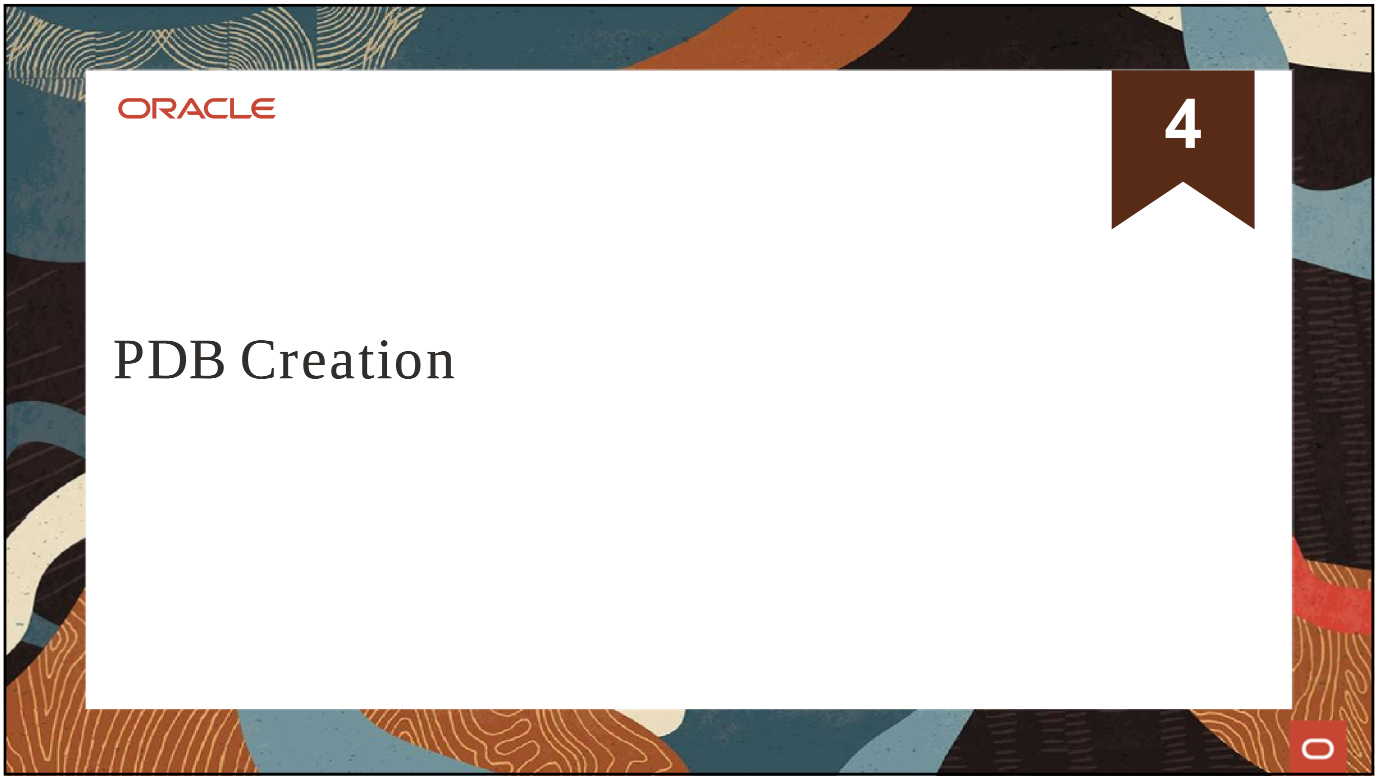
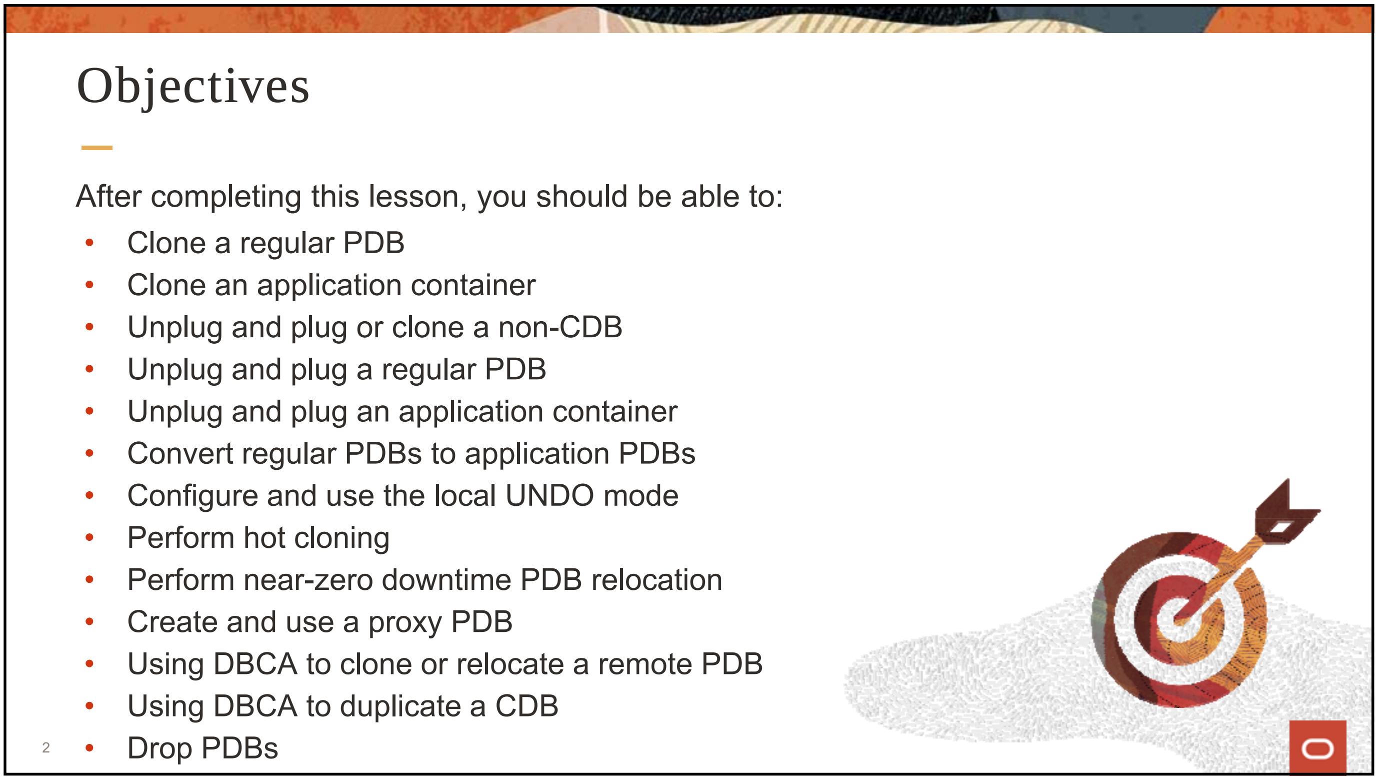


ORACLE

4

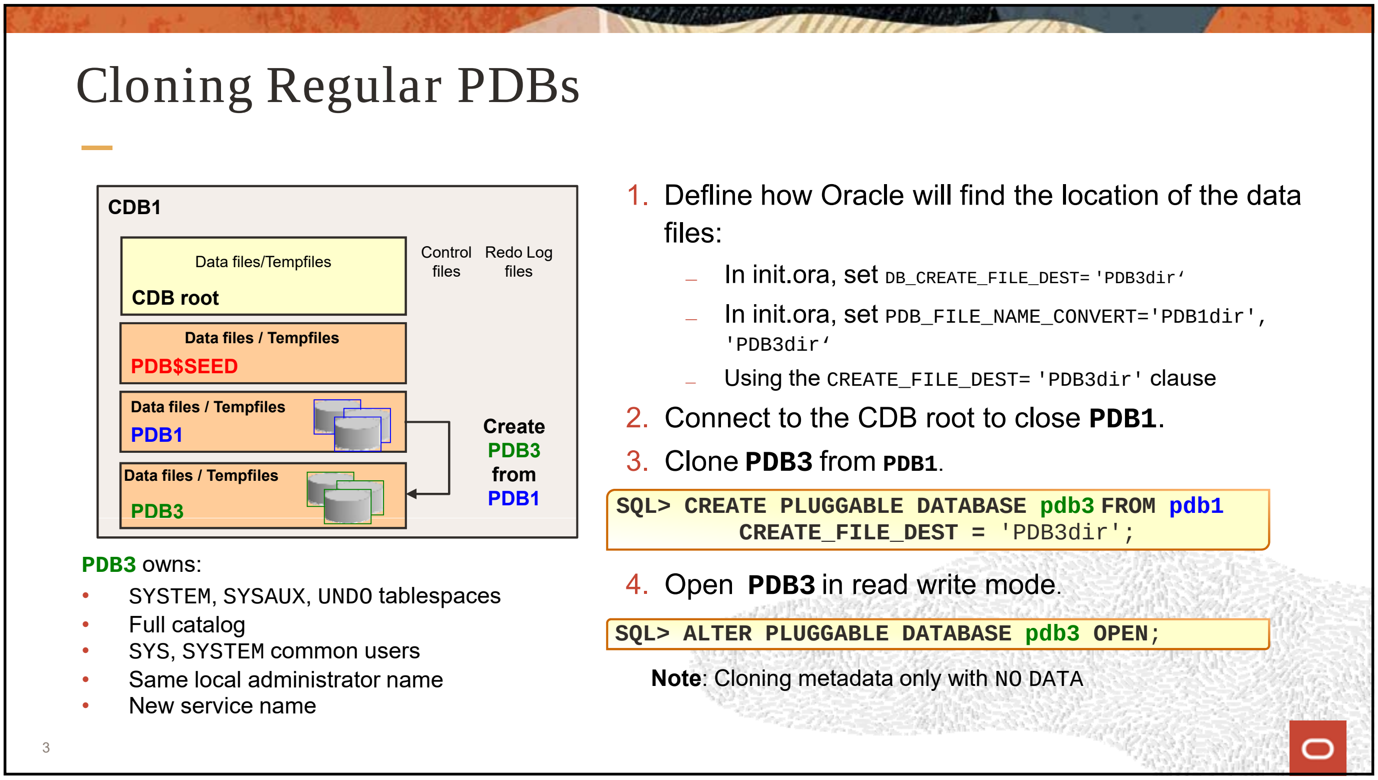
PDB Creation



Objectives

After completing this lesson, you should be able to:

- Clone a regular PDB
- Clone an application container
- Unplug and plug or clone a non-CDB
- Unplug and plug a regular PDB
- Unplug and plug an application container
- Convert regular PDBs to application PDBs
- Configure and use the local UNDO mode
- Perform hot cloning
- Perform near-zero downtime PDB relocation
- Create and use a proxy PDB
- Using DBCA to clone or relocate a remote PDB
- Using DBCA to duplicate a CDB
- Drop PDBs



Cloning Regular PDBs

CDB1

Data files/Tempfiles
CDB root

Control files

Redo Log files

Data files / Tempfiles
PDB$SEED

Data files / Tempfiles
PDB1

Data files / Tempfiles
PDB3

Create PDB3 from PDB1

PDB3 owns:

- SYSTEM, SYSAUX, UNDO tablespaces
- Full catalog
- SYS, SYSTEM common users
- Same local administrator name
- New service name

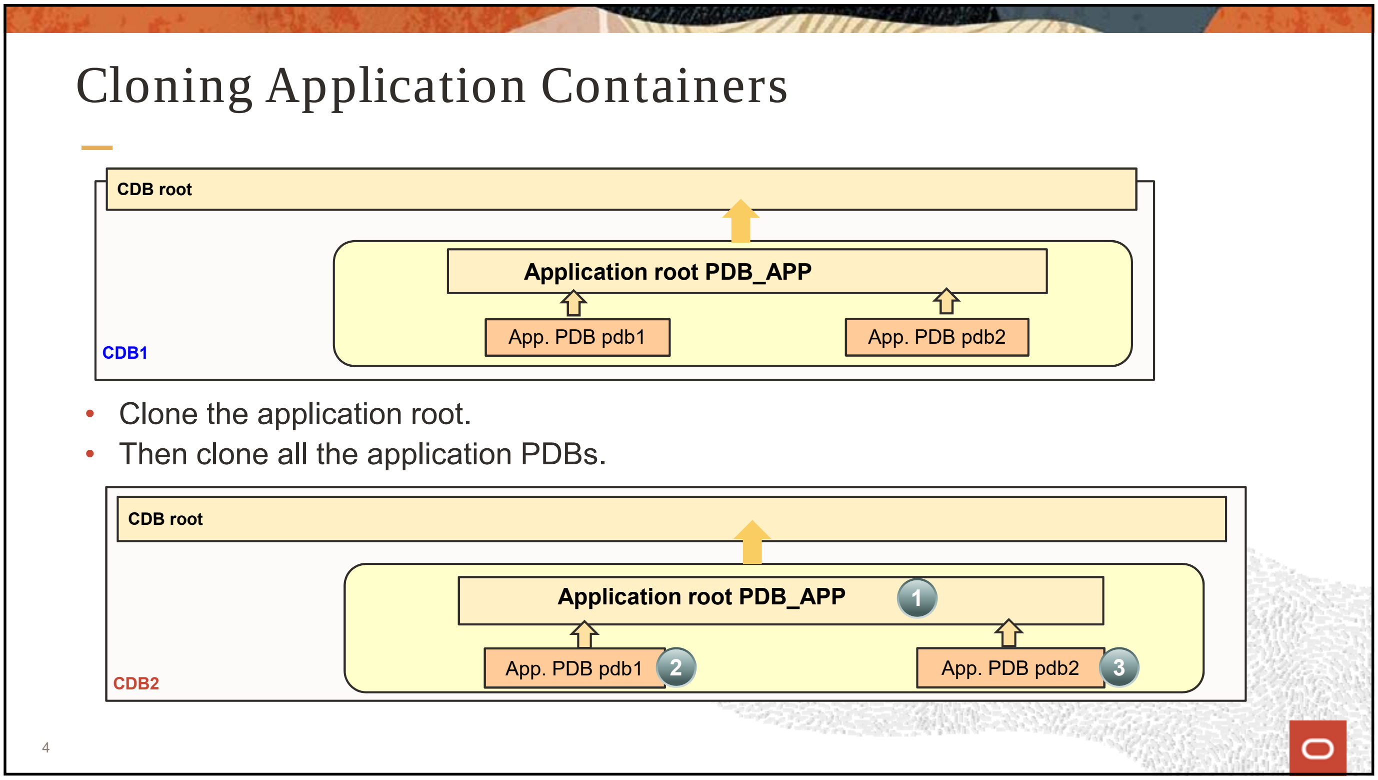
1. Defline how Oracle will find the location of the data files:
   - In init.ora, set `DB_CREATE_FILE_DEST= 'PDB3dir‘`
   - In init.ora, set `PDB_FILE_NAME_CONVERT='PDB1dir', 'PDB3dir‘`
   - Using the `CREATE_FILE_DEST= 'PDB3dir'` clause
2. Connect to the CDB root to close **PDB1**.
3. Clone **PDB3** from **PDB1**.

```
SQL> CREATE PLUGGABLE DATABASE pdb3 FROM pdb1
          CREATE_FILE_DEST = 'PDB3dir';
```

4. Open **PDB3** in read write mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb3 OPEN;
```

Note: Cloning metadata only with NO DATA



Cloning Application Containers

CDB root

Application root PDB_APP

App. PDB pdb1

App. PDB pdb2

CDB1

- Clone the application root.
- Then clone all the application PDBs.

CDB root

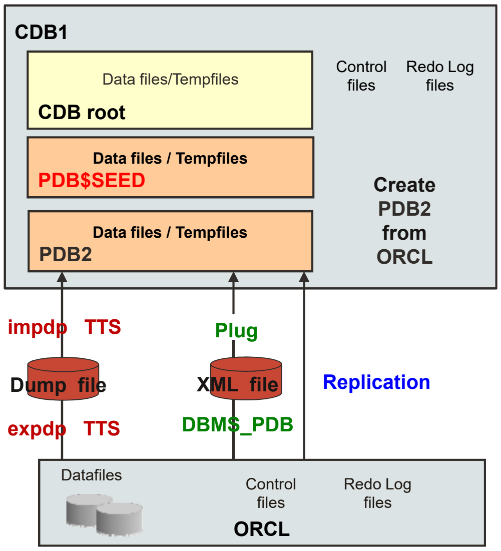
Application root PDB_APP 1

App. PDB pdb1 2

App. PDB pdb2 3

CDB2

Plugging a Non-CDB into CDB



CDB1

Data files/Tempfiles

CDB root

Control files

Redo Log files

Data files / Tempfiles

PDB$SEED

Data files / Tempfiles

PDB2

Create PDB2 from ORCL

impdp TTS

Dump file

expdp TTS

Plug

XML file

DBMS_PDB

Replication

Datafiles

Control files

Redo Log files

ORCL

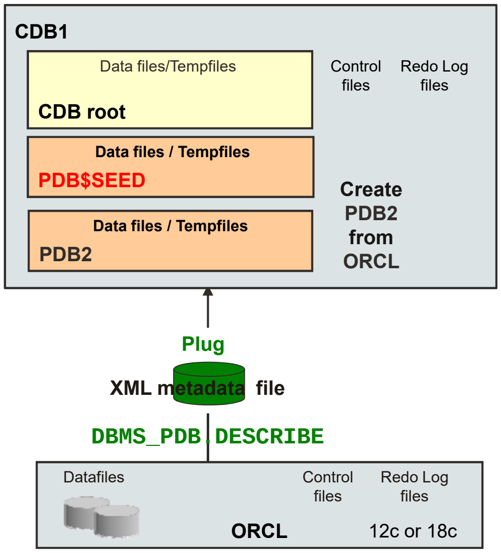
Possible methods:

- Data Pump (TTS or TDB or full export/import)
- Plugging (XML file definition with `DBMS_PDB`)
- Cloning
- Replication

Entities are created in the new PDB:

- Tablespaces: `SYSTEM`, `SYSAUX`, `UNDO`
- A full catalog
- Common users: `SYS`, `SYSTEM`
- A local administrator (PDBA)
- A new default service

Plugging a Non-CDB into CDB as PDB Using DBMS_PDB



CDB1

Data files/Tempfiles

CDB root

Control files

Redo Log files

Data files / Tempfiles

PDB$SEED

Data files / Tempfiles

PDB2

Create PDB2 from ORCL

Plug

XML metadata file

DBMS_PDB.DESCRIBE

Datafiles

Control files

Redo Log files

ORCL

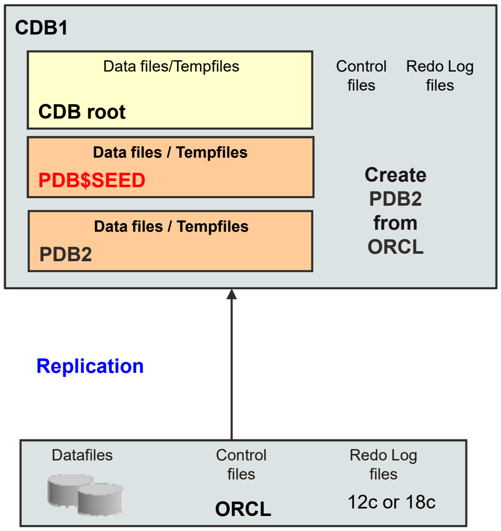
12c or 18c

1. Open **ORCL** in READ ONLY mode.

2.
   ```
   SQL> EXEC DBMS_PDB.DESCRIBE ('/tmp/ORCL.xml')
   ```
3. Connect to the target CDB root as a common user with `CREATE PLUGGABLE DATABASE` privilege.
4. Plug in the unplugged **ORCL** as **PDB2.**
   ```
   SQL> CREATE PLUGGABLE DATABASE PDB2
                 USING '/tmp/ORCL.xml';
   ```
5. Run the `noncdb_to_pdb.sql` script in **PDB2**.
   ```
   SQL> CONNECT sys@PDB2 AS SYSDBA
   SQL> @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb
   ```
6. Open `PDB2`.

Note: The `STATUS` of the PDB is `CONVERTING`.

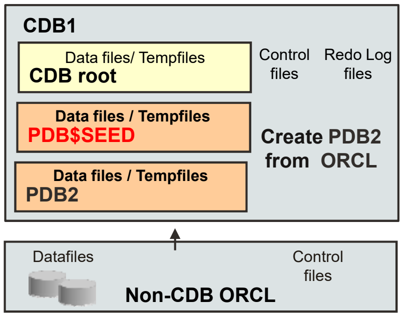
Replicating Non-CDB into CDB



1. Connect to the CDB root as a common user with `CREATE PLUGGABLE DATABASE` privilege.
2. Create new **`PDB2`** (from PDB$SEED).
3. Open **`PDB2`** in read-write mode.
4. Configure unidirectional replication environment from **`ORCL`** to **`PDB2`.**
5. Check application data.

```
SQL> CONNECT sys@PDB2
SQL> SELECT * FROM dba_tables;
SQL> SELECT * FROM HR.EMP;
```

Cloning a Non-CDB or Remote PDB



PDB_ORCL owns:

- SYSTEM, SYSAUX, UNDO tablespaces
- Full catalog
- A temporary tablespace
- SYS, SYSTEM common users
- New service name

1. Set **`ORCL`** in READ ONLY mode.
2. Connect to the CDB to create the database link:

```
SQL> CREATE DATABASE LINK link_orcl
            CONNECT TO system IDENTIFIED BY ***
            USING 'orcl';
```

3. Clone the non-CDB:

```
SQL> CREATE PLUGGABLE DATABASE pdb_orcl
            FROM NON$CDB@link_orcl
            CREATE_FILE_DEST = '…/PDB_orcl';
```

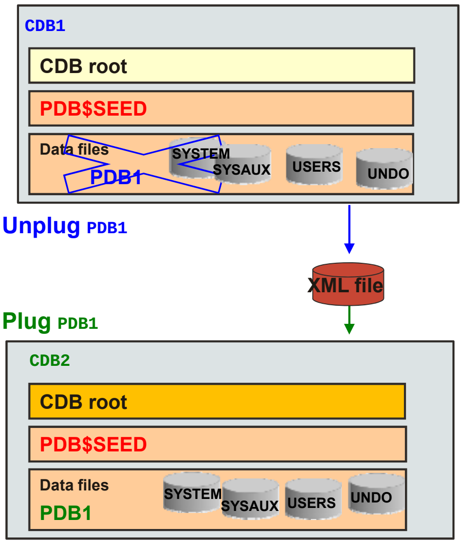
4. Run the `noncdb_to_pdb.sql` script.

```
SQL> CONNECT sys@pdb_orcl AS SYSDBA
SQL> @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb
```

5. Open **`PDB_ORCL`** in read-write mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb_orcl OPEN;
```

Plugging an Unplugged Regular PDB into CDB



Unplug **PDB1** from **CDB1**:

1. Connect to **CDB1** as a common user.
2. Verify that **PDB1** is closed.

3.
   ```
   SQL> ALTER PLUGGABLE DATABASE pdb1
                  UNPLUG INTO 'xmlfile1';
   ```
4. Drop **PDB1** from **CDB1**

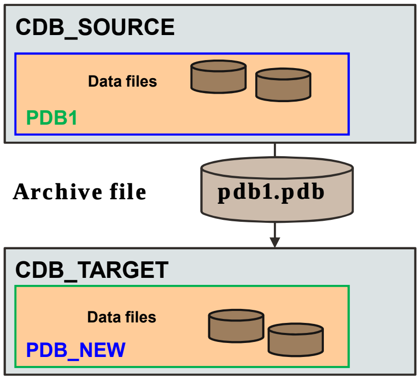
Plug **PDB1** into **CDB2** :

1. Connect to **CDB2** as a common user.
2. Use the `DBMS_PDB` package to check the compatibility of **PDB1** with **CDB2**.

3.
   ```
   SQL> CREATE PLUGGABLE DATABASE pdb1
                   USING 'xmlfile1' NOCOPY;
   ```
4. Open **PDB1** in read-write mode.

Plugging Using Archive File

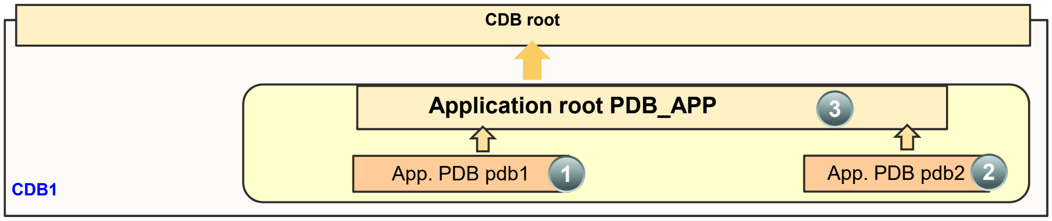
1. Unplugging a PDB into a single archive file includes:
   – XML file
   – Data files
2. Plugging the PDB requires only the archive file.



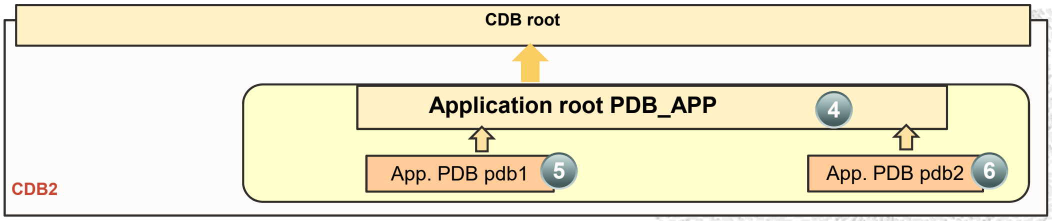
```
SQL> ALTER PLUGGABLE DATABASE pdb1
        UNPLUG INTO '/tmp/pdb1.pdb';
```

```
SQL> CREATE PLUGGABLE DATABASE pdb_new
        USING '/tmp/pdb1.pdb';
```

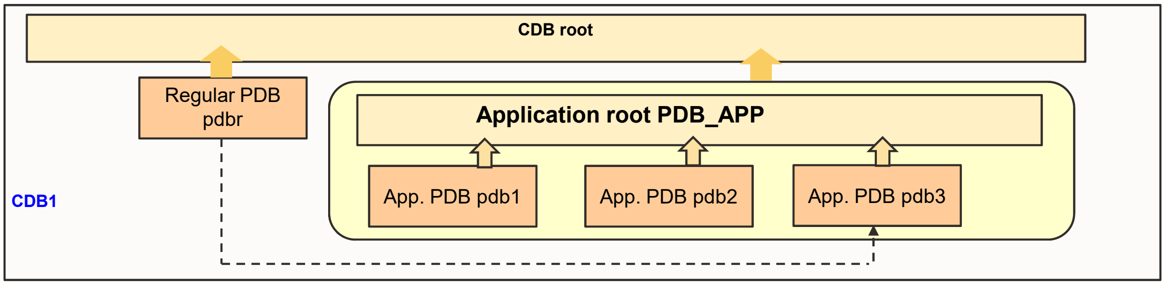
Unplugging and Plugging Application PDBs



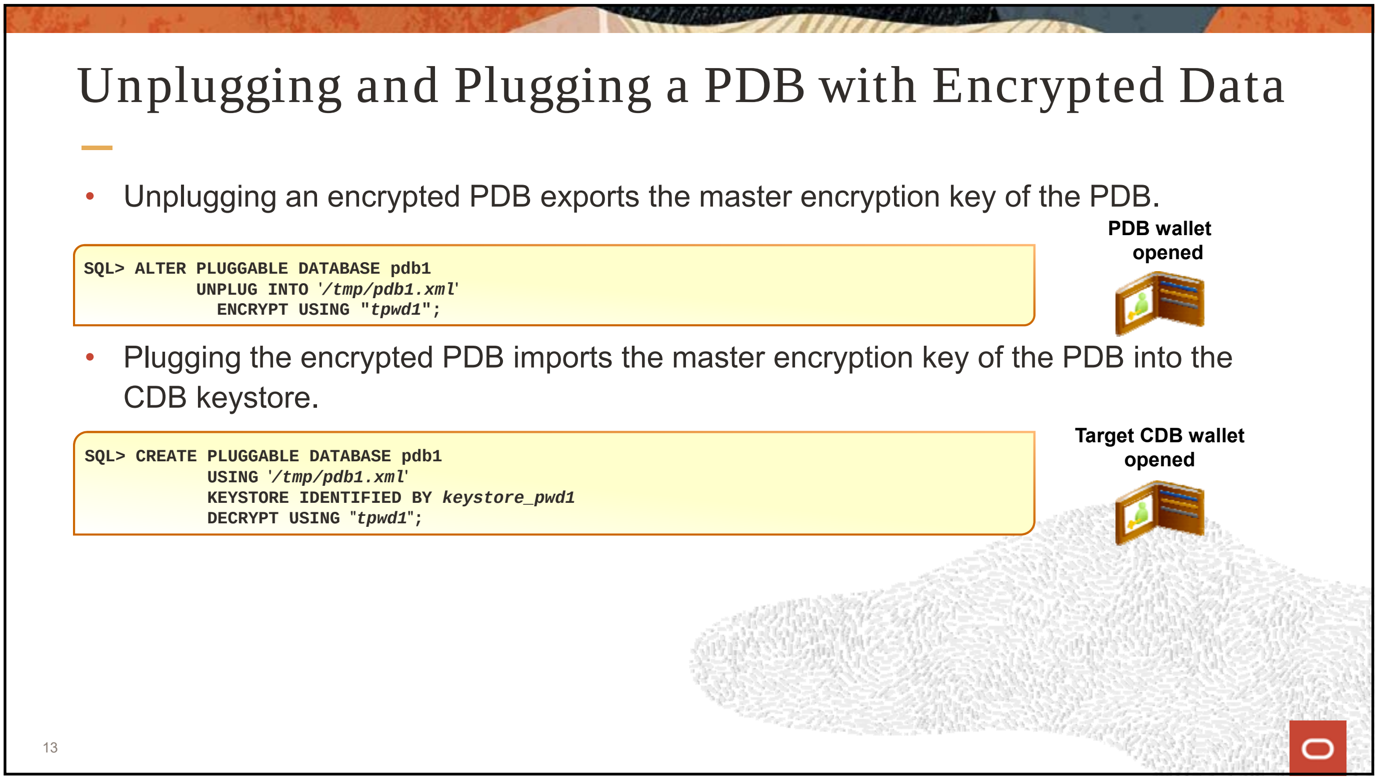
- In the source CDB, unplug all application PDBs and then the application root.
- In the target CDB, plug the application root first and then all the application PDBs.



Converting Regular PDBs to Application PDBs



- Two methods to convert the regular PDB to an application PDB:
  - Clone the regular PDB into an application root.
  - Unplug the regular PDB to plug it into an application root.
- Connect to the application PDB to execute the `pdb_to_apppdb.sql` script.
- Synchronize the application PDB with the application root.



Unplugging and Plugging a PDB with Encrypted Data

- Unplugging an encrypted PDB exports the master encryption key of the PDB.

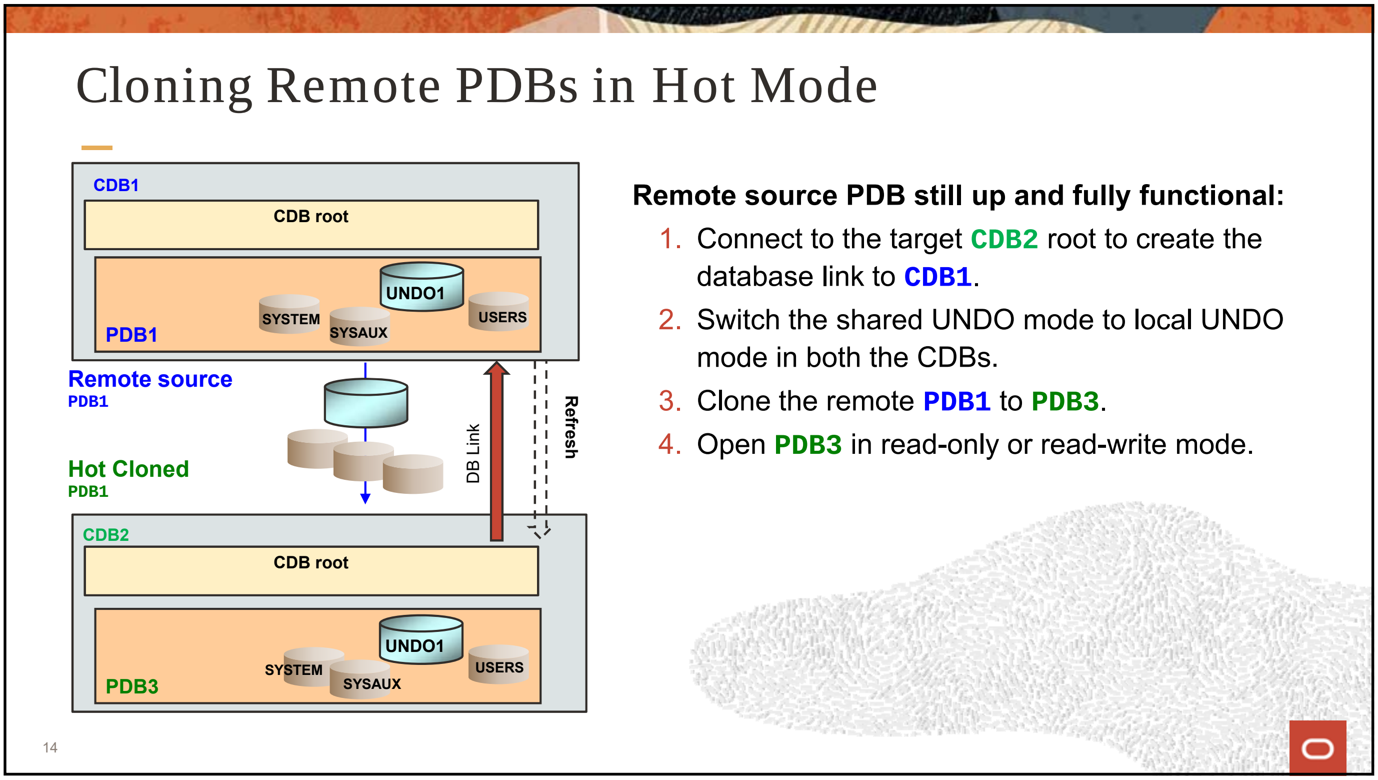
```
SQL> ALTER PLUGGABLE DATABASE pdb1
           UNPLUG INTO '/tmp/pdb1.xml'
             ENCRYPT USING "tpwd1";
```

PDB wallet
opened

- Plugging the encrypted PDB imports the master encryption key of the PDB into the CDB keystore.

```
SQL> CREATE PLUGGABLE DATABASE pdb1
            USING '/tmp/pdb1.xml'
            KEYSTORE IDENTIFIED BY keystore_pwd1
            DECRYPT USING "tpwd1";
```

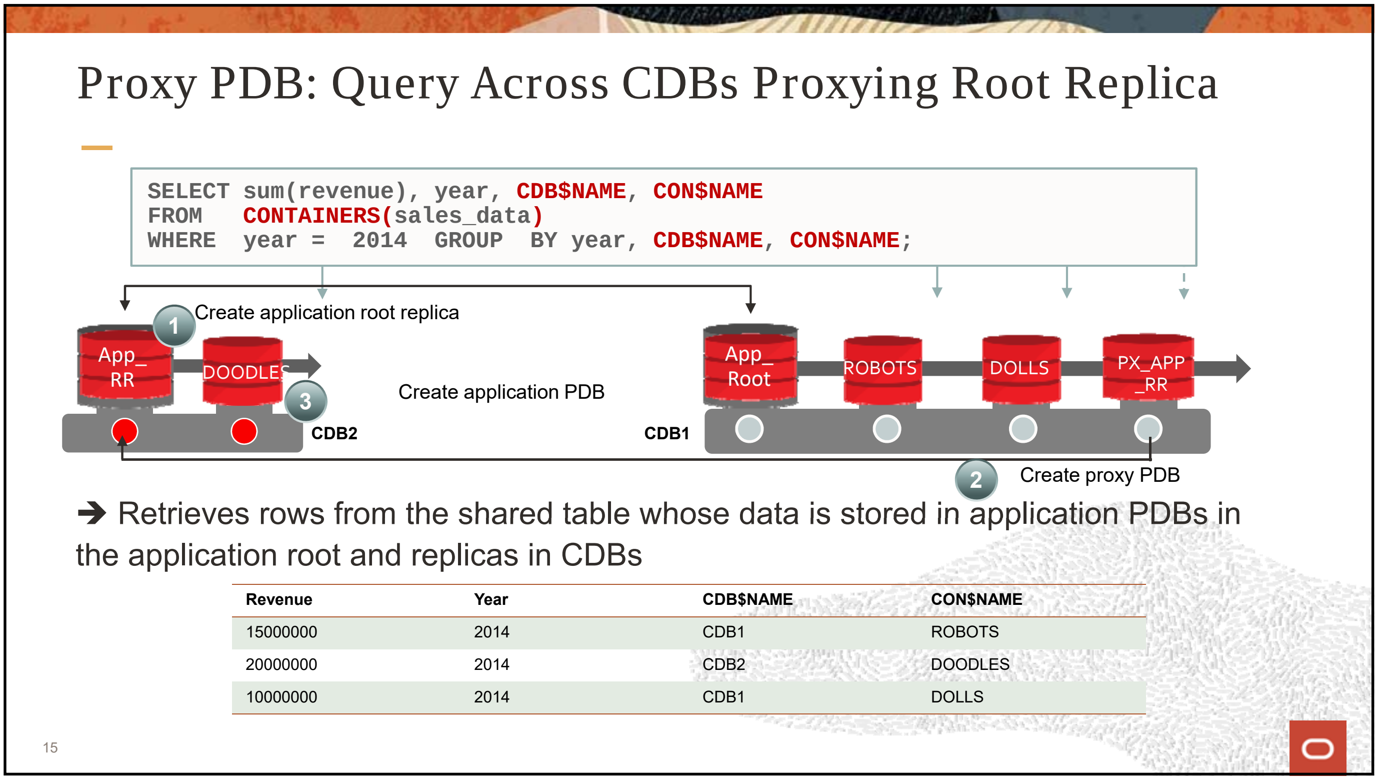
Target CDB wallet
opened



Cloning Remote PDBs in Hot Mode

Remote source PDB still up and fully functional:

1. Connect to the target **CDB2** root to create the database link to **CDB1**.
2. Switch the shared UNDO mode to local UNDO mode in both the CDBs.
3. Clone the remote **PDB1** to **PDB3**.
4. Open **PDB3** in read-only or read-write mode.



Proxy PDB: Query Across CDBs Proxying Root Replica

```
SELECT sum(revenue), year, CDB$NAME, CON$NAME
FROM   CONTAINERS(sales_data)
WHERE  year =  2014  GROUP  BY year, CDB$NAME, CON$NAME;
```

1 Create application root replica

App_ RR

DOODLES

3

Create application PDB

CDB2

App_ Root

ROBOTS

DOLLS

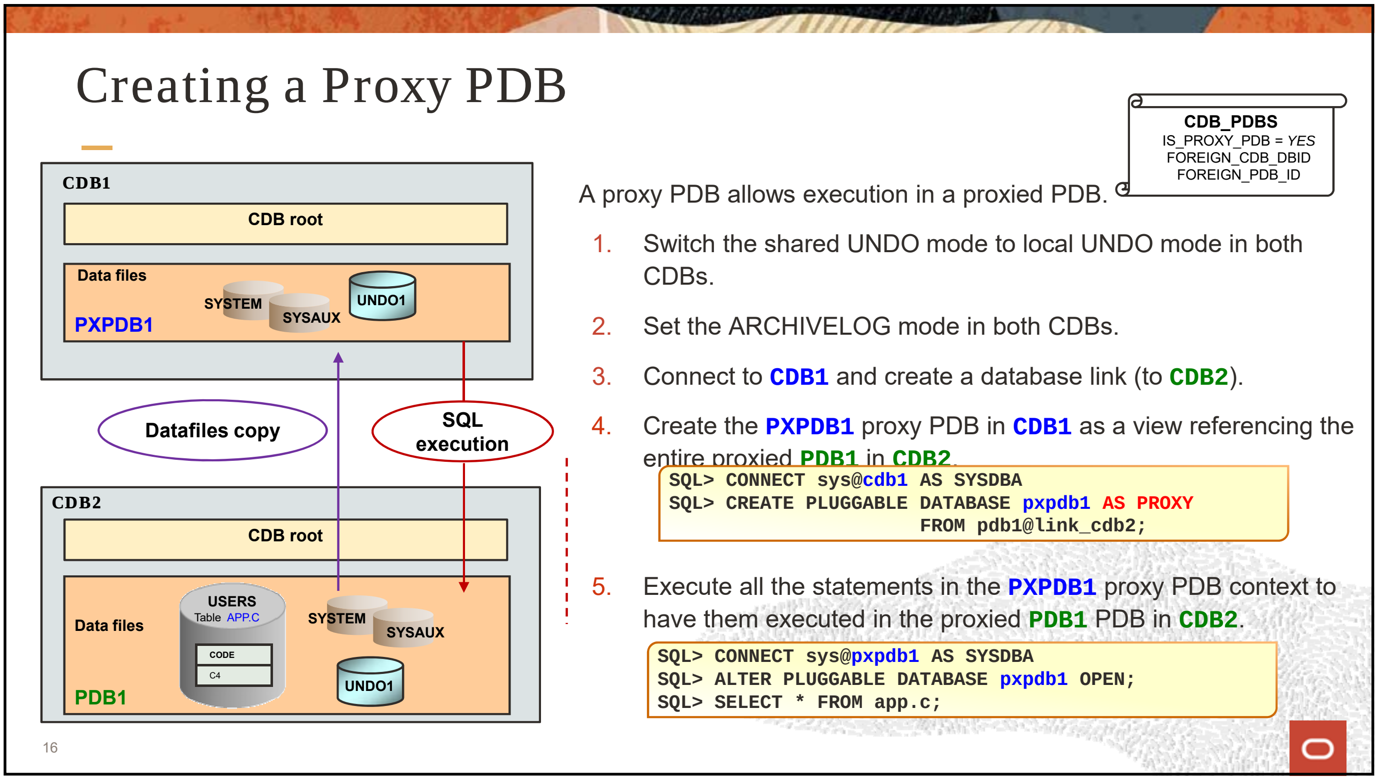
PX_APP _RR

CDB1

2 Create proxy PDB

➔ Retrieves rows from the shared table whose data is stored in application PDBs in the application root and replicas in CDBs

Revenue	Year	CDB$NAME	CON$NAME
15000000	2014	CDB1	ROBOTS
20000000	2014	CDB2	DOODLES
10000000	2014	CDB1	DOLLS



Creating a Proxy PDB

CDB1

CDB root

Data files

SYSTEM

SYSAUX

UNDO1

PXPDB1

Datafiles copy

SQL execution

CDB2

CDB root

Data files

USERS

Table APP.C

CODE

C4

SYSTEM

SYSAUX

UNDO1

PDB1

CDB_PDBS
IS_PROXY_PDB = *YES*
FOREIGN_CDB_DBID
FOREIGN_PDB_ID

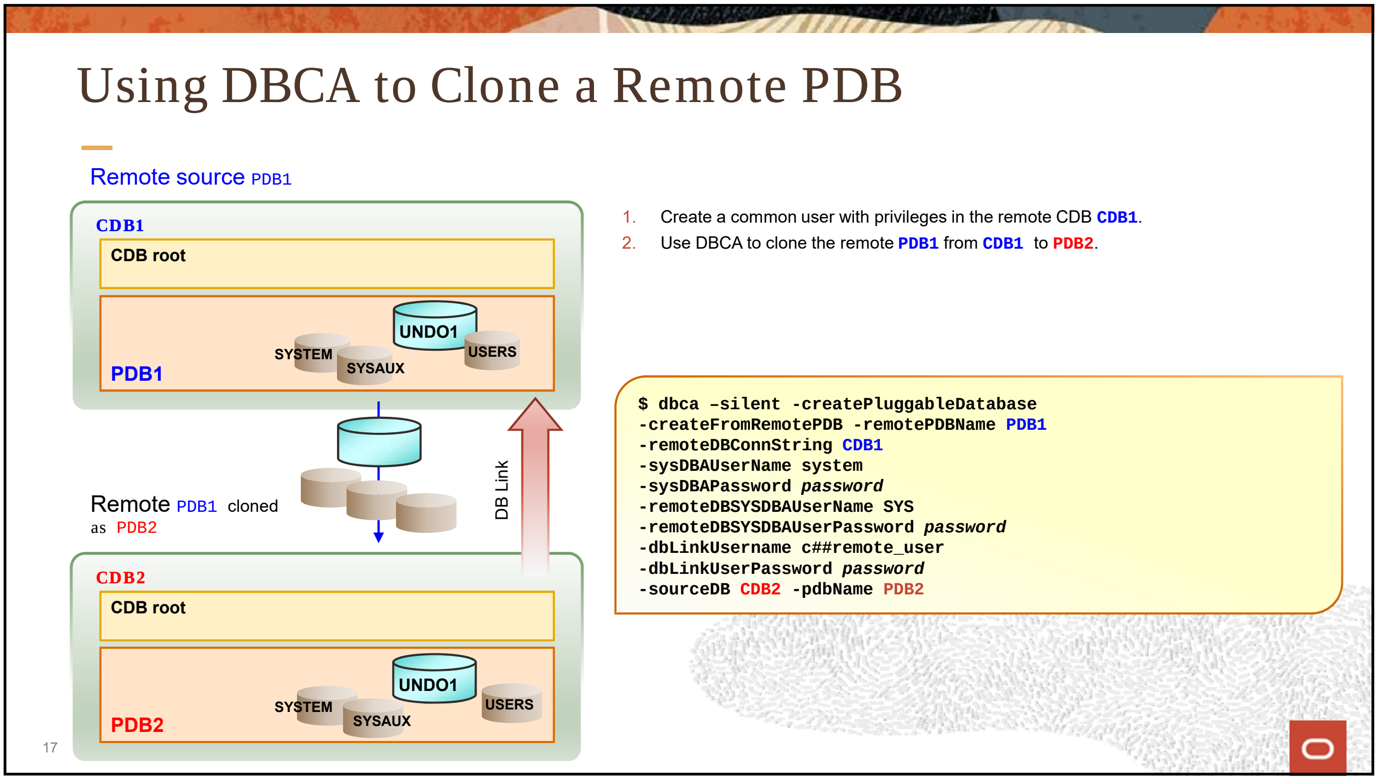
A proxy PDB allows execution in a proxied PDB.

1. Switch the shared UNDO mode to local UNDO mode in both CDBs.
2. Set the ARCHIVELOG mode in both CDBs.
3. Connect to **CDB1** and create a database link (to **CDB2**).
4. Create the **PXPDB1** proxy PDB in **CDB1** as a view referencing the entire proxied **PDB1** in **CDB2**.

```
SQL> CONNECT sys@cdb1 AS SYSDBA
SQL> CREATE PLUGGABLE DATABASE pxpdb1 AS PROXY
                              FROM pdb1@link_cdb2;
```

5. Execute all the statements in the **PXPDB1** proxy PDB context to have them executed in the proxied **PDB1** PDB in **CDB2**.

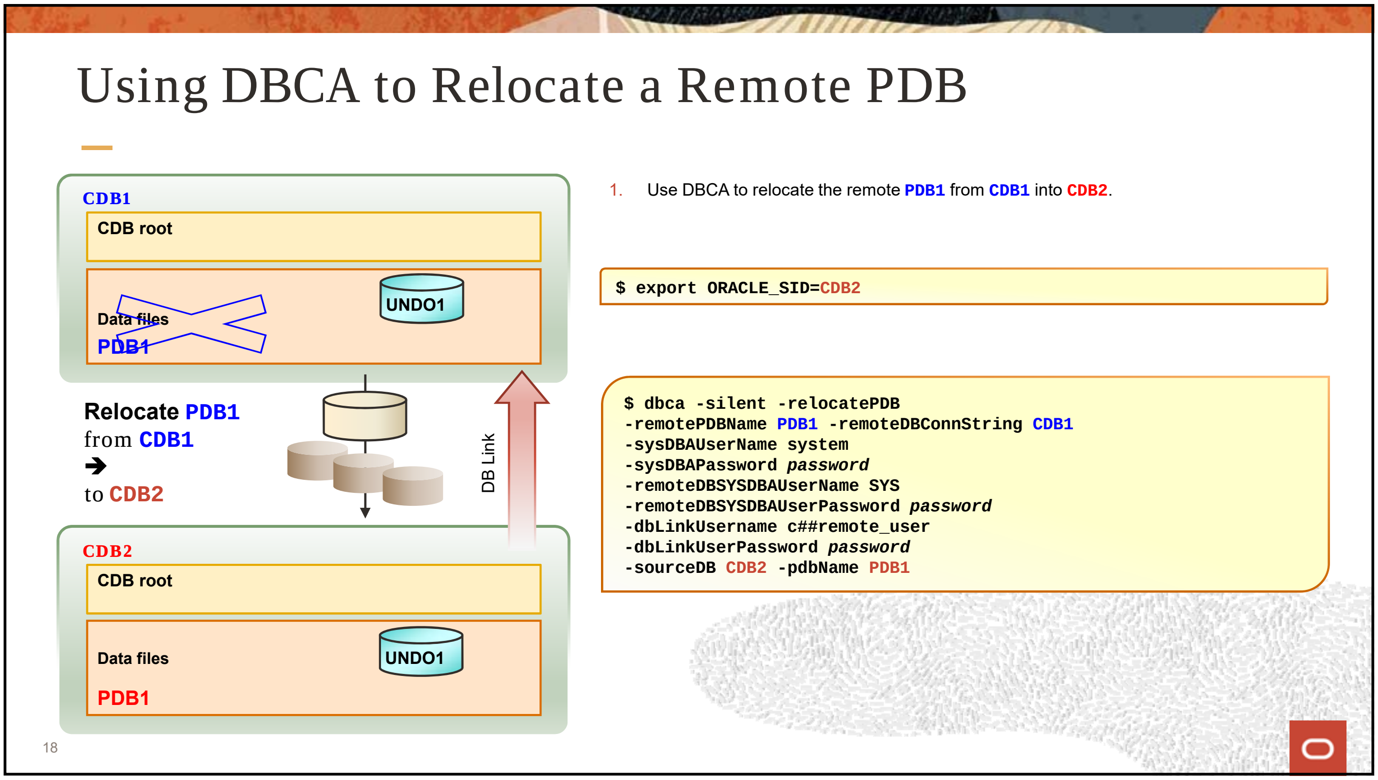
```
SQL> CONNECT sys@pxpdb1 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pxpdb1 OPEN;
SQL> SELECT * FROM app.c;
```



Using DBCA to Clone a Remote PDB

1. Create a common user with privileges in the remote CDB **CDB1**.
2. Use DBCA to clone the remote **PDB1** from **CDB1** to **PDB2**.

```
$ dbca –silent -createPluggableDatabase
-createFromRemotePDB -remotePDBName PDB1
-remoteDBConnString CDB1
-sysDBAUserName system
-sysDBAPassword password
-remoteDBSYSDBAUserName SYS
-remoteDBSYSDBAUserPassword password
-dbLinkUsername c##remote_user
-dbLinkUserPassword password
-sourceDB CDB2 -pdbName PDB2
```

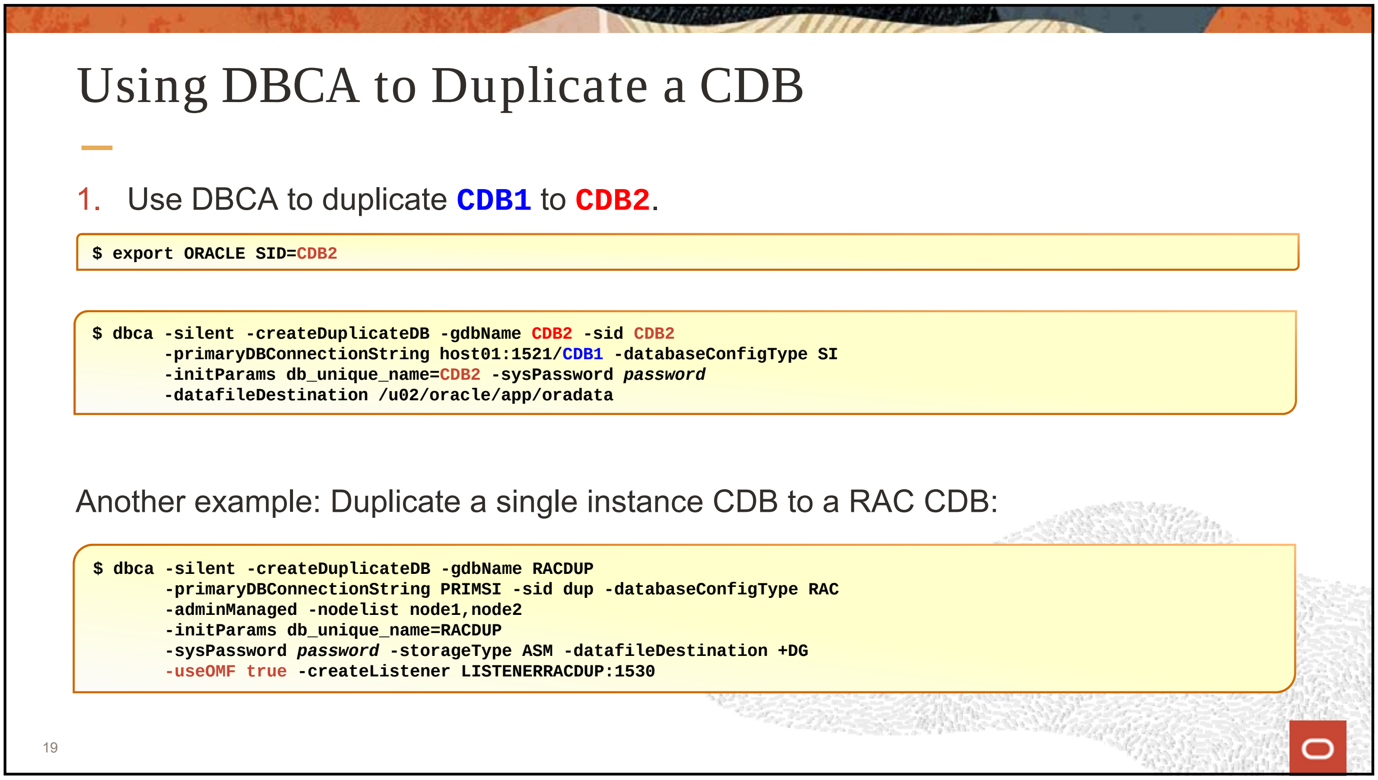


Using DBCA to Relocate a Remote PDB

1. Use DBCA to relocate the remote **PDB1** from **CDB1** into **CDB2**.

```
$ export ORACLE_SID=CDB2
```

```
$ dbca -silent -relocatePDB
-remotePDBName PDB1 -remoteDBConnString CDB1
-sysDBAUserName system
-sysDBAPassword password
-remoteDBSYSDBAUserName SYS
-remoteDBSYSDBAUserPassword password
-dbLinkUsername c##remote_user
-dbLinkUserPassword password
-sourceDB CDB2 -pdbName PDB1
```



Using DBCA to Duplicate a CDB

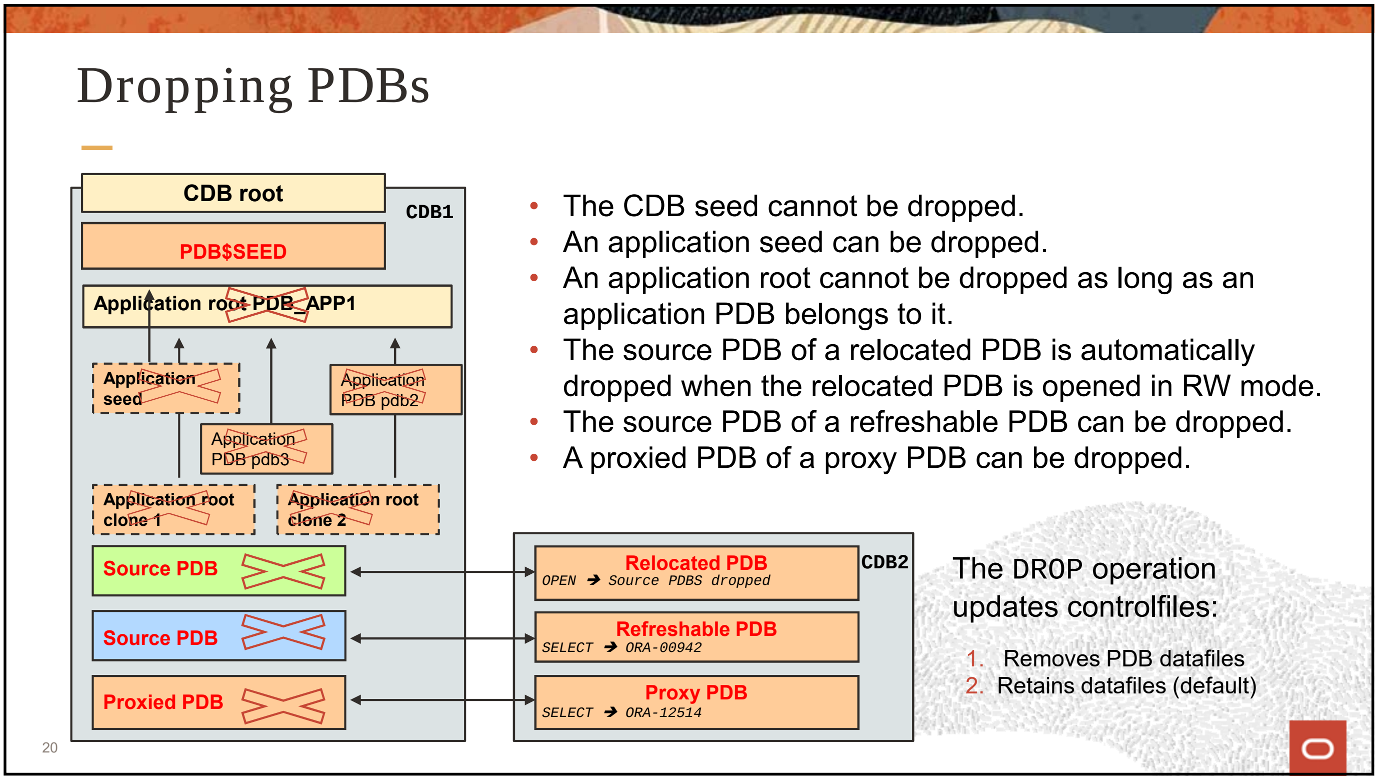
1. Use DBCA to duplicate **CDB1** to **CDB2**.

```
$ export ORACLE SID=CDB2
```

```
$ dbca -silent -createDuplicateDB -gdbName CDB2 -sid CDB2
       -primaryDBConnectionString host01:1521/CDB1 -databaseConfigType SI
       -initParams db_unique_name=CDB2 -sysPassword password
       -datafileDestination /u02/oracle/app/oradata
```

Another example: Duplicate a single instance CDB to a RAC CDB:

```
$ dbca -silent -createDuplicateDB -gdbName RACDUP
       -primaryDBConnectionString PRIMSI -sid dup -databaseConfigType RAC
       -adminManaged -nodelist node1,node2
       -initParams db_unique_name=RACDUP
       -sysPassword password -storageType ASM -datafileDestination +DG
       -useOMF true -createListener LISTENERRACDUP:1530
```



Dropping PDBs

CDB root

CDB1

PDB$SEED

Application ~~root PDB~~_APP1

~~Application seed~~

~~Application PDB pdb2~~

~~Application PDB pdb3~~

~~Application root clone 1~~

~~Application root clone 2~~

Source PDB

Source PDB

Proxied PDB

CDB2

Relocated PDB
OPEN ➔ Source PDBS dropped

Refreshable PDB
SELECT ➔ ORA-00942

Proxy PDB
SELECT ➔ ORA-12514

- The CDB seed cannot be dropped.
- An application seed can be dropped.
- An application root cannot be dropped as long as an application PDB belongs to it.
- The source PDB of a relocated PDB is automatically dropped when the relocated PDB is opened in RW mode.
- The source PDB of a refreshable PDB can be dropped.
- A proxied PDB of a proxy PDB can be dropped.

The DROP operation updates controlfiles:

1. Removes PDB datafiles
2. Retains datafiles (default)

Summary

In this lesson, you should have learned how to:

- Clone a regular PDB
- Clone an application container
- Unplug and plug or clone a non-CDB
- Unplug and plug a regular PDB
- Unplug and plug an application container
- Convert regular PDBs to application PDBs
- Configure and use the local UNDO mode
- Perform hot cloning
- Perform near-zero down time PDB relocation
- Create and use a proxy PDB
- Drop PDBs

Practice 4: Overview

- 4-1: Cloning remote regular PDBs in hot mode
- 4-2: Cloning an application container
- 4-3: Unplugging and plugging application containers
- 4-4: Converting a regular PDB to an application PDB
- 4-5: Relocating PDBs
- 4-6: Querying data across CDBs by using proxy PDBs
- 4-7: Dropping unnecessary PDBs