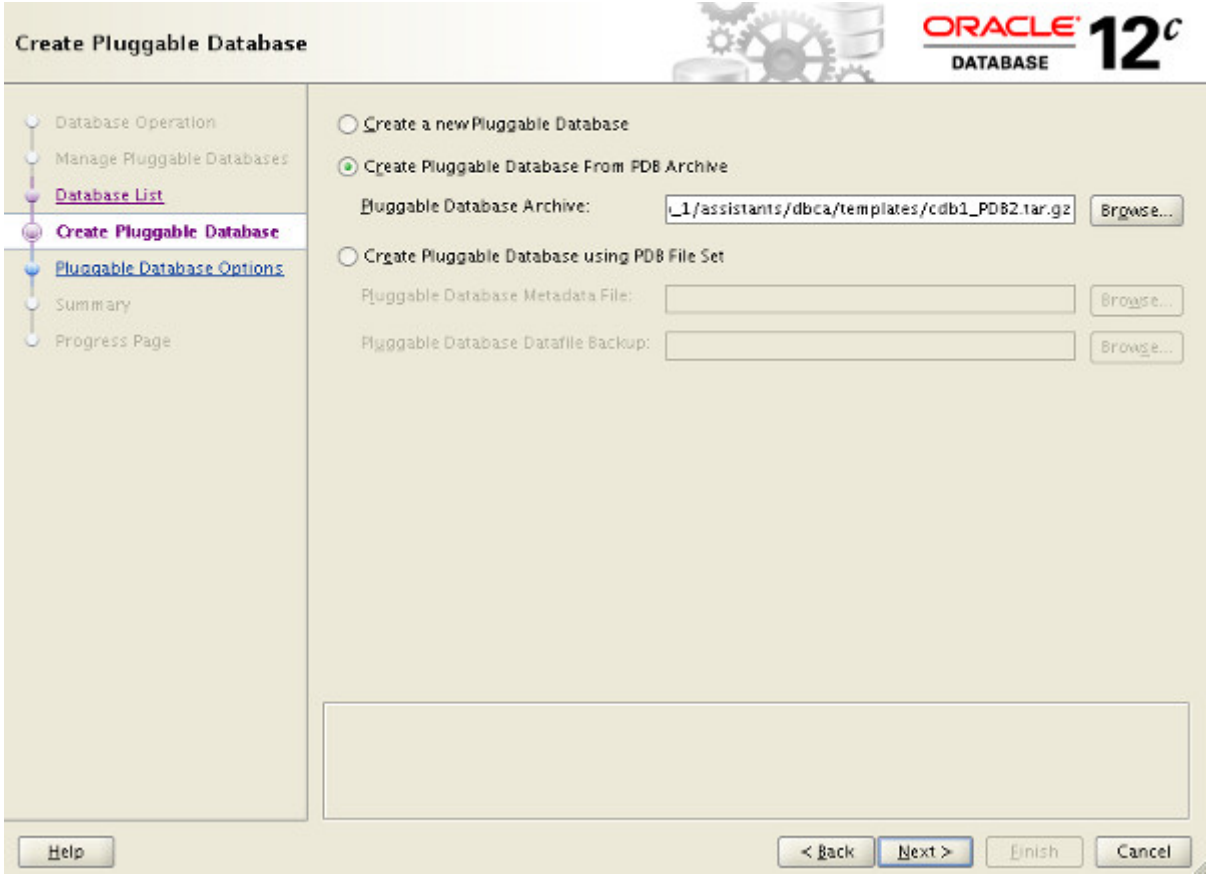
On the "Manage Pluggable Databases" screen shown previously, select the "Create a Pluggable Database" option and click the "Next" button. On the resulting screen, select the container database to house the new pluggable database and click the "Next" button.

Translate



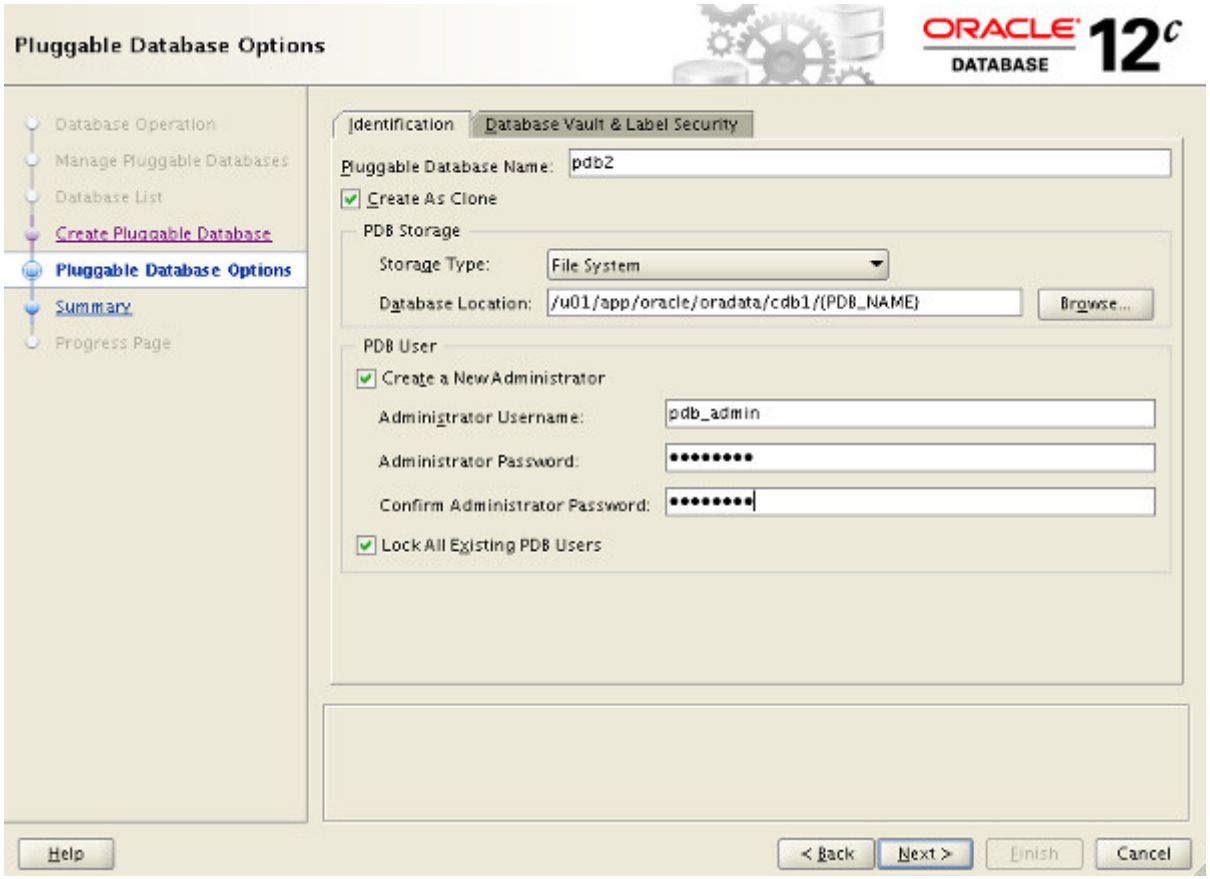
Select the "Create Pluggable Database From PDB Archive" or "Create Pluggable Database using PDB File Set" option and enter the location of the required files. You can browse for the files using the "Browse" button.

Translate



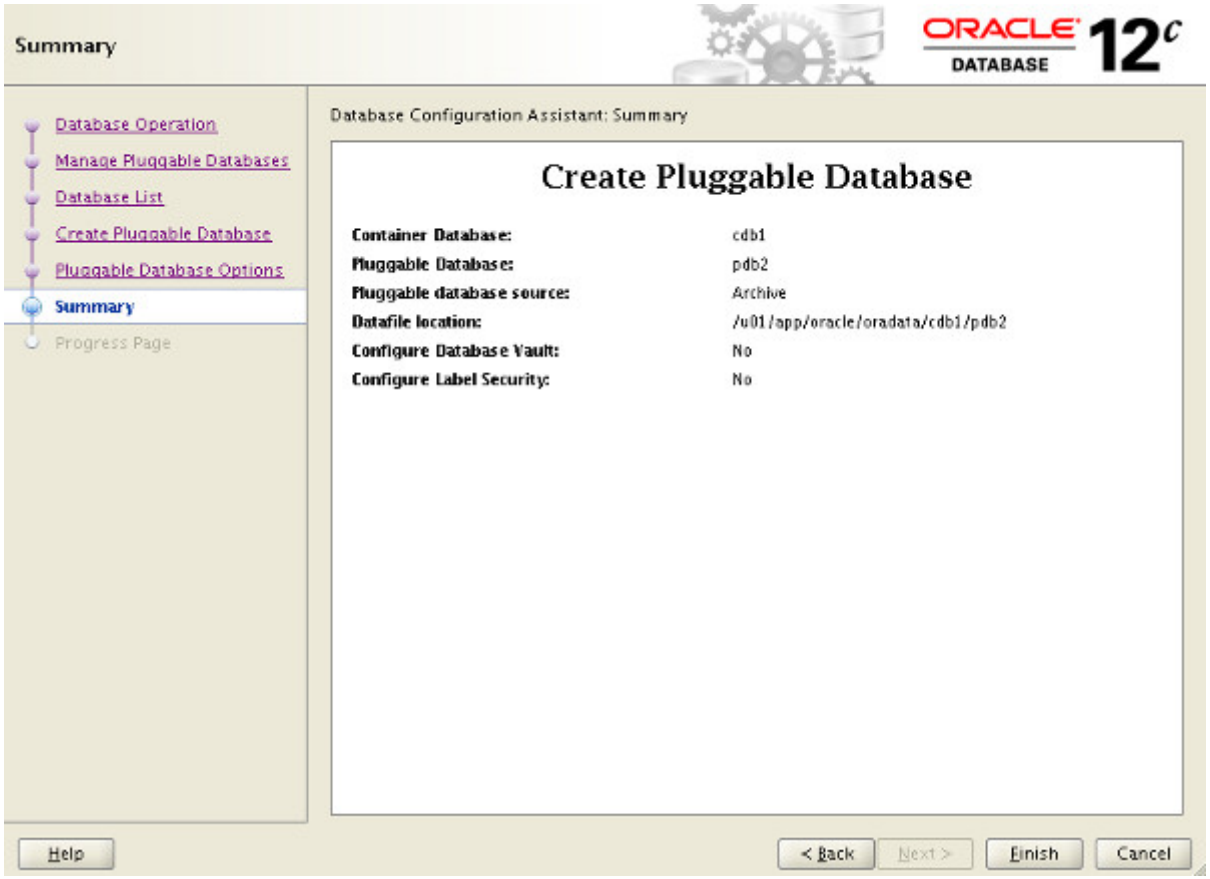
Enter the pluggable database name, database location and admin credentials, then click the "Next" button.

Translate



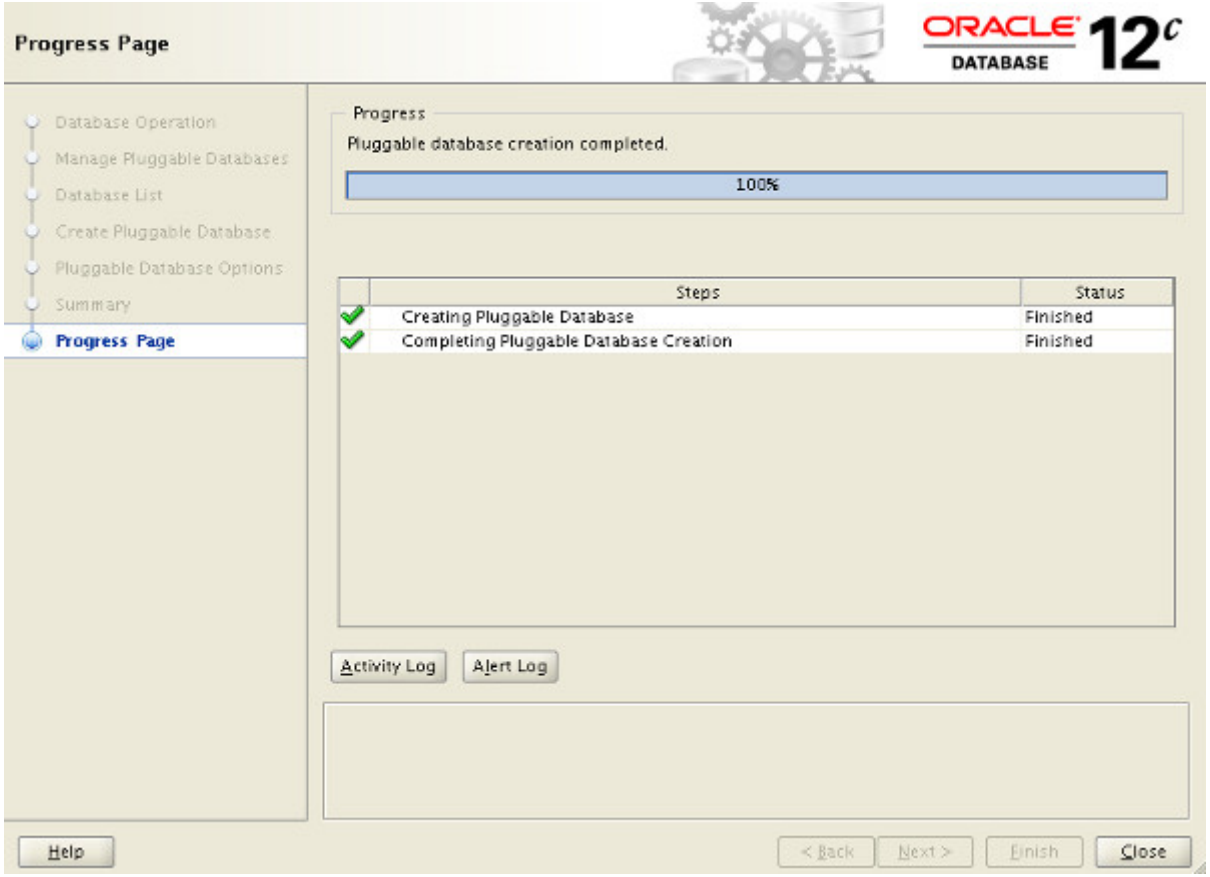
If you are happy with the summary information, click the "Finish" button.

Translate



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.

Translate

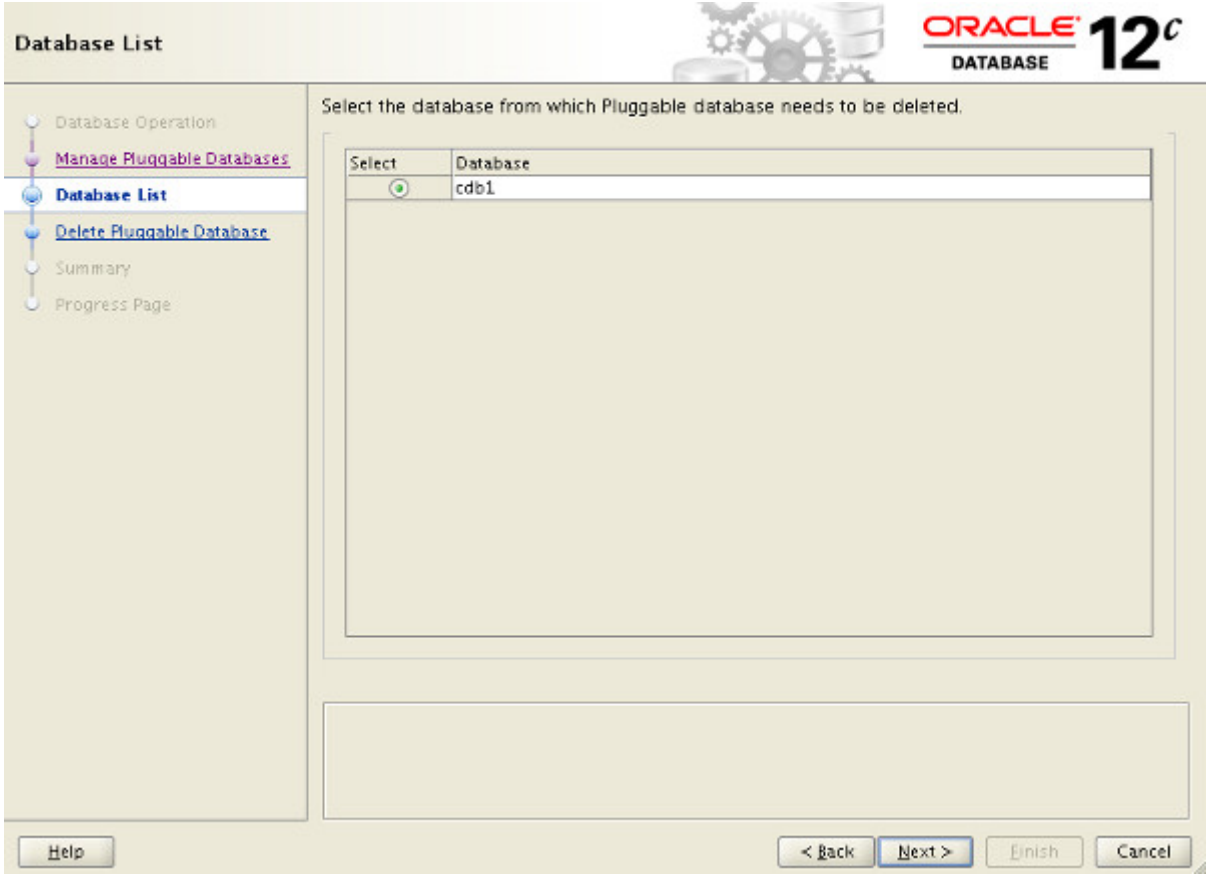


The pluggable database has been plugged into the container database.

Delete a Pluggable Database (PDB) using the DBCA

On the "Manage Pluggable Databases" screen shown previously, select the "Delete a Pluggable Database" option and click the "Next" button. On the resulting screen, select the container database that houses the pluggable database to be deleted and click the "Next" button.

Translate

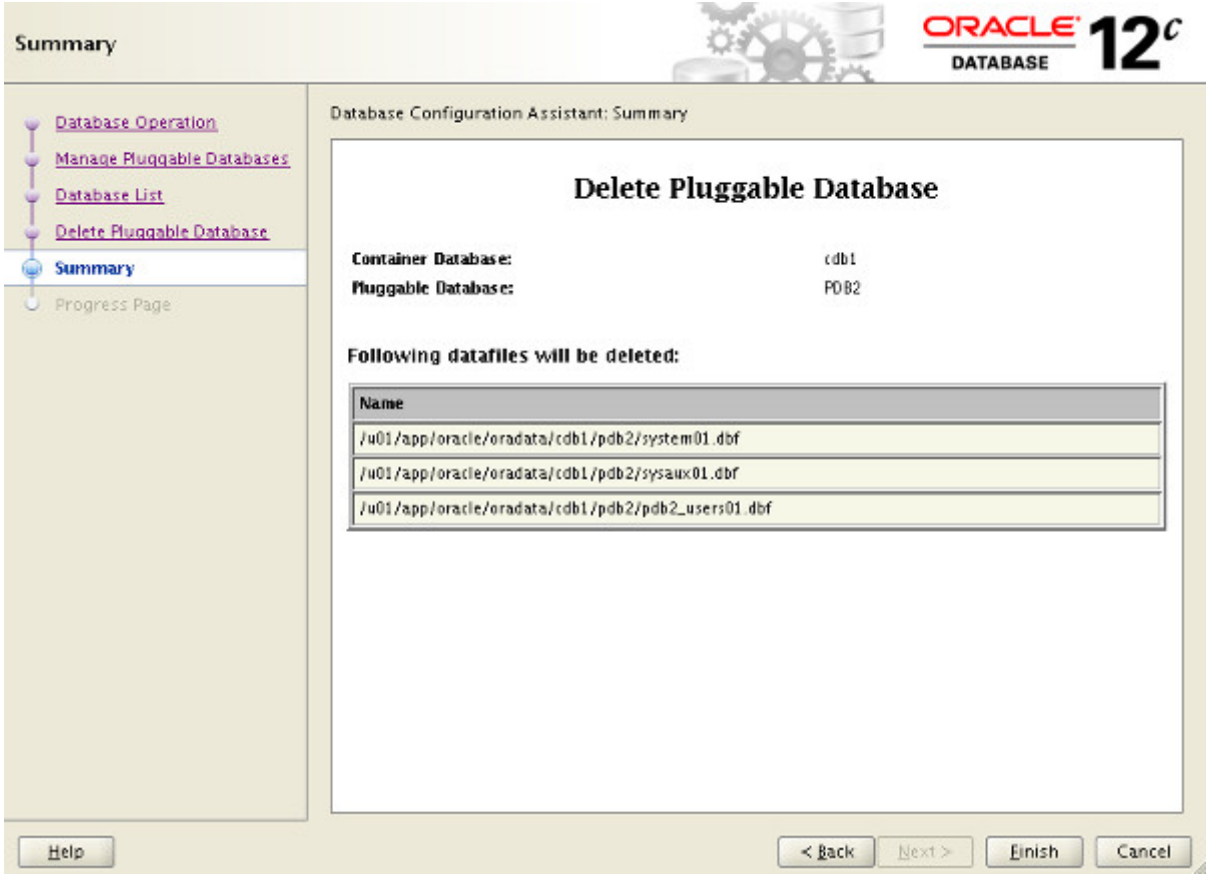


Select the PDB to delete and click the "Next" button.

Translate

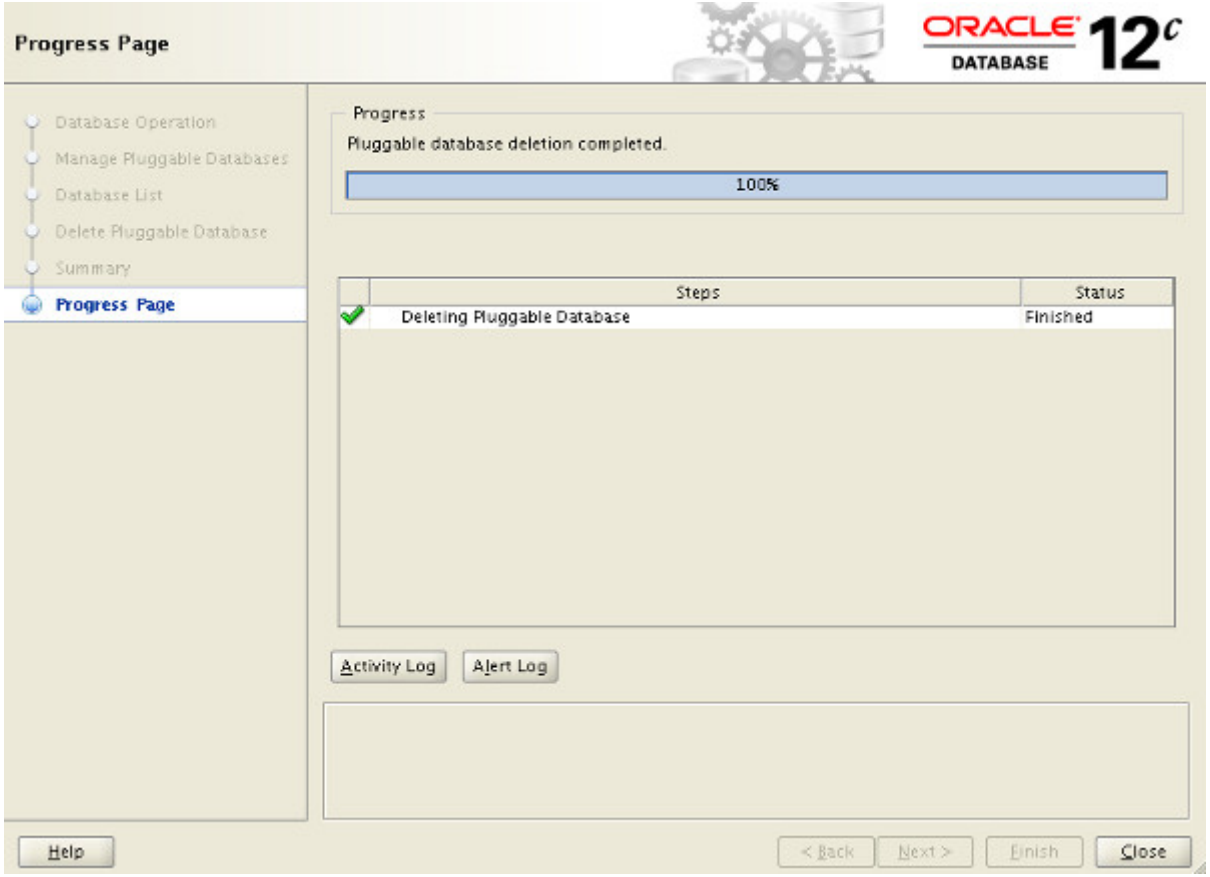
If you are happy with the summary information, click the "Finish" button.

Translate



Wait while the pluggable database is deleted. Once complete, click the "OK" button on the message dialog and the "Close" button on the main screen.

Translate

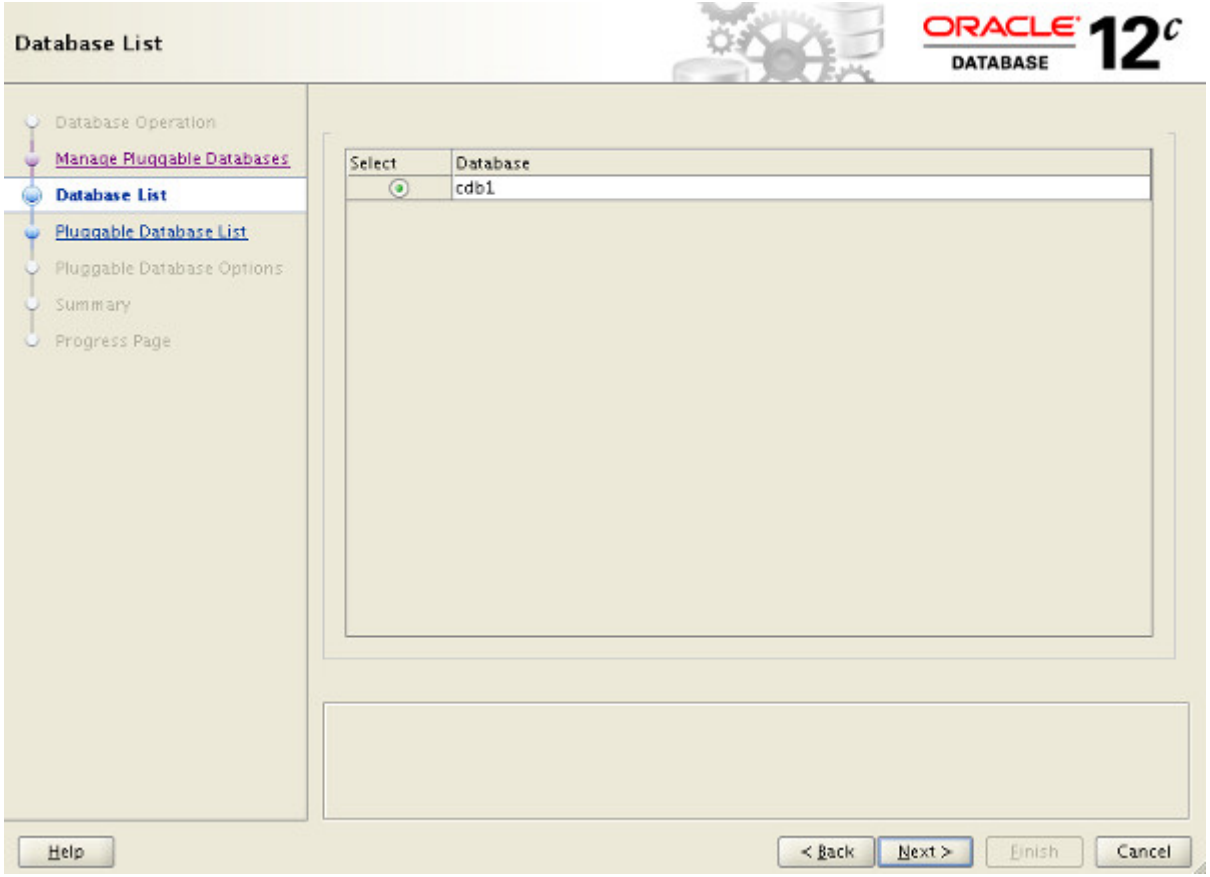


The pluggable database has been deleted from the container database.

Configure a Pluggable Database (PDB) using the DBCA

On the "Manage Pluggable Databases" screen shown previously, select the "Configure a Pluggable Database" option and click the "Next" button. On the resulting screen, select the container database that houses the pluggable database to be configured and click the "Next" button.

Translate

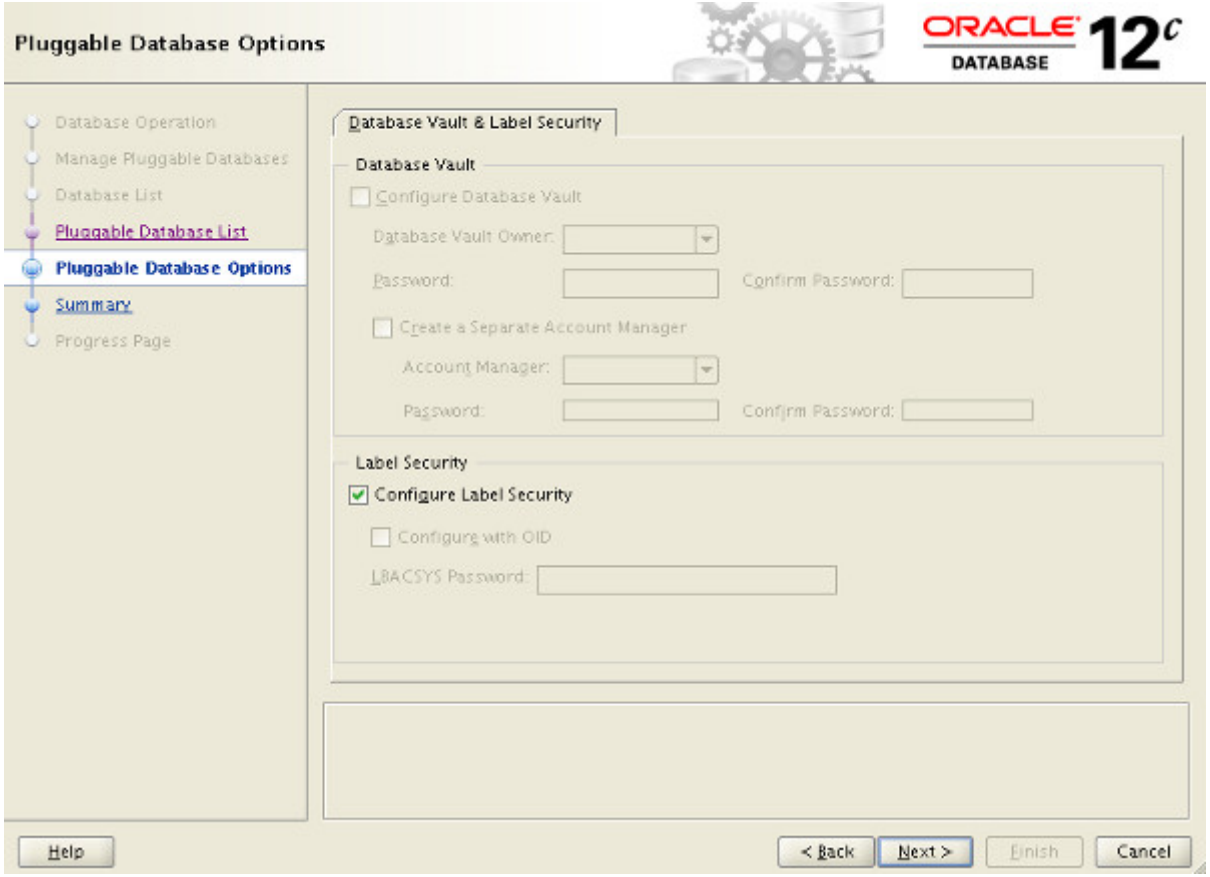


Select the PDB to configure and click the "Next" button.

Translate

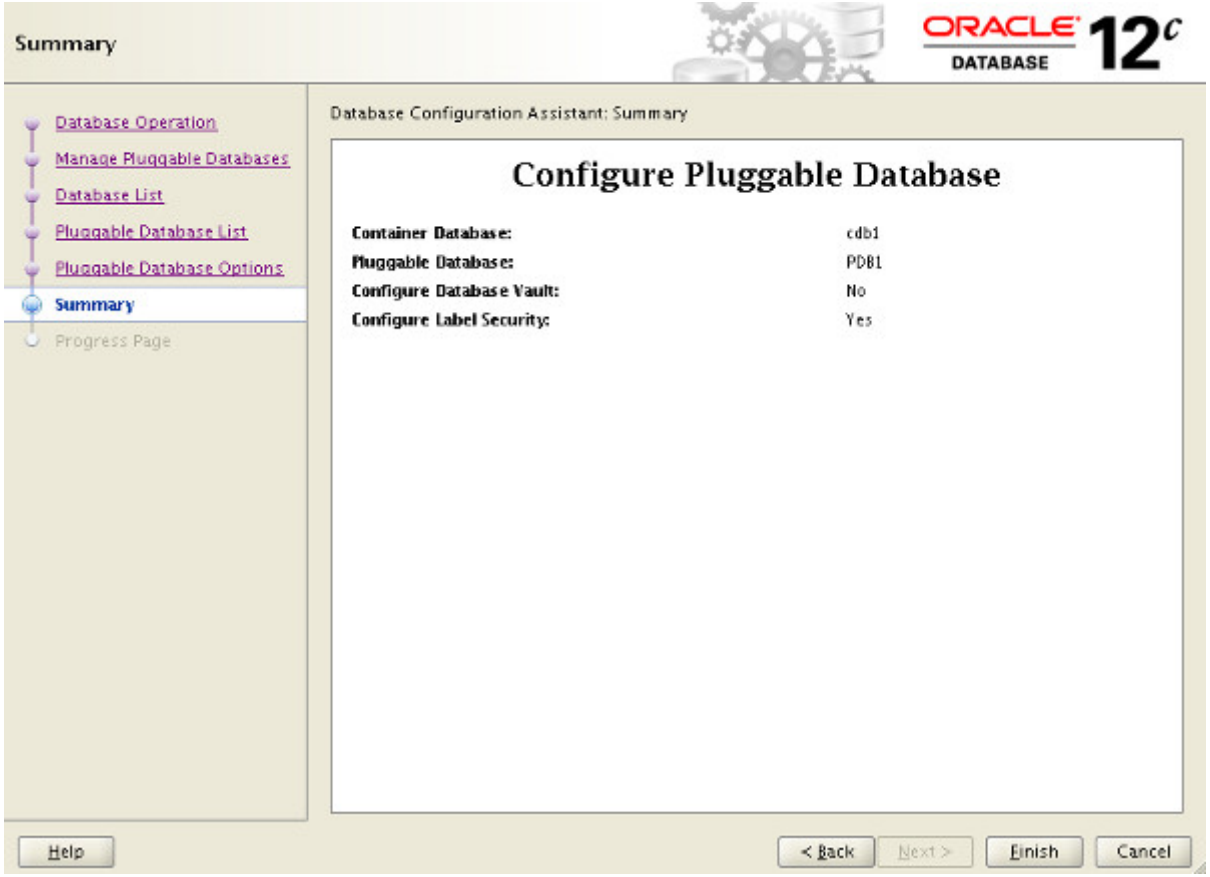
Select any additional options you would like to configure, then click the "Next" button.

Translate



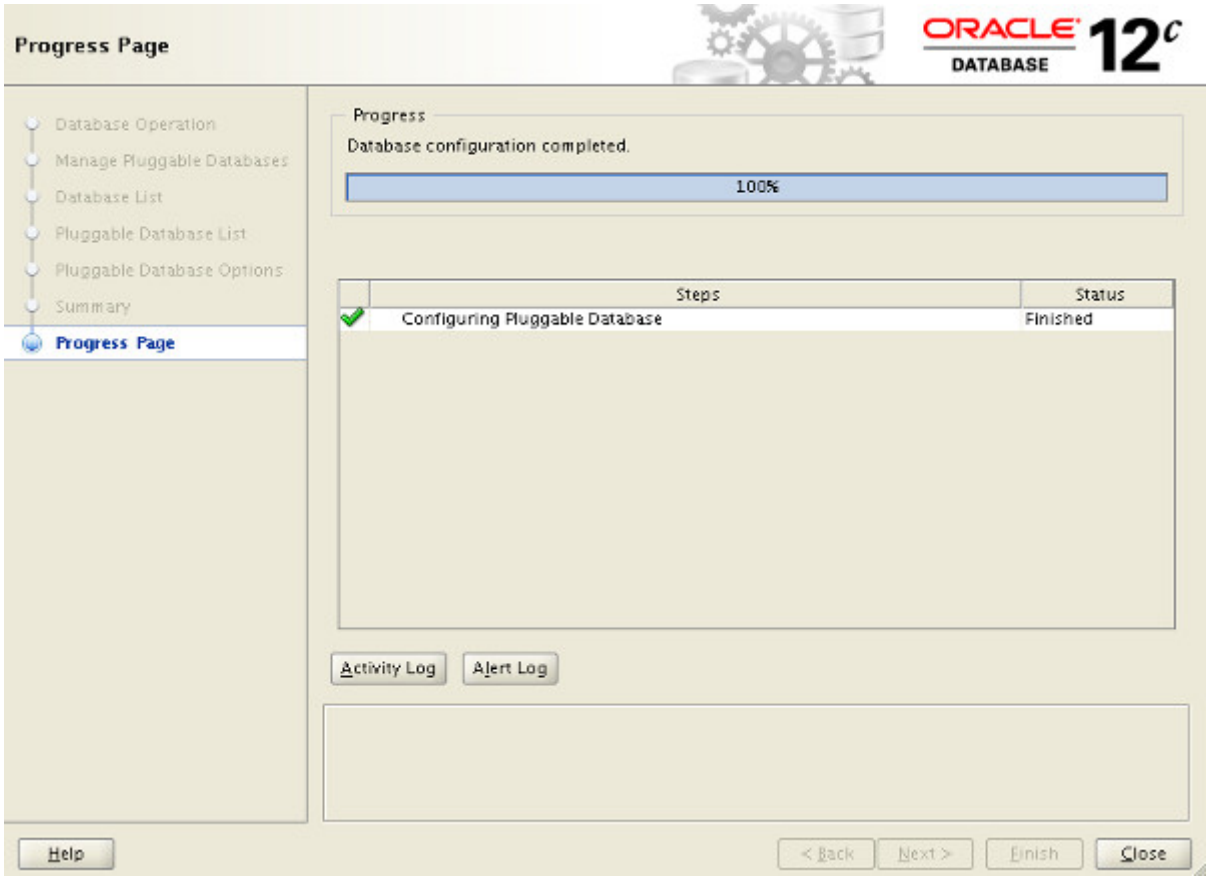
If you are happy with the summary information, click the "Finish" button.

Translate



Wait while the pluggable database is configured. Once complete, click the "OK" button on the message dialog and the "Close" button on the main screen.

Translate



The pluggable database has been configured.

Manual (SQL*Plus)

There are lots of variations on the CREATE PLUGGABLE DATABASE (http://docs.oracle.com/cd/E16655_01/server.121/e17209/statements_6009.htm) and ALTER PLUGGABLE DATABASE (http://docs.oracle.com/cd/E16655_01/server.121/e17209/statements_2007.htm) commands, so we will keep things simp only focus on those that mimic what is possible in the DBCA.

For all the operations listed here you must be connected to the CDB with the container set to root (the default). Typically you will be connected to a common user with SYSDBA or SYSOPER privilege. When creating a new pluggable database, the user must have the `CREATE PLUGGABLE DATABASE` system privilege.

Create a Pluggable Database (PDB) Manually

To create a new pluggable database from the seed database, all we have to do is tell Oracle where the file should be placed. We can do this using one of two methods. The first method uses the `FILE_NAME_CONVERT` clause in the `CREATE PLUGGABLE DATABASE` statement.

```
CONN / AS SYSDBA

CREATE PLUGGABLE DATABASE pdb2 ADMIN USER pdb_adm IDENTIFIED BY Password1
  FILE_NAME_CONVERT=('/u01/app/oracle/oradata/cdb1/pdbseed/','/u01/app/oracle/oradata/cdb1/pdb2/');
```

Alternatively, we can specify the `PDB_FILE_NAME_CONVERT` initialization parameter before calling the command without using the `FILE_NAME_CONVERT` clause.

```
CONN / AS SYSDBA

ALTER SESSION SET PDB_FILE_NAME_CONVERT='/u01/app/oracle/oradata/cdb1/pdbseed/','/u01/app/oracle/oradata/cdb1/pdb

CREATE PLUGGABLE DATABASE pdb3 ADMIN USER pdb_adm IDENTIFIED BY Password1;
```

Every time there is a need to convert file locations, either of these two methods will work. For the remainder of the article I will stick to using the `FILE_NAME_CONVERT` method to cut down on the variations I have to display.

We can see the PDBs are present by querying the `DBA_PDBS` and `V$PDBS` views.

Translate

```
COLUMN pdb_name FORMAT A20

SELECT pdb_name, status
FROM   dba_pdbs
ORDER BY pdb_name;

PDB_NAME             STATUS
-------------------- -------------
PDB$SEED             NORMAL
PDB1                 NORMAL
PDB2                 NEW
PDB3                 NEW

SQL>

SELECT name, open_mode
FROM   v$pdbs
ORDER BY name;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB1                           MOUNTED
PDB2                           MOUNTED
PDB3                           MOUNTED

SQL>
```

Translate

The PDBs are created with the status of 'NEW'. They must be opened in `READ WRITE` mode at least once for the integration of the PDB into the CDB to be complete.

```
ALTER PLUGGABLE DATABASE pdb2 OPEN READ WRITE;
ALTER PLUGGABLE DATABASE pdb3 OPEN READ WRITE;

SELECT pdb_name, status
FROM   dba_pdbs
ORDER BY pdb_name;

PDB_NAME             STATUS
-------------------- -------------
PDB$SEED             NORMAL
PDB1                 NORMAL
PDB2                 NORMAL
PDB3                 NORMAL

SQL>

SELECT name, open_mode
FROM   v$pdbs
ORDER BY name;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB1                           MOUNTED
PDB2                           READ WRITE
PDB3                           READ WRITE

SQL>
```

Translate

Depending on the syntax used, you may need to grant the PDB_DBA role to the local admin users for the PDB.

Unplug a Pluggable Database (PDB) Manually

Before attempting to unplug a PDB, you must make sure it is closed. To unplug the database use the `ALTER PLUGGABLE DATABASE` command with the `UNPLUG INTO` clause to specify the location of the XML metadata file.

```
ALTER PLUGGABLE DATABASE pdb2 CLOSE;
ALTER PLUGGABLE DATABASE pdb2 UNPLUG INTO '/u01/app/oracle/oradata/cdb1/pdb2/pdb2.xml';
```

The pluggable database is still present, but you shouldn't open it until the metadata file and all the datafiles are copied somewhere safe.

```
SELECT name, open_mode
FROM   v$pdbs
ORDER BY name;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB1                           MOUNTED
PDB2                           MOUNTED
PDB3                           READ WRITE

SQL>
```

You can delete the PDB, choosing to keep the files on the file system.

Translate

```
DROP PLUGGABLE DATABASE pdb2 KEEP DATAFILES;

SELECT name, open_mode
FROM   v$pdbs
ORDER BY name;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB1                           MOUNTED
PDB3                           READ WRITE

SQL>
```

Plugin a Pluggable Database (PDB) Manually

Plugging in a PDB into the CDB is similar to creating a new PDB. First check the PBD is compatible with the CDB by calling the `DBMS_PDB.CHECK_PLUG_COMPATIBILITY` function, passing in the XML metadata file and the name of the PDB you want to create using it.

Translate

```
SET SERVEROUTPUT ON
DECLARE
  l_result BOOLEAN;
BEGIN
  l_result := DBMS_PDB.check_plug_compatibility(
               pdb_descr_file => '/u01/app/oracle/oradata/cdb1/pdb2/pdb2.xml',
               pdb_name       => 'pdb2');

  IF l_result THEN
    DBMS_OUTPUT.PUT_LINE('compatible');
  ELSE
    DBMS_OUTPUT.PUT_LINE('incompatible');
  END IF;
END;
/
compatible

PL/SQL procedure successfully completed.

SQL>
```

If the PDB is not compatible, violations are listed in the `PDB_PLUG_IN_VIOLATIONS` view. If the PDB is compatible, create a new PDB using it as the source. If we were creating it with a new name we might do something like this.

```
CREATE PLUGGABLE DATABASE pdb5 USING '/u01/app/oracle/oradata/cdb1/pdb2/pdb2.xml'
  FILE_NAME_CONVERT=('/u01/app/oracle/oradata/cdb1/pdb2/','/u01/app/oracle/oradata/cdb1/pdb5/');
```

Translate

Instead, we want to plug the database back into the same container, so we don't need to copy the files or recreate the temp file, so we can do the following.

```
CREATE PLUGGABLE DATABASE pdb2 USING '/u01/app/oracle/oradata/cdb1/pdb2/pdb2.xml'
  NOCOPY
  TEMPFILE REUSE;

ALTER PLUGGABLE DATABASE pdb2 OPEN READ WRITE;

SELECT name, open_mode
FROM   v$pdbs
ORDER BY name;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB1                           MOUNTED
PDB2                           READ WRITE
PDB3                           READ WRITE

SQL>
```

Clone a Pluggable Database (PDB) Manually

Cloning an existing local PDB is similar to creating a new PDB from the seed PDB, except now we are using non-seed PDB as the source, which we have to identify using the `FROM` clause. Make sure the source PDB is open in `READ ONLY` mode.

Translate

```
ALTER PLUGGABLE DATABASE pdb3 CLOSE;
ALTER PLUGGABLE DATABASE pdb3 OPEN READ ONLY;

CREATE PLUGGABLE DATABASE pdb4 FROM pdb3
  FILE_NAME_CONVERT=('/u01/app/oracle/oradata/cdb1/pdb3/','/u01/app/oracle/oradata/cdb1/pdb4/');

ALTER PLUGGABLE DATABASE pdb4 OPEN READ WRITE;

-- Switch the source PDB back to read/write
ALTER PLUGGABLE DATABASE pdb3 CLOSE;
ALTER PLUGGABLE DATABASE pdb3 OPEN READ WRITE;
```

The cloning syntax also allows for cloning from remote databases using a database link in the local CBD. There are a few restriction associated with this functionality.

- The database link can point directly to the remote PDB or to a common user in the remote CBD that owns the remote PDB.
- If it points to a common user in the remote CBD that owns the remote PDB, that user must have the `CREATE PLUGGABLE DATABASE` system privilege.
- The source and target CDBs must have the same endians.
- The source and target CDBs must have the same options installed.
- The source and target CDBs must have the same character set and national character set.

Assuming the remote PDB was in READ ONLY mode, the following command should perform the required operation.

```
CREATE PLUGGABLE DATABASE pdb5 FROM remote_pdb5@remotecdb1
  FILE_NAME_CONVERT=('/u01/app/oracle/oradata/cdb1/remote_pdb5/','/u01/app/oracle/oradata/cdb1/pdb5/');

ALTER PLUGGABLE DATABASE pdb4 OPEN READ WRITE;
```

Translate

This functionality does not work properly in the 12.1.0.1 release of the database, but it has been fixed in 12.1.0.2. You can see an article specifically on this subject here (multitenant-clone-remote-pdb-or-non-cdb-12cr1).

Clone a Pluggable Database (PDB) Manually (Metadata Only : NO DATA)

The 12.1.0.2 patchset introduced the ability to do a metadata-only clone. Adding the `NO DATA` clause when cloning a PDB signifies that only the metadata for the user-created objects should be cloned, not the data in the tables and indexes. You can read more about this feature in the following article.

- Multitenant : Metadata Only PDB Clones in Oracle Database 12c Release 1 (12.1.0.2) (multitenant-metadata-only-pdb-clones-12cr1)

Delete a Pluggable Database (PDB) Manually

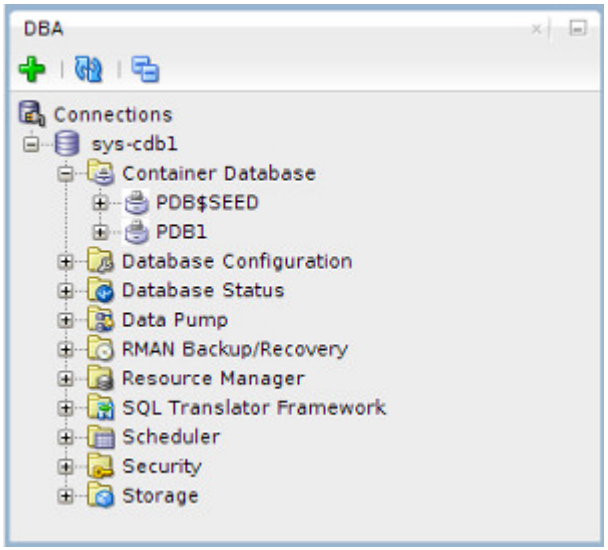
When dropping a pluggable database, you must decide whether to keep or drop the associated datafiles. The PDBs must be closed before being dropped.

```
ALTER PLUGGABLE DATABASE pdb2 CLOSE;
DROP PLUGGABLE DATABASE pdb2 KEEP DATAFILES;

ALTER PLUGGABLE DATABASE pdb3 CLOSE;
DROP PLUGGABLE DATABASE pdb3 INCLUDING DATAFILES;

ALTER PLUGGABLE DATABASE pdb4 CLOSE;
DROP PLUGGABLE DATABASE pdb4 INCLUDING DATAFILES;

SELECT name, open_mode
FROM   v$pdbs
ORDER BY name;

NAME                           OPEN_MODE
------------------------------ ----------
PDB$SEED                       READ ONLY
PDB1                           MOUNTED

SQL>
```

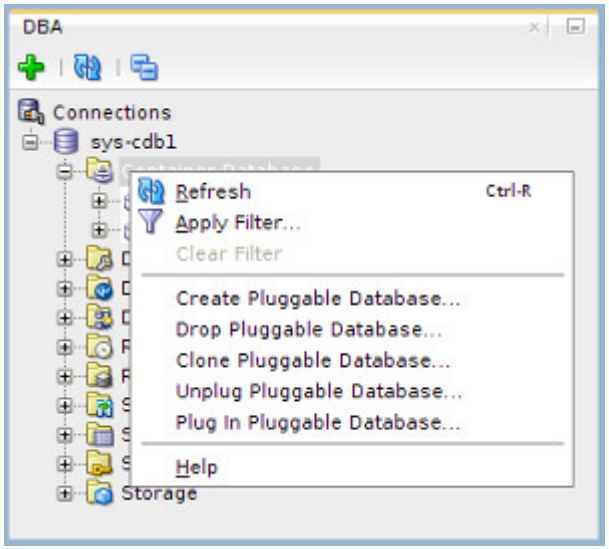
Translate

SQL Developer

The DBA section of SQL Developer includes tree node called "Container Database".

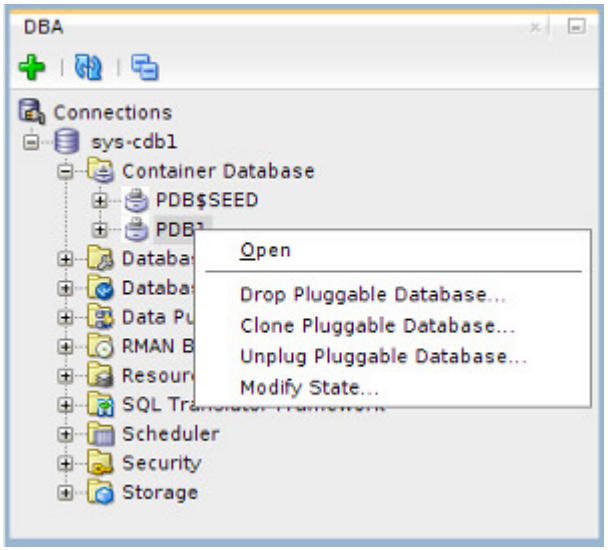


Right-clicking on the "Container Database" node produces a popup menu showing you what operations are available.



Translate

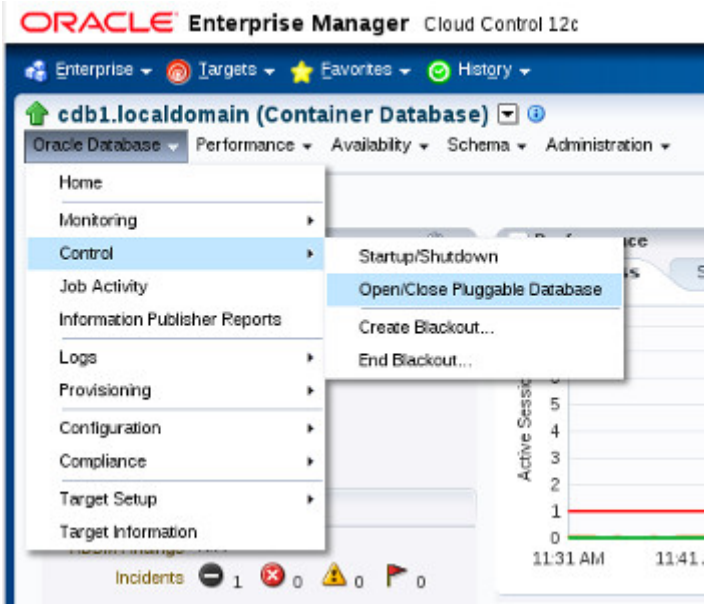
Right-clicking on a specific PDB node produces a popup menu showing only those operations that are relevant to that PDB.



If you understand the DBCA and SQL*Plus approach to managing PDBs, these SQL Developer screens are very straight forward.

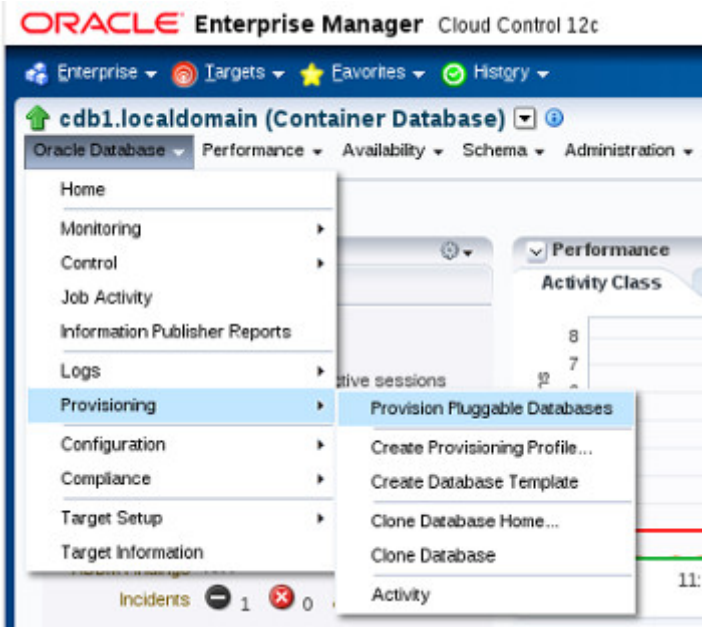
Cloud Control

Cloud Control 12cR3 onward supports pluggable database functionality. Once you click on the container database, the "Oracle Database > Control > Open/Close Pluggable Database" menu option allows you to control the state of the PDBs owned by the CDB.



Translate

The "Oracle Database > Provision > Provision Pluggable Database" menu option allows you to perform other operations PDBs owned by the CDB, including cloning, unplugging amongst other things.



As with SQL Developer, if you understand how the pluggable database functionality works, the Cloud Control screens are self explanatory.

For more information see:

- Introduction to the Multitenant Architecture (http://docs.oracle.com/database/121/CNCPT/cdbovrvw.htm)
- Overview of the Multitenant Architecture (http://docs.oracle.com/database/121/CNCPT/cdblogic.htm)
- Managing a Multitenant Environment (http://docs.oracle.com/database/121/ADMIN/part_cdb.htm)
- CREATE PLUGGABLE DATABASE (http://docs.oracle.com/database/121/SQLRF/statements_6010.htm)
- ALTER PLUGGABLE DATABASE (http://docs.oracle.com/database/121/SQLRF/statements_2007.htm)
- DBMS_PDB (http://docs.oracle.com/database/121/ARPLS/d_pdb.htm)
- Oracle Enterprise Manager Cloud Control 12c Release 3 Installation on Oracle Linux 5.9 and 6.4 (cloud-control-12cr3-installation-on-oracle-linux-5-and-6)
- Multitenant : Create a Pluggable Database (https://www.youtube.com/watch?v=dPHerZHvUyk)
- Multitenant : Unplug and Plugin a Pluggable Database (https://www.youtube.com/watch?v=yMBcKhWFMwE)
- Multitenant : Migrate a Non-Container Database (CDB) to a Pluggable Database (PDB) in Oracle Database 12c Release 1 (12.1) (multitenant-migrate-non-cdb-to-pdb-12cr1)

- Multitenant : Clone a Remote PDB or Non-CDB in Oracle Database 12c (12.1.0.2) (multitenant-clone-remote-pdb-or-non-cdb-12cr1)
- Multitenant : Configure Instance Parameters and Modify Container Databases (CDB) and Pluggable Databases (PDB) in Oracle Database 12c Release 1 (12.1) (multitenant-configure-instance-parameters-of-cdb-and-pdb-12cr1)
- Multitenant : Metadata Only PDB Clones in Oracle Database 12c Release 1 (12.1.0.2) (multitenant-metadata-only-pdb-clones-12cr1)
- Multitenant : PDB Subset Cloning in Oracle Database 12c Release 1 (12.1.0.2) (multitenant-pdb-subset-cloning-12cr1)

Hope this helps. Regards Tim...

Back to the Top.