

UNIT 5

Multitenant Database Architecture

(7 Hours) — Database Administration (CACS405)

*Note: These notes are prepared for study purposes and **may contain errors or omissions**. Syntax, file paths and screen names differ between Oracle versions and operating systems (examples here follow Oracle 19c on Linux). Please verify important points with your textbook, class notes and the official Oracle documentation, and test commands on a practice database, never on a production system.*

5.1 Understanding the Multitenant Architecture

Before Oracle 12c, every database was a stand-alone **non-CDB** with its own instance, memory and background processes. Running many small databases on one server wasted memory and CPU and multiplied patching and backup work. The **multitenant architecture** (introduced in 12c) solves this by letting one **container database (CDB)** hold many **pluggable databases (PDBs)**. To an application, a PDB looks and behaves like an ordinary database.

Note: Oracle 19c supports both non-CDB and CDB architectures. From Oracle 21c the non-CDB architecture is no longer supported, so new databases are created as CDBs.

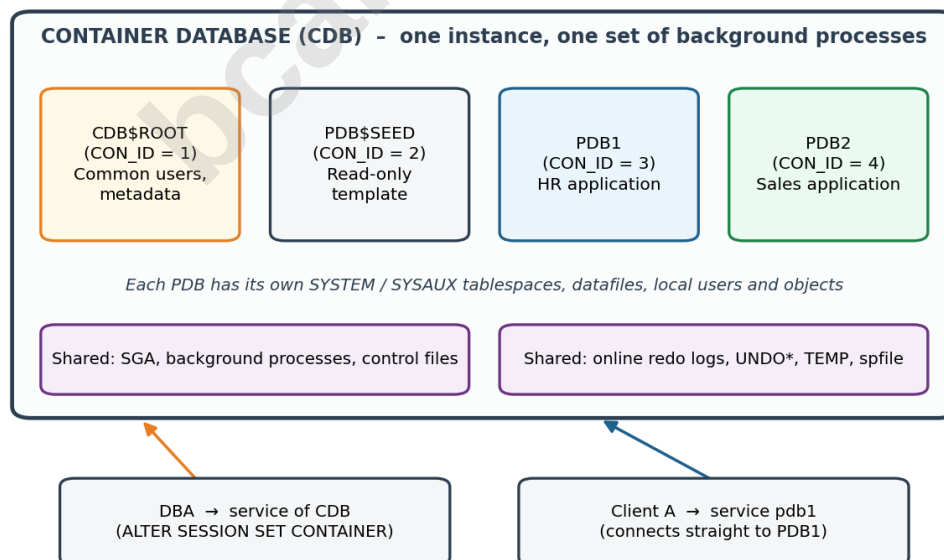


Figure 5.1: A container database with root, seed and pluggable databases

Container	CON_ID	Description
CDB\$ROOT (root)	1	Holds Oracle-supplied metadata and common users . No application data should be stored here.
PDB\$SEED (seed)	2	Read-only template used to create new PDBs quickly.
User-created PDB	3 and above	A portable database with its own application data, local users and tablespaces.

5.1.1 What is Shared and What is Separate

Shared by the whole CDB	Separate for each PDB
One instance: SGA, PGA and background processes	Own SYSTEM and SYSAUX tablespaces (PDB-specific data dictionary)
Control files and online redo logs	Own datafiles for application tablespaces
Undo tablespace (shared undo) — or local undo per PDB if enabled	Local users, roles, schemas and objects
Common users (C##...), root metadata, spfile	Own default temporary tablespace (optional) and service name

5.1.2 Advantages

- **Consolidation:** many databases share one instance, reducing memory, CPU and administration cost.
- **Rapid provisioning:** new databases are created by cloning the seed or an existing PDB in seconds or minutes.
- **Patch and upgrade once:** patching the CDB applies to all PDBs; a PDB can be unplugged and plugged into a CDB of a newer version.
- **Portability:** a PDB can be unplugged and moved to another CDB or cloned to a remote CDB.
- **Resource control:** Resource Manager can share CPU and I/O between PDBs; each PDB is isolated from others.

Point	Non-CDB	CDB with PDBs
Instances needed for 10 databases	10	1
Patching	Each database separately	Once for the CDB
Data dictionary	One per database	Root plus one per PDB (metadata links)

Point	Non-CDB	CDB with PDBs
Users	Local only	Common (C##) and local users

5.2 Creating a Container Database (CDB)

A CDB is created using DBCA by selecting **Create as Container Database** (and optionally the number of PDBs and the PDB name). A manual creation uses the `ENABLE PLUGGABLE DATABASE` clause of `CREATE DATABASE`.

```
CREATE DATABASE cdb1
  USER SYS IDENTIFIED BY sys_pwd USER SYSTEM IDENTIFIED BY system_pwd
  ... (same clauses as a normal database) ...
  ENABLE PLUGGABLE DATABASE
  SEED FILE_NAME_CONVERT = ('/u01/oradata/CDB1/',
                           '/u01/oradata/CDB1/pdbseed/');
```

Listing 5.1: Creating a CDB (outline)

After creation, verify that the database is a CDB and see which container you are in.

```
SQL> SELECT name, cdb, con_id FROM v$database;
SQL> SHOW CON_NAME
SQL> SHOW CON_ID
SQL> SELECT con_id, name, open_mode FROM v$pdb;
SQL> SELECT pdb_id, pdb_name, status FROM dba_pdb;
SQL> SELECT con_id, name FROM v$containers;
```

Listing 5.2: Checking the CDB

5.3 Creating Pluggable Databases (PDBs) within a CDB

PDBs are created while connected to the **root** (CDB\$ROOT) as a user with the `CREATE PLUGGABLE DATABASE` privilege (for example SYS). There are several methods.

5.3.1 Create from the Seed

```
CREATE PLUGGABLE DATABASE pdb1
  ADMIN USER pdb1_admin IDENTIFIED BY "Pdb1#Admin"
  FILE_NAME_CONVERT = ('/u01/oradata/CDB1/pdbseed/',
                      '/u01/oradata/CDB1/pdb1/');

ALTER PLUGGABLE DATABASE pdb1 OPEN;
SHOW PDBS
```

Listing 5.3: New PDB from PDB\$SEED

If Oracle Managed Files or the parameter `PDB_FILE_NAME_CONVERT` is set, `FILE_NAME_CONVERT` can be omitted. A newly created PDB is in `MOUNTED` state until it is opened.

5.3.2 Clone an Existing PDB

```
-- local clone within the same CDB
CREATE PLUGGABLE DATABASE pdb2 FROM pdb1
  FILE_NAME_CONVERT = ('/pdb1/', '/pdb2/');
ALTER PLUGGABLE DATABASE pdb2 OPEN;

-- remote clone through a database link
CREATE PLUGGABLE DATABASE pdb3 FROM pdb1@remote_cdb_link
  FILE_NAME_CONVERT = ('/pdb1/', '/pdb3/');
```

Listing 5.4: Cloning PDBs

Note: In 12.1 the source PDB had to be read-only during cloning. From 12.2 onward, hot cloning of an open PDB is possible if the CDB uses local undo and ARCHIVELOG mode.

5.3.3 Unplug and Plug

```
-- unplug from CDB1
ALTER PLUGGABLE DATABASE pdb1 CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE pdb1 UNPLUG INTO '/tmp/pdb1.xml';
DROP PLUGGABLE DATABASE pdb1 KEEP DATAFILES;

-- plug into CDB2 (after checking compatibility)
-- SET SERVEROUTPUT ON
-- DECLARE ok BOOLEAN; BEGIN
--   ok := DBMS_PDB.CHECK_PLUG_COMPATIBILITY('/tmp/pdb1.xml', 'PDB1');
--   DBMS_OUTPUT.PUT_LINE(CASE WHEN ok THEN 'YES' ELSE 'NO' END); END;
CREATE PLUGGABLE DATABASE pdb1 USING '/tmp/pdb1.xml' NOCOPY TEMPFILE REUSE;
ALTER PLUGGABLE DATABASE pdb1 OPEN;
```

Listing 5.5: Unplugging and plugging a PDB

The XML manifest describes the PDB's files and settings. Options: `NOCOPY` (use files where they are), `COPY` (copy files to a new location) and `MOVE` (move files). A non-CDB can also be converted to a PDB by plugging it in with the `DBMS_PDB.DESCRIBE` manifest and running the `noncdb_to_pdb.sql` script.

5.4 Managing CDBs and PDBs

5.4.1 Connecting and Switching Containers

```
$ sqlplus / as sysdba                -- connects to CDB$ROOT
SQL> ALTER SESSION SET CONTAINER = pdb1;    -- switch to PDB1
SQL> SHOW CON_NAME
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT; -- back to root

$ sqlplus pdb1_admin/"Pdb1#Admin"@//dbserver:1521/pdb1 -- direct PDB service
```

Listing 5.6: Connecting to root and to a PDB

Each PDB automatically gets a **service** with its name, registered with the listener. Applications should connect using this service, not through the root.

5.4.2 Starting, Stopping and Removing PDBs

```
ALTER PLUGGABLE DATABASE pdb1 OPEN;                -- READ WRITE
ALTER PLUGGABLE DATABASE pdb1 OPEN READ ONLY;
ALTER PLUGGABLE DATABASE pdb1 CLOSE IMMEDIATE;
ALTER PLUGGABLE DATABASE ALL OPEN;
ALTER PLUGGABLE DATABASE pdb1 SAVE STATE;          -- auto-open after CDB restart

-- inside a PDB you may also use STARTUP / SHUTDOWN
SQL> ALTER SESSION SET CONTAINER = pdb1;
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP

DROP PLUGGABLE DATABASE pdb2 INCLUDING DATAFILES;
```

Listing 5.7: PDB open, close and drop

- Shutting down the CDB closes all PDBs. After a CDB restart, PDBs return to MOUNTED unless their state was saved with `SAVE STATE`.
- A PDB must be closed before it can be dropped or unplugged.
- Tablespaces and datafiles are created inside a PDB while connected to it: `CREATE TABLESPACE ...` affects only that PDB.
- Some initialization parameters can be set for one PDB (`ALTER SYSTEM SET ...` inside the PDB); `V$PARAMETER.ISPDB_MODIFIABLE` shows which.

5.4.3 Useful Views

View	Use
V\$CONTAINERS, V\$PDBS	Containers and their state from the root.
DBA_PDBS, CDB_PDBS	PDB details and status.
CDB_USERS, CDB_TABLESPACES, CDB_DATA_FILES	CDB_ views show information from all containers (from the root).
DBA_ views inside a PDB	Show only that PDB's objects.

Example 5.1

Create a PDB named `hrpdb`, open it, make it open automatically after every restart, and create a tablespace inside it.

Solution

```
CREATE PLUGGABLE DATABASE hrpdb ADMIN USER hradmin IDENTIFIED BY
"Hr#Admin1" FILE_NAME_CONVERT=('/pdbseed/', '/hrpdb/');
ALTER PLUGGABLE DATABASE hrpdb OPEN; then ALTER PLUGGABLE DATABASE hrpdb
SAVE STATE;
ALTER SESSION SET CONTAINER = hrpdb; then CREATE TABLESPACE hr_data
DATAFILE '/u01/oradata/CDB1/hrpdb/hr_data01.dbf' SIZE 100M;
```

5.5 Backup and Duplicate in a Multitenant Database

RMAN (Unit 7) fully supports multitenant databases. A backup can cover the entire CDB, the root only, or individual PDBs.

```
RMAN> CONNECT TARGET / -- to CDB root
RMAN> BACKUP DATABASE PLUS ARCHIVELOG; -- whole CDB (root, seed and all PDBs)
RMAN> BACKUP DATABASE ROOT; -- root only
RMAN> BACKUP PLUGGABLE DATABASE pdb1, pdb2; -- selected PDBs
RMAN> BACKUP TABLESPACE pdb1:hr_data; -- one tablespace of a PDB

-- restore and recover a single PDB (root remains open)
RMAN> ALTER PLUGGABLE DATABASE pdb1 CLOSE;
RMAN> RESTORE PLUGGABLE DATABASE pdb1;
RMAN> RECOVER PLUGGABLE DATABASE pdb1;
RMAN> ALTER PLUGGABLE DATABASE pdb1 OPEN;
```

Listing 5.8: RMAN backup and recovery for CDB and PDB

- You can also connect RMAN **directly to a PDB** as a local user with the `SYSDBA` or `SYSPRIV` privilege and back up only that PDB.

- **Point-in-time recovery of a PDB** (`RECOVER PLUGGABLE DATABASE ... UNTIL`) is easiest when the CDB uses **local undo**; otherwise RMAN builds an auxiliary instance.
- Cloning a PDB (`CREATE PLUGGABLE DATABASE ... FROM ...`) is a quick way to make a copy for testing.

Duplicate: RMAN `DUPLICATE` creates a copy of a database, for example to build a test system. In a multitenant environment you can duplicate a whole CDB or selected PDBs into a new CDB.

```

RMAN> CONNECT TARGET sys@cdb1
RMAN> CONNECT AUXILIARY sys@cdb_test
RMAN> DUPLICATE TARGET DATABASE TO cdbtest FROM ACTIVE DATABASE;    -- whole CDB
-- only PDB1 (and root)
RMAN> DUPLICATE TARGET DATABASE TO cdbtest PLUGGABLE DATABASE pdb1
      FROM ACTIVE DATABASE;

```

Listing 5.9: RMAN duplicate (outline)

Note: Duplicate requires an auxiliary instance started in NOMOUNT state, network connectivity and password files. Exact steps vary by version; refer to the Backup and Recovery documentation.

5.6 Managing Security in Multitenant Databases

5.6.1 Common and Local Users and Roles

Feature	Common user / role	Local user / role
Created in	Root container	A specific PDB
Name	Must start with C## (or c##)	Any valid name; must not start with C##
Exists in	Root and every PDB (identity)	Only that PDB
Privileges	Granted in root, in one PDB or in all (CONTAINER=ALL)	Only within that PDB

```

-- in CDB$ROOT
CREATE USER c##dba_asst IDENTIFIED BY "Asst#123" CONTAINER = ALL;
GRANT CREATE SESSION TO c##dba_asst CONTAINER = ALL;    -- allowed in every PDB
GRANT CREATE SESSION TO c##dba_asst CONTAINER = CURRENT; -- root only

-- in a PDB
ALTER SESSION SET CONTAINER = pdb1;
CREATE USER app_user IDENTIFIED BY "App#User1";        -- local user
GRANT CREATE SESSION, CREATE TABLE TO app_user;

SELECT username, common, con_id FROM cdb_users WHERE username LIKE 'C##%';

```

Listing 5.10: Common and local users

5.6.2 Other Security Features

- **Isolation:** a local PDB user cannot see or access data in other PDBs.
- **Lockdown profiles** (12.2 and later) restrict what actions or features are available inside a PDB (for example disabling `ALTER SYSTEM` for PDB admins).
- **Encryption (TDE):** in *united* mode the root and all PDBs share one keystore managed from the root; in *isolated* mode (18c and later) a PDB manages its own keystore.
- **Auditing:** unified audit policies can be common (created in root) or local to a PDB.
- **Least privilege:** use local PDB administrators for PDB work; use common users only for CDB-wide duties and avoid granting `DBA` commonly.

Key Takeaways

- ✓ A **CDB** has one instance and contains **CDB\$ROOT**, **PDB\$SEED** and user **PDBs**; PDBs share memory, processes, redo and control files.
- ✓ Benefits: consolidation, fast provisioning by cloning, one-time patching, and portability by unplug/plug.
- ✓ Create a PDB: from seed, by cloning, or plug in an XML-described PDB; open it with `ALTER PLUGGABLE DATABASE ... OPEN` and use `SAVE STATE`.
- ✓ Switch containers with `ALTER SESSION SET CONTAINER`; applications connect to the PDB's service.
- ✓ RMAN can back up the CDB, root or individual PDBs, and can restore/recover a PDB while others stay open; `DUPLICATE` copies databases.
- ✓ **Common users** (C##) exist across containers; **local users** exist only in one PDB. Lockdown profiles and TDE add further protection.

Reminder: these notes may contain errors — cross-check with the official syllabus, your textbook and Oracle documentation.

— End of Unit 5 —