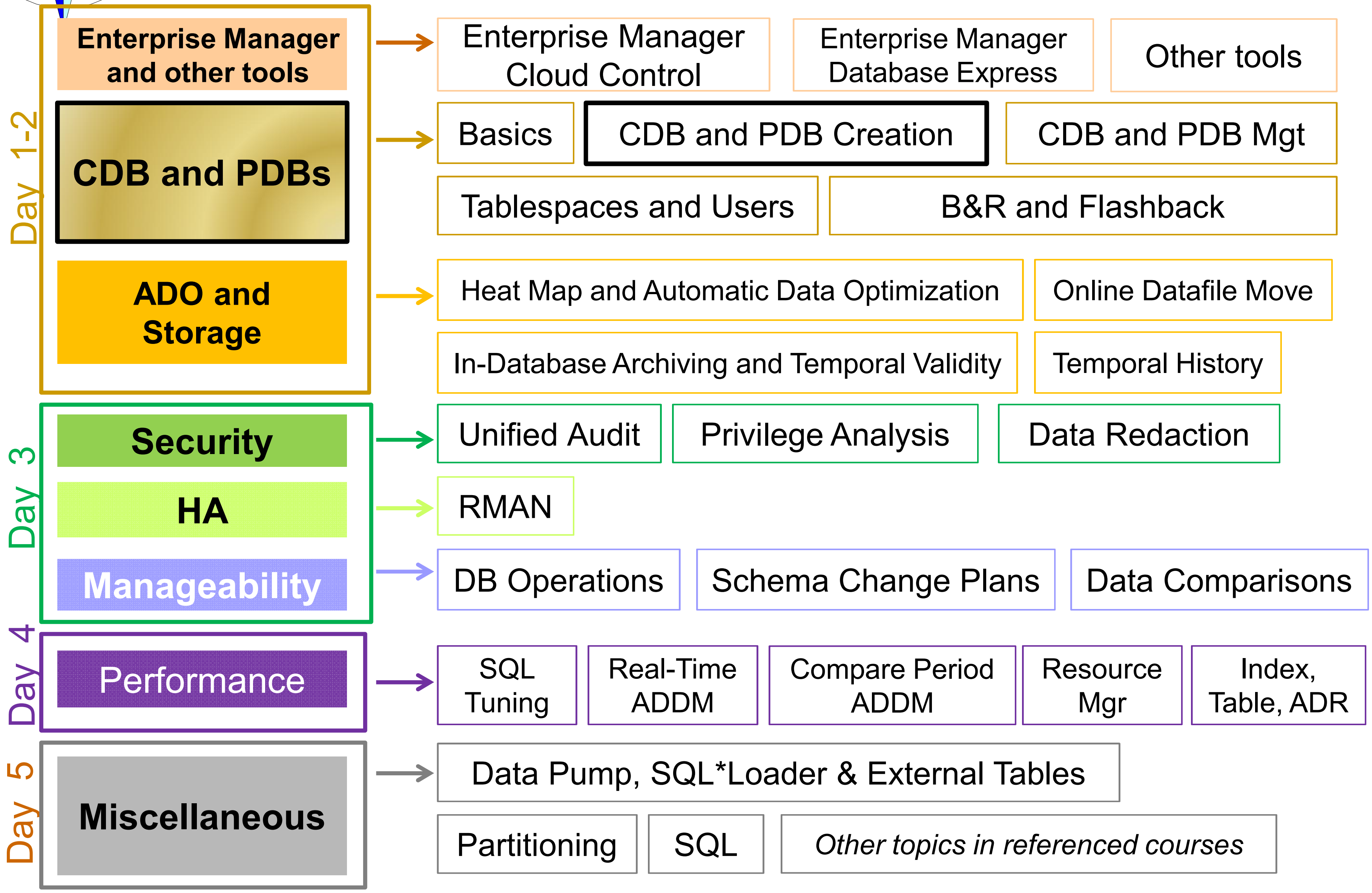
BCI

2

Creating Multitenant Container Databases and Pluggable Databases

Oracle Database 12*c* New and Enhanced Features



Day 1-2

Enterprise Manager and other tools → Enterprise Manager Cloud Control | Enterprise Manager Database Express | Other tools

CDB and PDBs → Basics | CDB and PDB Creation | CDB and PDB Mgt | Tablespaces and Users | B&R and Flashback

ADO and Storage → Heat Map and Automatic Data Optimization | Online Datafile Move | In-Database Archiving and Temporal Validity | Temporal History

Day 3

Security → Unified Audit | Privilege Analysis | Data Redaction

HA → RMAN

Manageability → DB Operations | Schema Change Plans | Data Comparisons

Day 4

Performance → SQL Tuning | Real-Time ADDM | Compare Period ADDM | Resource Mgr | Index, Table, ADR

Day 5

Miscellaneous → Data Pump, SQL*Loader & External Tables | Partitioning | SQL | *Other topics in referenced courses*

Objectives

After completing this lesson, you should be able to:

- Configure and create a CDB
- Create a PDB from `PDB$SEED`
- Create a PDB from a non-CDB
- Clone a PDB into the same CDB
- Unplug and plug a PDB from one CDB to another a CDB
- Explore the instance
- Explore the structure of PDBs
- Drop a PDB
- Migrate a pre-12.1 non-CDB database to CDB

Goals

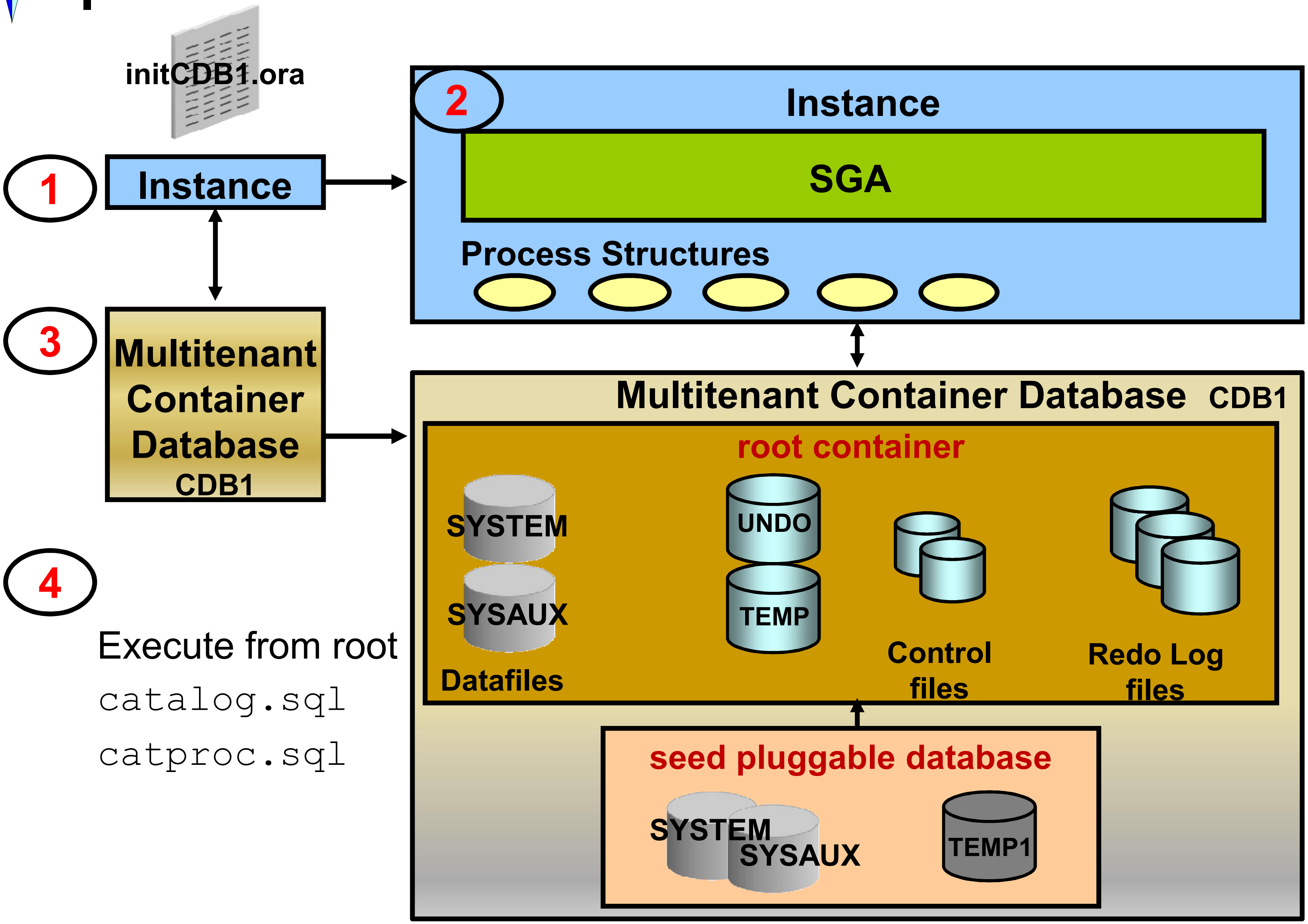
Create a multitenant container database:

- To consolidate many pre-12.1 non-CDBs into a single, larger database
- To prepare a container
  - For plugging any future new application
  - For testing applications
  - For diagnosing application performance
- To simplify and reduce time for patching and upgrade

Tools

	SQL*Plus	OUI	DBCA	EM Cloud Control	SQL Developer	DBUA
Create a new CDB or PDB	Yes	Yes	Yes	Yes (PDB only)	Yes (PDB only)	
Explore CDB instance, architecture, PDBs	Yes			Yes	Yes	
Upgrade a 12.1 CDB to 12.x CDB	Yes			Yes		Yes

Steps to Create a Multitenant Container Database



initCDB1.ora

1 Instance

2 Instance

SGA

Process Structures

3 Multitenant Container Database CDB1

Multitenant Container Database CDB1

root container

SYSTEM

SYSAUX

Datafiles

UNDO

TEMP

Control files

Redo Log files

seed pluggable database

SYSTEM

SYSAUX

TEMP1

4

Execute from root

```
catalog.sql
catproc.sql
```

Creating a Multitenant Container Database: Using SQL*Plus

1. Instance startup:
   a. Set ORACLE_SID=CDB1
   b. Set in initCDB1.ora:
      - Set CONTROL_FILES to CDB control file names
      - Set DB_NAME to CDB name
      - Set ENABLE_PLUGGABLE_DATABASE to TRUE

```
SQL> CONNECT / AS SYSDBA
SQL> STARTUP NOMOUNT
```

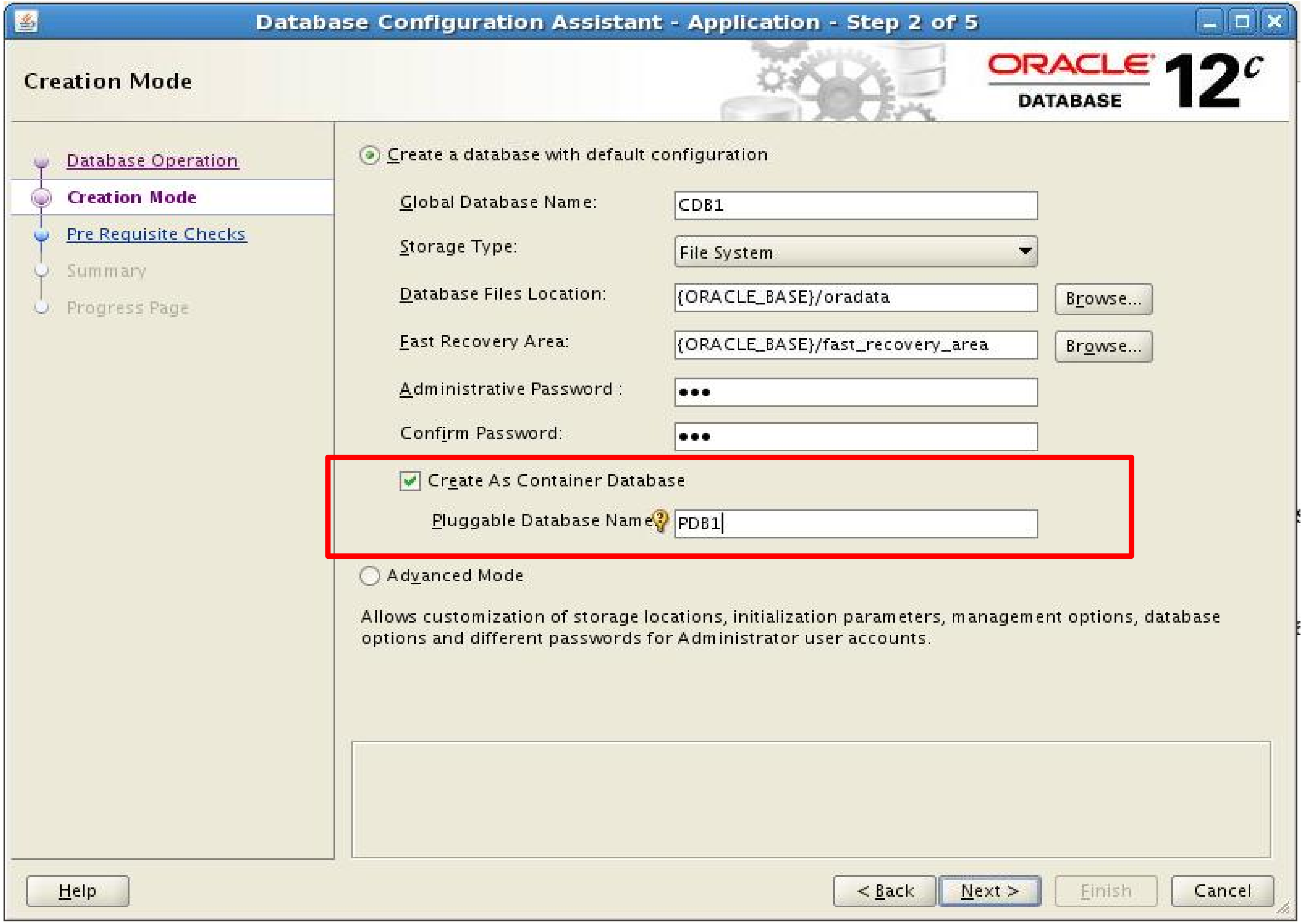
2. Create the database:

```
SQL> CREATE DATABASE CDB1 ENABLE PLUGGABLE DATABASE …
  2  SEED FILE_NAME_CONVERT ('/oracle/dbs','/oracle/seed');
```

   - CDB$ROOT container
   - PDB$SEED pluggable database

3. Close/open the seed PDB and run post-creation scripts.

Creating a Multitenant Container Database: Using DBCA



Creation Mode

● Create a database with default configuration

Global Database Name: CDB1

Storage Type: File System

Database Files Location: {ORACLE_BASE}/oradata

Fast Recovery Area: {ORACLE_BASE}/fast_recovery_area

Administrative Password : •••

Confirm Password: •••

☑ Create As Container Database

Pluggable Database Name: PDB1

○ Advanced Mode

Allows customization of storage locations, initialization parameters, management options, database options and different passwords for Administrator user accounts.

New Clause: SEED FILE_NAME_CONVERT

CREATE DATABASE new clauses:

```
SQL> CREATE DATABASE cdb1
  2   USER SYS IDENTIFIED BY p1 USER SYSTEM IDENTIFIED BY p2
  3   LOGFILE GROUP 1 ('/u01/app/oradata/CDB1/redo1a.log',
  4                    '/u02/app/oradata/CDB1/redo1b.log') SIZE 100M,
  5           GROUP 2 ('/u01/app/oradata/CDB1/redo2a.log',
  6                    '/u02/app/oradata/CDB1/redo2b.log') SIZE 100M
  7   CHARACTER SET AL32UTF8 NATIONAL CHARACTER SET AL16UTF16
  8   EXTENT MANAGEMENT LOCAL DATAFILE
  9                    '/u01/app/oradata/CDB1/system01.dbf' SIZE 325M
 10   SYSAUX DATAFILE  '/u01/app/oradata/CDB1/sysaux01.dbf' SIZE 325M
 11   DEFAULT TEMPORARY TABLESPACE tempts1
 12           TEMPFILE '/u01/app/oradata/CDB1/temp01.dbf' SIZE 20M
 13   UNDO TABLESPACE undotbs
 14           DATAFILE '/u01/app/oradata/CDB1/undotbs01.dbf' SIZE 200M
 15  ENABLE PLUGGABLE DATABASE
 16  SEED   FILE_NAME_CONVERT =
 17        ('/u01/app/oradata/CDB1',
 18         '/u01/app/oradata/CDB1/seed');
```

New Clause: **ENABLE PLUGGABLE DATABASE**

Without **SEED FILE_NAME_CONVERT**:

- OMF: **DB_CREATE_FILE_DEST=**'/u01/app/oradata'
- Or new instance parameter:
  PDB_FILE_NAME_CONVERT =
  '/u01/app/oradata/CDB1','/u01/app/oradata/seed'

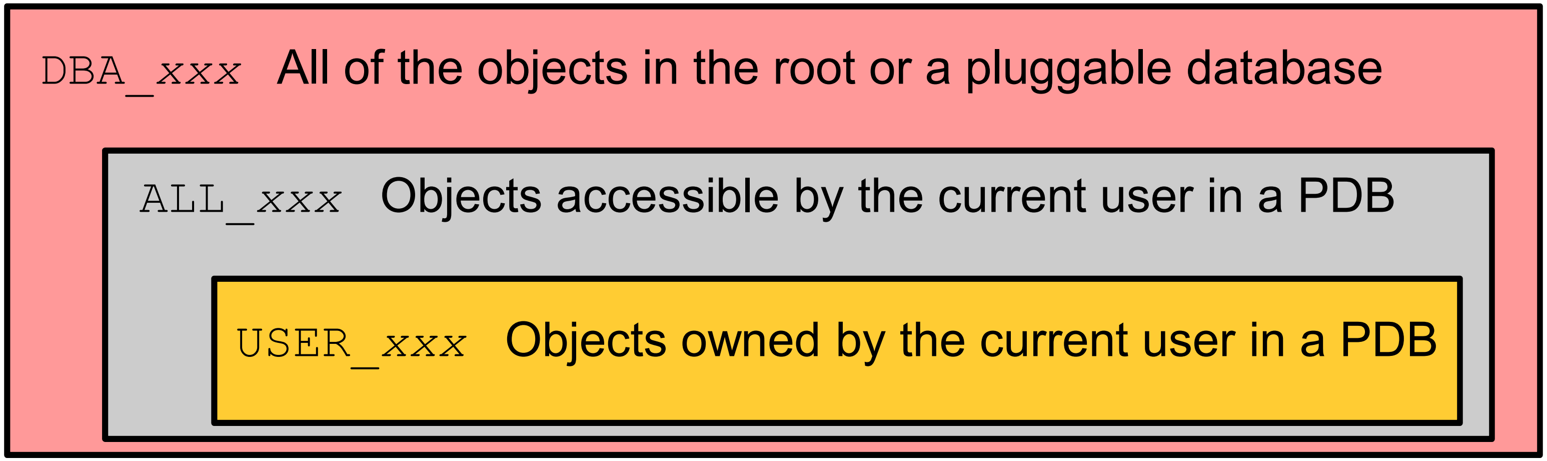
```
SQL> CONNECT / AS SYSDBA
SQL> STARTUP NOMOUNT

SQL> CREATE DATABASE cdb2
  2   USER SYS IDENTIFIED BY p1 USER SYSTEM IDENTIFIED BY p2
  3   EXTENT MANAGEMENT LOCAL
  4   DEFAULT TEMPORARY TABLESPACE temp
  5   UNDO TABLESPACE undotbs
  6   DEFAULT TABLESPACE users
  7   ENABLE PLUGGABLE DATABASE;
```

After CDB Creation: What’s New in CDB

A CDB has new characteristics compared to non-CDBs:

- **Two containers:**
  - The root (`CDB$ROOT`)
  - The seed PDB (`PDB$SEED`)
- **Several services:** one per container
  - Name of root service = name of the CDB (`cdb1`)
- **Common** users in root and seed: `SYS`, `SYSTEM` …
- **Common** privileges granted to common users
- **Pre-defined** common roles
- **Tablespaces** and data files associated to each container:
  - root:
    - `SYSTEM`: system-supplied metadata and no user data
    - `SYSAUX`
  - seed: `SYSTEM`, `SYSAUX`

Data Dictionary Views: DBA_*xxx*



DBA_*xxx* All of the objects in the root or a pluggable database

ALL_*xxx* Objects accessible by the current user in a PDB

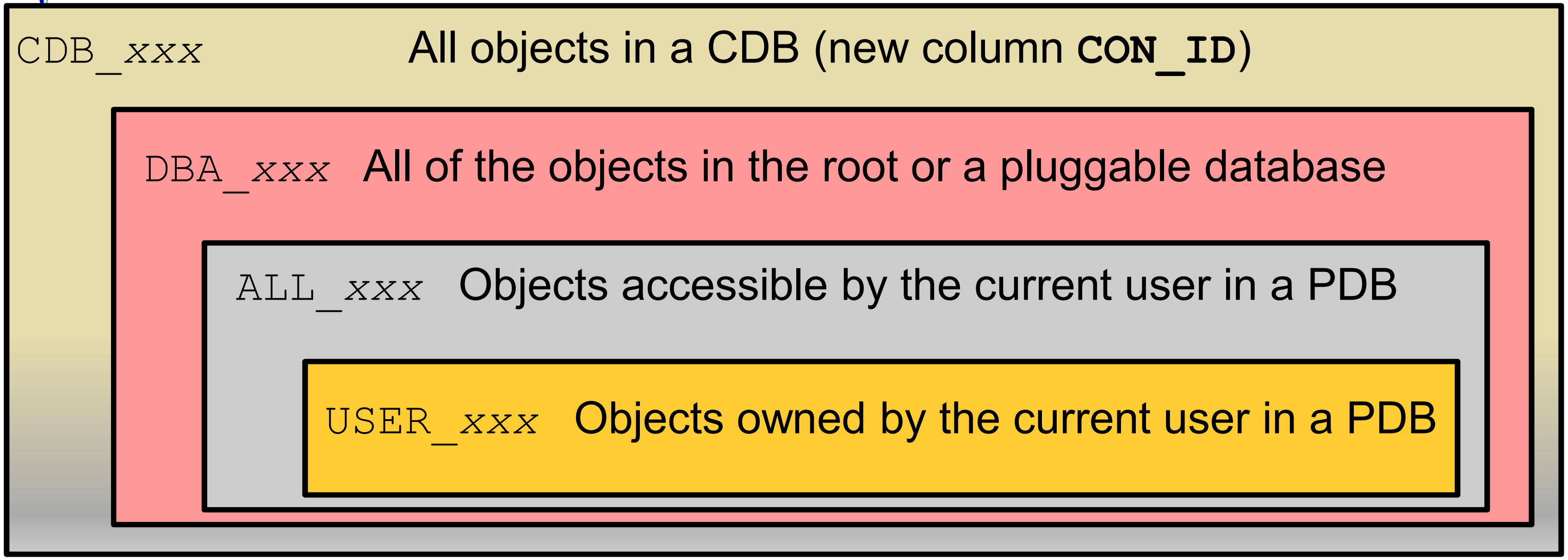
USER_*xxx* Objects owned by the current user in a PDB

DBA dictionary views providing information within PDB:

```
SQL> SELECT table_name FROM   dict
  2  WHERE  table_name like 'DBA%';
```

- DBA_tablespaces: All tablespaces of the PDB
- DBA_data_files: All data files of the PDB
- DBA_tables: All tables in the PDB
- DBA_users: All common and local users of the PDB

Data Dictionary Views: CDB_*xxx*



CDB_*xxx* All objects in a CDB (new column **CON_ID**)

DBA_*xxx* All of the objects in the root or a pluggable database

ALL_*xxx* Objects accessible by the current user in a PDB

USER_*xxx* Objects owned by the current user in a PDB

CDB dictionary views provide information across PDBs:

```
SQL> SELECT view_name FROM dba_views
  2  WHERE  view_name like 'CDB%';
```

- CDB_pdbs: All PDBS within the CDB
- CDB_tablespaces: All tablespaces within the CDB
- CDB_data_files: All datafiles within the CDB
- CDB_users: All users within the CDB (common and local)

Data Dictionary Views: Examples

- Comparisons:

1

```
SQL> CONNECT / AS SYSDBA
SQL> SELECT role, common, con_id FROM cdb_roles;
```

2

```
SQL> SELECT role, common FROM dba_roles;
```

3

```
SQL> CONNECT sys@PDB1 AS SYSDBA
SQL> SELECT role, common, con_id FROM cdb_roles;
```

4

```
SQL> SELECT role, common FROM dba_roles;
```

- Access to data in V$ or GV$ views showing data from multiple PDBs can be secured using privilege.

```
SQL> SELECT name,open_mode FROM v$pdbs;

NAME             OPEN_MODE
---------------- ----------
PDB$SEED       READ ONLY
PDB1           READ WRITE
PDB2           READ WRITE
```

Data Dictionary Views: V$xxx Views

SGA accessed by all containers: V$ views and CON_ID column

```
SQL> SELECT distinct status, con_id FROM v$bh order by 2;

STATUS           CON_ID
------------ ------
cr                    1              root
free                  1
xcur                  1
xcur                  2              seed  PDB
cr                    3              PDB1  PDB
xcur                  3
```

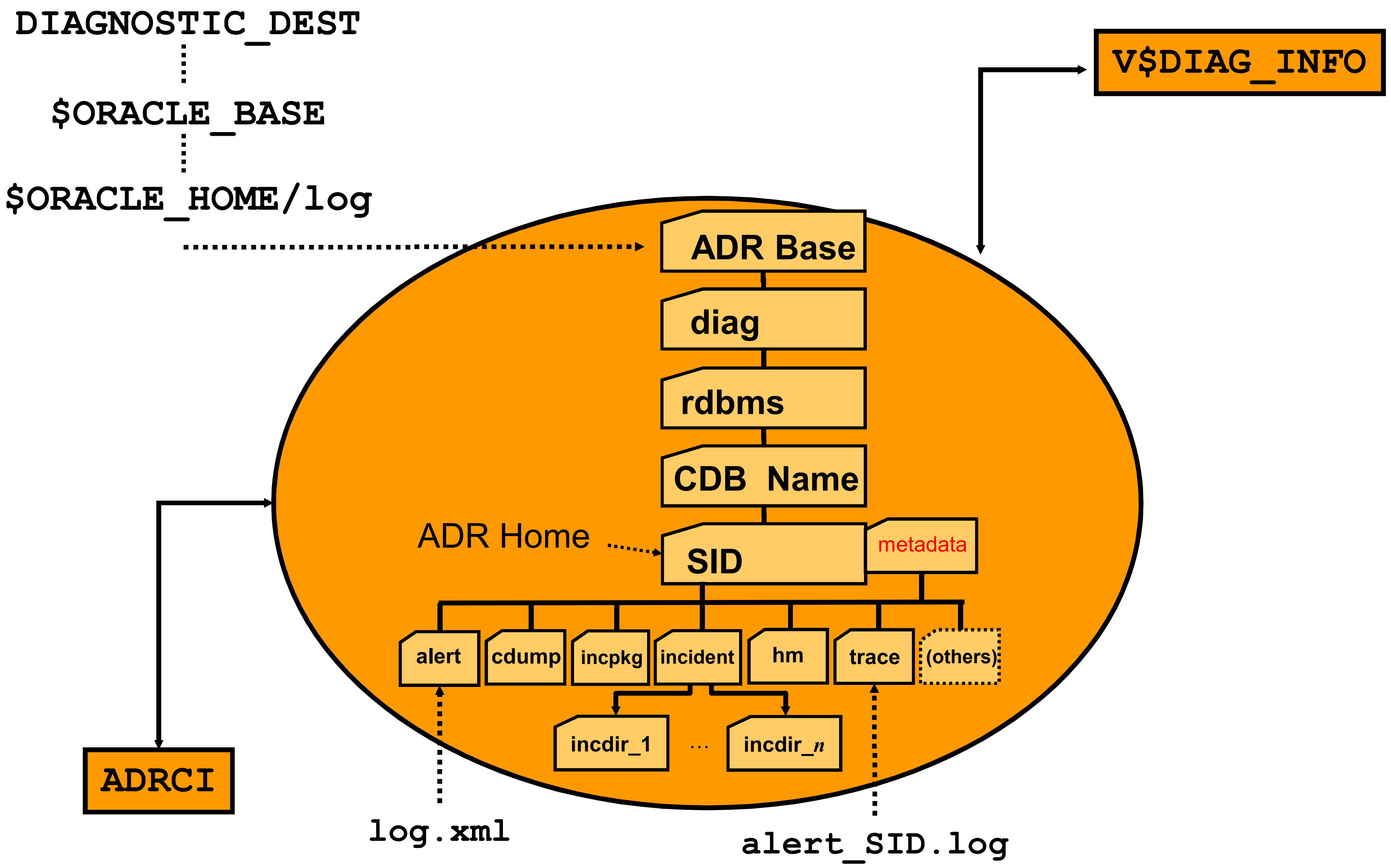
```
SQL> select OBJECT_ID, ORACLE_USERNAME, LOCKED_MODE, CON_ID
  2  from   V$LOCKED_OBJECT;

OBJECT_ID  ORACLE_USERNAME  LOCKED_MODE  CON_ID
---------- ---------------- ----------- -------
     83711 SYS                        3       3        PDB1  PDB
     83710 DOM                        3       4        PDB2  PDB
```

After CDB Creation: To-Do List

After CDB creation, the CDBA has to:

- Set a separate default tablespace for the root and for each PDB
- Set a default temporary tablespace for the entire CDB (optionally create additional temporary tablespaces in individual PDBs)
- Start the listener
- Plug non-CDBs
- Test startup/shutdown procedures
- Create new event triggers to automate PDBs opening
- Create backup and recovery procedures

Automatic Diagnostic Repository



DIAGNOSTIC_DEST

$ORACLE_BASE

$ORACLE_HOME/log

V$DIAG_INFO

ADR Base

diag

rdbms

CDB Name

ADR Home

SID

metadata

alert

cdump

incpkg

incident

hm

trace

(others)

incdir_1

…

incdir_*n*

ADRCI

log.xml

alert_SID.log

Automatic Diagnostic Repository: `alert.log` File

The `alert_CDB1.log` shows new DDL statements.

```
CREATE DATABASE cdb1
  …
ENABLE PLUGGABLE DATABASE
SEED
FILE_NAME_CONVERT=('/u01/app/oradata/CDB1','/u01/app/oradata
/seed');

CREATE PLUGGABLE DATABASE pdb1 … ;
ALTER PLUGGABLE DATABASE pdb1 UNPLUG INTO … ;
ALTER PLUGGABLE DATABASE ALL OPEN ;
ALTER PLUGGABLE DATABASE pdb2 CLOSE IMMEDIATE ;
```

Quiz

Which is a characteristic of the seed pluggable database of a CDB?

a. It is always kept in `READ ONLY` mode.

b. It is a not a container.

c. The seed can be dropped.

Quiz

You create a CDB. What is true about the seed pluggable database?

a. Copy the seed data files yourself.

b. Use the new clause `SEED FILE_NAME_CONVERT` in the `CREATE DATABASE` statement.

c. The seed pluggable database is not required.

d. The seed pluggable database does not require data files.

Practice 3 Overview: Creating a CDB and PDBs

The first practice covers the following topic:

- Creating a CDB with no PDBs

Provisioning New Pluggable Databases

Four methods:

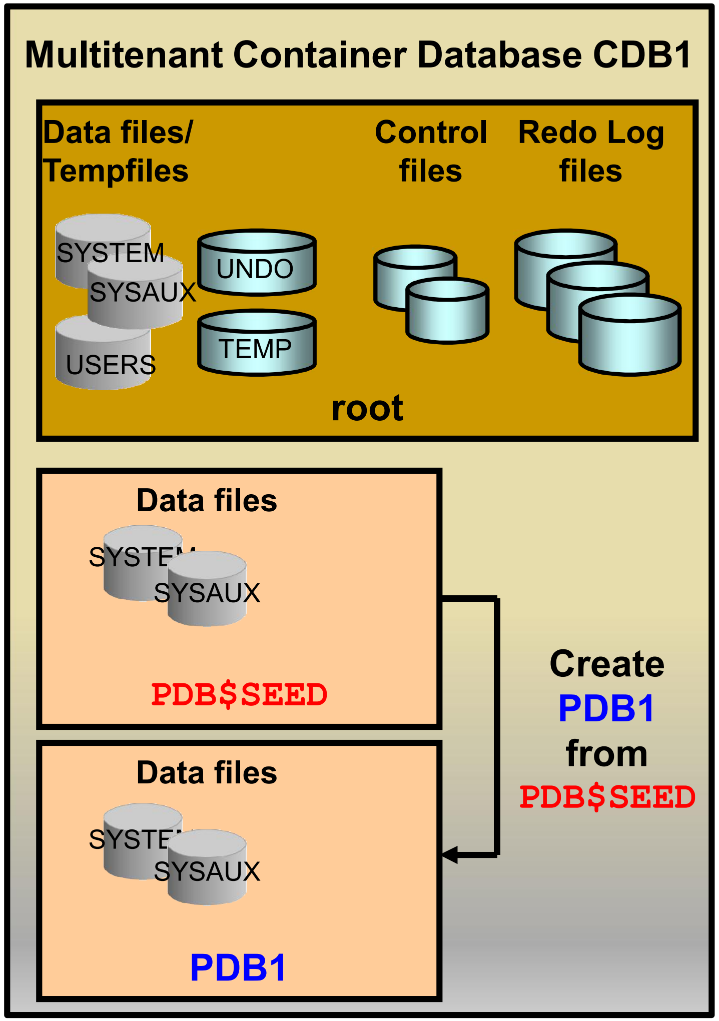
- Create a new PDB from the seed PDB.
- Plug a non-CDB in a CDB.
- Clone a PDB from another PDB:
  - Into the same CDB
- Plug an unplugged PDB into another CDB.

Tools

To provision new PDBs, you can use:

- SQL*Plus
- SQL Developer
- Enterprise Manager Cloud Control
- DBCA
  - Copy from seed
  - By unplugging / plugging method

Method 1: Create New PDB from PDB$SEED



- Copies the data files from PDB$SEED data files
- Creates tablespaces SYSTEM, SYSAUX
- Creates a full catalog including metadata pointing to Oracle-supplied objects
- Creates common users:
  - Superuser SYS
  - SYSTEM
- Creates a local user (PDBA) granted local PDB_DBA role
- Creates a new default service

Steps: With FILE_NAME_CONVERT

Create a new PDB from the seed using **FILE_NAME_CONVERT**:

1. Connect to the root as a common user with CREATE PLUGGABLE DATABASE system privilege:

```
SQL> CREATE PLUGGABLE DATABASE pdb1
  2  ADMIN USER admin1 IDENTIFIED BY p1 ROLES=(CONNECT)
  3  FILE_NAME_CONVERT = ('PDB$SEEDdir', 'PDB1dir');
```

2. Use views to verify:

```
SQL> CONNECT / AS SYSDBA
SQL> SELECT * FROM cdb_pdbs;
SQL> SELECT * FROM cdb_tablespaces;
SQL> SELECT * FROM cdb_data_files;
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN RESTRICTED;
SQL> CONNECT sys@pdb1 AS SYSDBA
SQL> CONNECT admin1@pdb1
```

Steps: Without FILE_NAME_CONVERT

Create a new PDB from seed without **FILE_NAME_CONVERT**:

- OMF: **DB_CREATE_FILE_DEST =** '/u01/app/oradata/CDB1/**pdb1**'

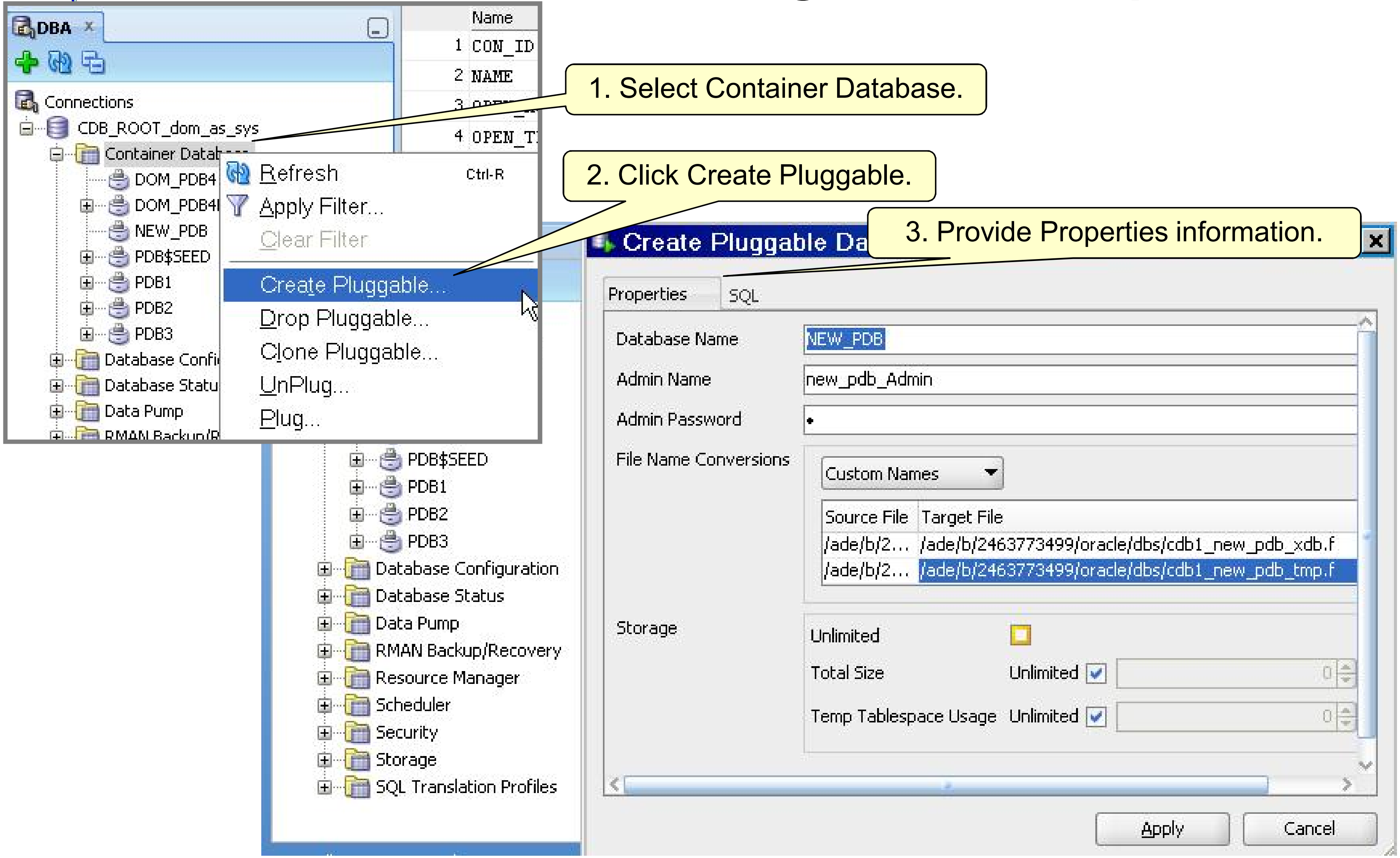
```
SQL> CREATE PLUGGABLE DATABASE pdb1
  2  ADMIN USER pdb1_admin IDENTIFIED BY p1
  3  ROLES=(CONNECT);
```

Or

- New parameter: **PDB_FILE_NAME_CONVERT =** '/u01/app/oradata/CDB1/seed','/u01/app/oradata/CDB1/**pdb1**'

```
SQL> CREATE PLUGGABLE DATABASE pdb1
  2  ADMIN USER pdb1_admin IDENTIFIED BY p1
  3  ROLES=(CONNECT);
```

Method 1: Using SQL Developer



1. Select Container Database.

2. Click Create Pluggable.

3. Provide Properties information.

Create Pluggable Da...

Properties | SQL

Database Name: NEW_PDB

Admin Name: new_pdb_Admin

Admin Password: •

File Name Conversions: Custom Names

Source File	Target File
/ade/b/2...	/ade/b/2463773499/oracle/dbs/cdb1_new_pdb_xdb.f
/ade/b/2...	/ade/b/2463773499/oracle/dbs/cdb1_new_pdb_tmp.f

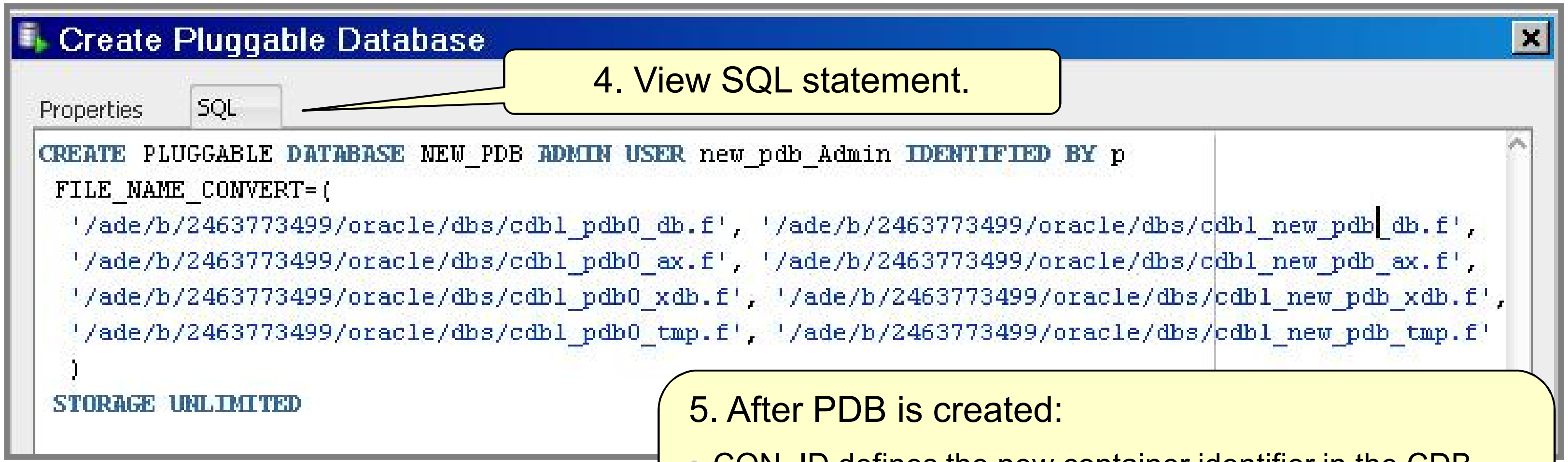
Storage

Unlimited □

Total Size: Unlimited ☑ 0

Temp Tablespace Usage: Unlimited ☑ 0

Method 1: Using SQL Developer

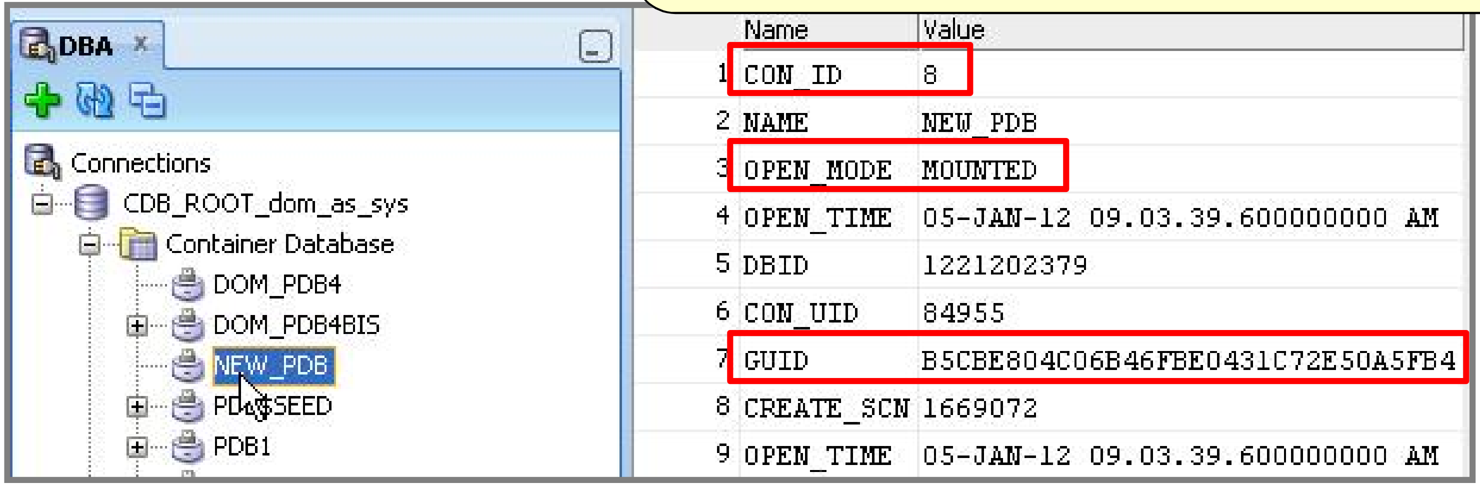


4. View SQL statement.

```
CREATE PLUGGABLE DATABASE NEW_PDB ADMIN USER new_pdb_Admin IDENTIFIED BY p
 FILE_NAME_CONVERT=(
  '/ade/b/2463773499/oracle/dbs/cdb1_pdb0_db.f', '/ade/b/2463773499/oracle/dbs/cdb1_new_pdb_db.f',
  '/ade/b/2463773499/oracle/dbs/cdb1_pdb0_ax.f', '/ade/b/2463773499/oracle/dbs/cdb1_new_pdb_ax.f',
  '/ade/b/2463773499/oracle/dbs/cdb1_pdb0_xdb.f', '/ade/b/2463773499/oracle/dbs/cdb1_new_pdb_xdb.f',
  '/ade/b/2463773499/oracle/dbs/cdb1_pdb0_tmp.f', '/ade/b/2463773499/oracle/dbs/cdb1_new_pdb_tmp.f'
  )
 STORAGE UNLIMITED
```

5. After PDB is created:

- CON_ID defines the new container identifier in the CDB.
- `OPEN_MODE` defines the open status, by default `MOUNTED`.
- `GUID` defines the new global unique container identifier.

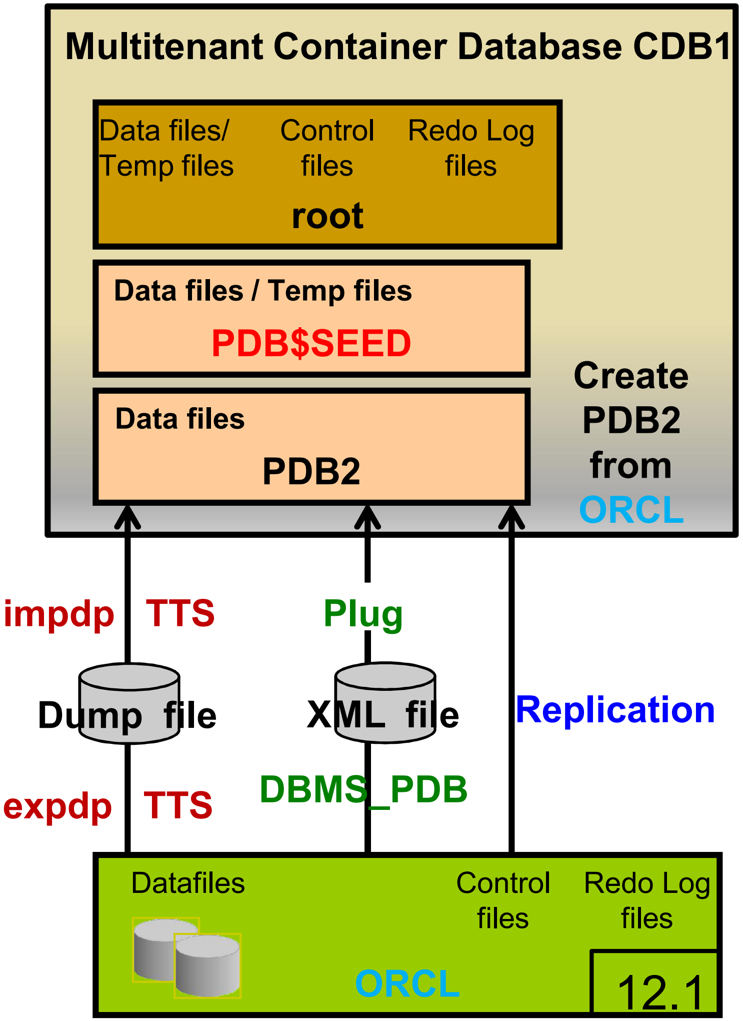


	Name	Value
1	CON_ID	8
2	NAME	NEW_PDB
3	OPEN_MODE	MOUNTED
4	OPEN_TIME	05-JAN-12 09.03.39.600000000 AM
5	DBID	1221202379
6	CON_UID	84955
7	GUID	B5CBE804C06B46FBE0431C72E50A5FB4
8	CREATE_SCN	1669072
9	OPEN_TIME	05-JAN-12 09.03.39.600000000 AM

Synchronization

1. Customer-created common users or roles in root:
   - Cannot be created, modified, dropped when PDB is in `READ-ONLY` mode
   - Can be created, modified, dropped when PDB is in `MOUNTED` mode
2. When opening the PDB:
   - In `READ-ONLY` mode, an error is returned.
   - In `READ-WRITE` mode, synchronization with the target CDB is automatically completed.
   - A compatibility check is automatically performed:
     - Any violation is reported in the `PDB_PLUG_IN_VIOLATIONS` view.
     - If there are no violation, the PDB status is changed to `NORMAL`.

Method 2: Plug a Non-CDB into CDB



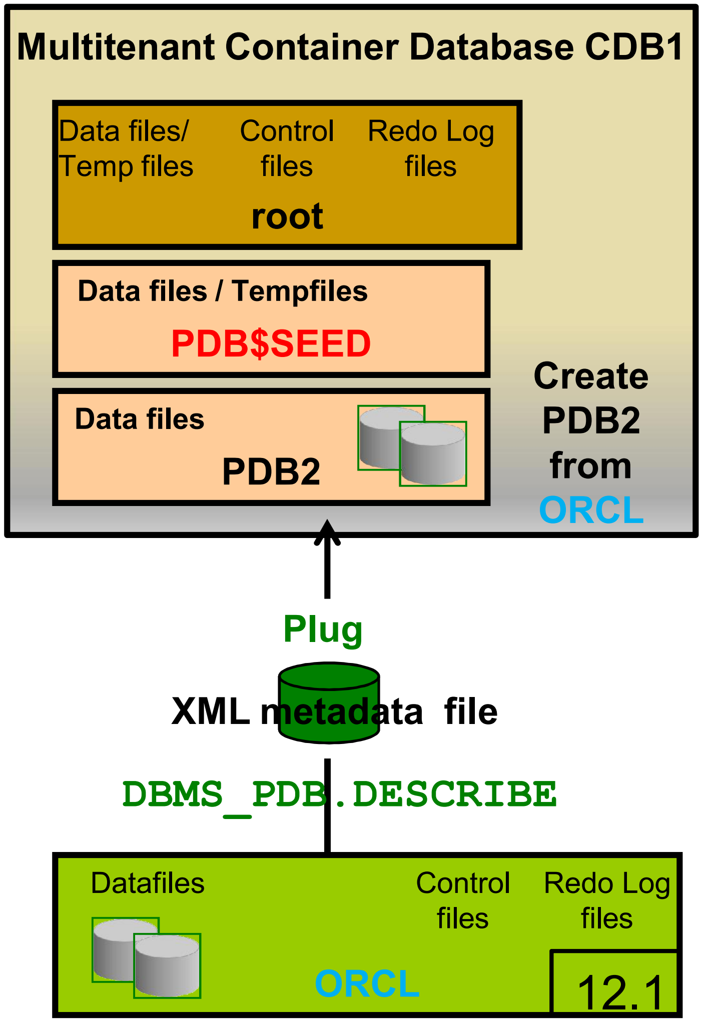
Three possible methods:

- TTS or TDB or full export/import
- XML file definition with `DBMS_PDB`
- Replication

Entities are created in the new PDB:

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

Plug a Non-CDB in to CDB Using DBMS_PDB



1. Open ORCL in READ ONLY mode

2.

```
SQL> EXEC DBMS_PDB.DESCRIBE
  ('/tmp/ORCL.xml')
```

3. Connect to the target cdb as a common user with CREATE PLUGGABLE DATABASE privilege

4. Plug in the unplugged PDB ORCL as PDB2

```
SQL> CREATE PLUGGABLE DATABASE
  2  PDB2 USING '/tmp/ORC.xml';
```

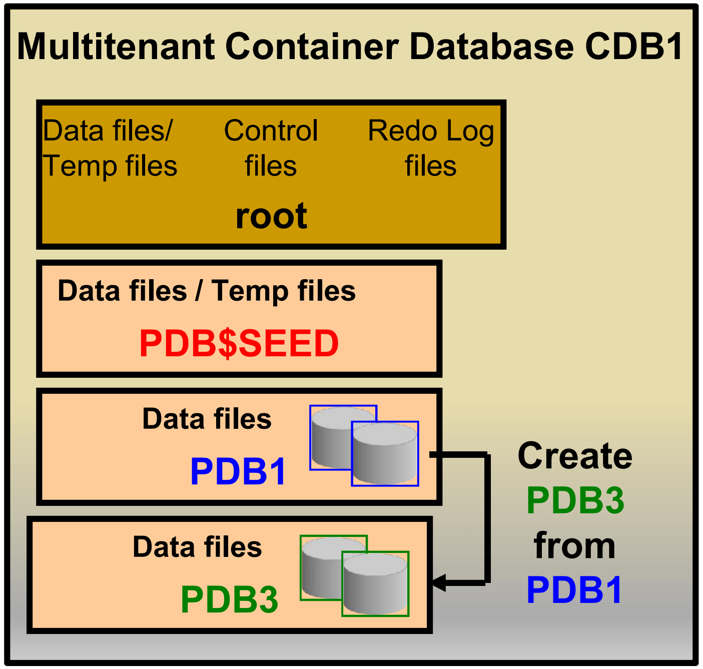
5. Run the noncdb_to_pdb.sql script

```
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.

Method 3: Clone PDBs



PDB3 owns:

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

1. In init.ora, set `DB_CREATE_FILE_DEST= 'PDB3dir'` or `PDB_FILE_NAME_CONVERT='PDB1dir' , 'PDB3dir'`
2. Connect to the root.
3. Quiesce **PDB1** (Close **PDB1** before):

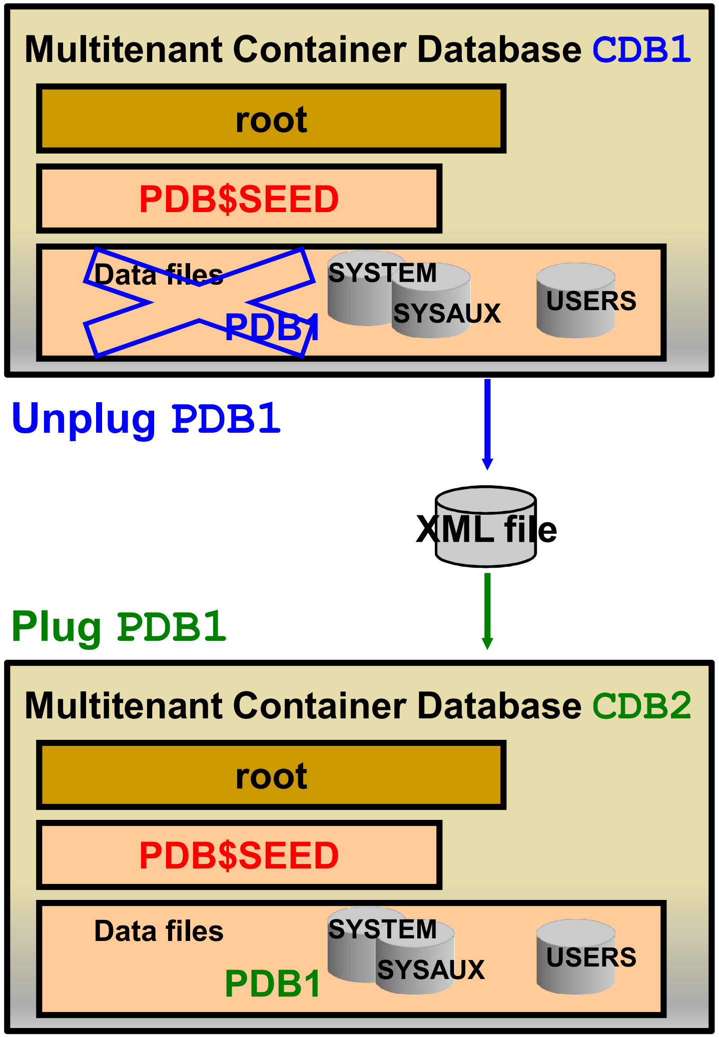
```
SQL> ALTER PLUGGABLE DATABASE
  2  pdb1 OPEN READ ONLY;
```

4. Clone **PDB3** from **PDB1:**

```
SQL> CREATE PLUGGABLE DATABASE
  2  pdb3 FROM pdb1;
```

5. Open **PDB3** in read-write mode.

Method 4: Plug Unplugged PDB in to CDB



Unplug PDB1 from CDB1:

1. Connect to CDB1 as a common user.
2. Verify that PDB1 is opened `READ ONLY`.
3.

```
SQL> ALTER PLUGGABLE DATABASE
  2  pdb1 UNPLUG INTO
  3                 'xmlfile1.xml';
```

4. Drop PDB1 from CDB1.

Plug PDB1 in to CDB2:

1. Connect to CDB2 as a common user.
2. Use `DBMS_PDB` package to check the compatibility of PDB1 with CDB2.
3.

```
SQL> CREATE PLUGGABLE DATABASE
  2     pdb1 USING 'xmlfile1.xml'
  3  NOCOPY;
```

4. Open PDB1 in read-write mode.

Method 4: Flow

Several clauses can be used in conjunction:

Are new PDB files based on same files that were used to create existing PDB in CDB?

If not, `AS CLONE` clause is required and so,it ensures that Oracle Database generates unique PDB DBID, GUID, and other identifiers expected for the new PDB.

XML file accurately describes current locations of files?

If not, the `SOURCE_FILE_NAME_CONVERT` clause is required.

Are files are in correct location?

If not, specify `COPY` to copy files to new location or `MOVE` to move them to another location.

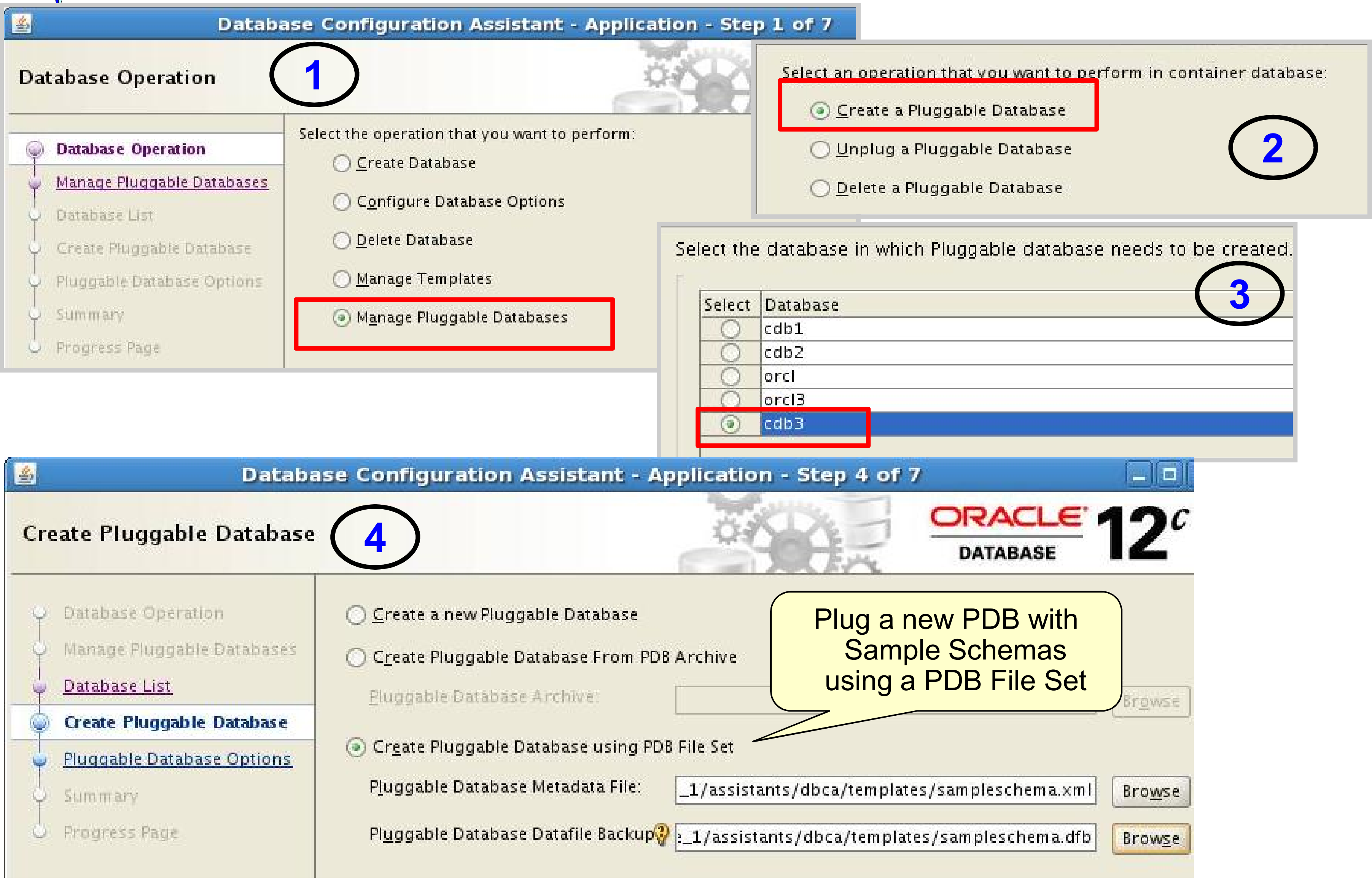
If yes, use `NOCOPY`. `COPY` is the default.

- `FILE_NAME_CONVERT` clause of `CREATE PLUGGABLE DATABASE` statement
- OMF: `DB_CREATE_FILE_DEST` parameter
- `PDB_FILE_NAME_CONVERT` parameter

Do you want to specify storage limits for PDB?

If yes, specify the `STORAGE` clause.

Plug Sample Schemas PDB: Using DBCA



Database Configuration Assistant - Application - Step 1 of 7

Database Operation

1

Database Operation
Manage Pluggable Databases
Database List
Create Pluggable Database
Pluggable Database Options
Summary
Progress Page

Select the operation that you want to perform:

○ Create Database

○ Configure Database Options

○ Delete Database

○ Manage Templates

● Manage Pluggable Databases

2

Select an operation that you want to perform in container database:

● Create a Pluggable Database

○ Unplug a Pluggable Database

○ Delete a Pluggable Database

3

Select the database in which Pluggable database needs to be created.

Select	Database
○	cdb1
○	cdb2
○	orcl
○	orcl3
●	cdb3

Database Configuration Assistant - Application - Step 4 of 7

Create Pluggable Database

4

ORACLE
DATABASE
12c
Database Operation
Manage Pluggable Databases
Database List
Create Pluggable Database
Pluggable Database Options
Summary
Progress Page

○ Create a new Pluggable Database

○ Create Pluggable Database From PDB Archive

Pluggable Database Archive: Browse

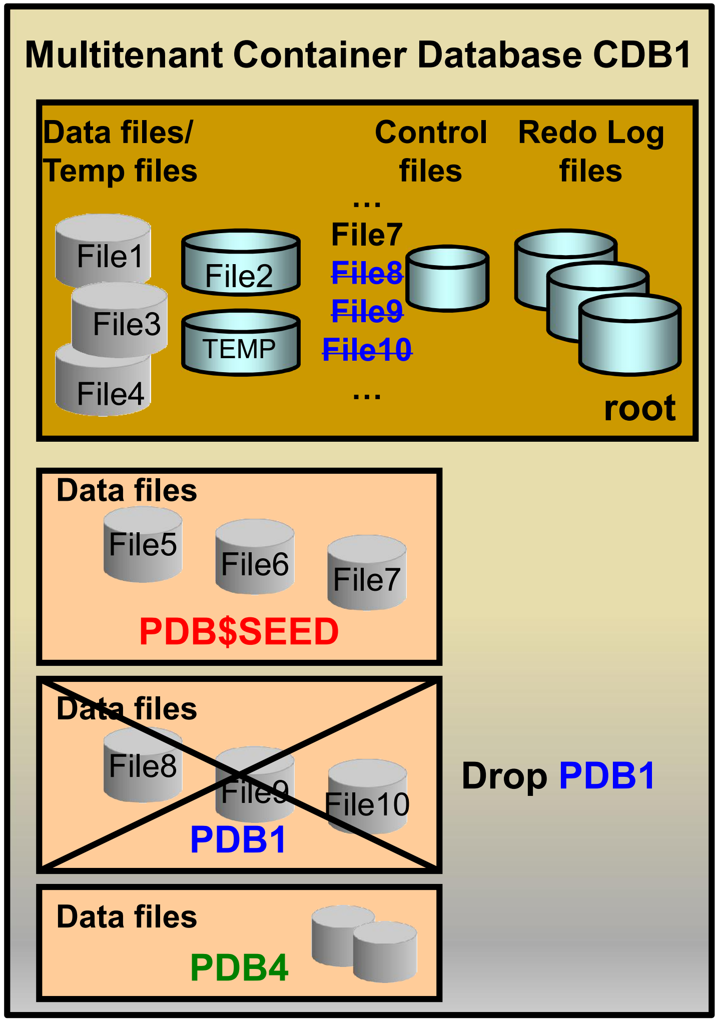
● Create Pluggable Database using PDB File Set

Pluggable Database Metadata File: _1/assistants/dbca/templates/sampleschema.xml Browse

Pluggable Database Datafile Backup: :_1/assistants/dbca/templates/sampleschema.dfb Browse

Plug a new PDB with Sample Schemas using a PDB File Set

Dropping a PDB



```
SQL> ALTER PLUGGABLE DATABASE
  2  pdb1 CLOSE;
SQL> DROP PLUGGABLE DATABASE
  2  pdb1 [INCLUDING DATAFILES];
```

- Updates control files
- If `INCLUDING DATAFILES`:
  - Removes **PDB1** datafiles
- If `KEEP DATAFILES` (default):
  - Retain data files
  - Can be plugged in another or the same CDB
- Requires `SYSDBA` privilege
- Cannot drop seed PDB

Migrating pre-12.1 Databases to 12.1 CDB

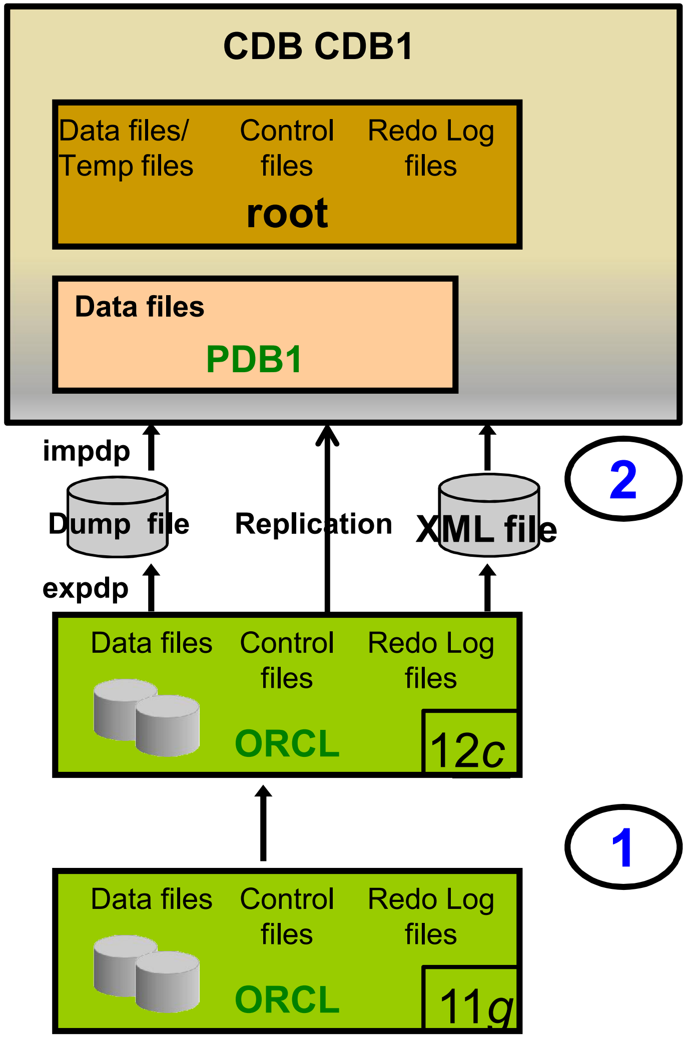
There are two methods:

1. Upgrade an existing pre-12.1 database to 12*c.*
2. Plug-in non-CDB into a CDB.

Or

1. Pre-create a PDB in CDB.
2. Use 11*g* expdp / 12*c* impdp

   or replication between non-CDB and PDB.



Quiz

Which of the following are true about cloning a PDB into the same CDB? Select all that apply.

a. It is not possible. You can only clone a PDB into another CDB.

b. You can clone only one PDB into the same CDB.

c. Cloning a PDB can use the source files copy method to the target PDB files.

d. Cloning a PDB can use the clause NOCOPY if the target PDB files will use the source files.

Quiz

Which of the following are true about dropping a PDB?

a. You can drop a PDB only if the PDB is closed.

b. You can possibly drop the seed PDB, but you will not be able to create any other PDB within the CDB.

c. You can drop a PDB and keep the data files to be reused by another PDB.

d. When you drop a PDB, the data files and redo log files are automatically removed from the storage file system.

Summary

In this lesson, you should have learned how to:

- Configure and create a CDB
- Create a PDB from `PDB$SEED`
- Create a PDB from a non-CDB
- Clone a PDB into the same CDB
- Unplug and plug a PDB from one CDB to another a CDB
- Explore the instance and structure of PDBs
- Drop a PDB
- Migrate pre-12.1 non-CDB database to CDB

Practice 3 Overview: Creating a CDB and PDBs

These practices cover the following topics:

- Creating a new PDB into a CDB using the seed
- Cloning a PDB from a CDB into the same CDB
- Plugging a non-CDB in to a CDB
- Merging two CDBs into a single one
- Dropping a PDB