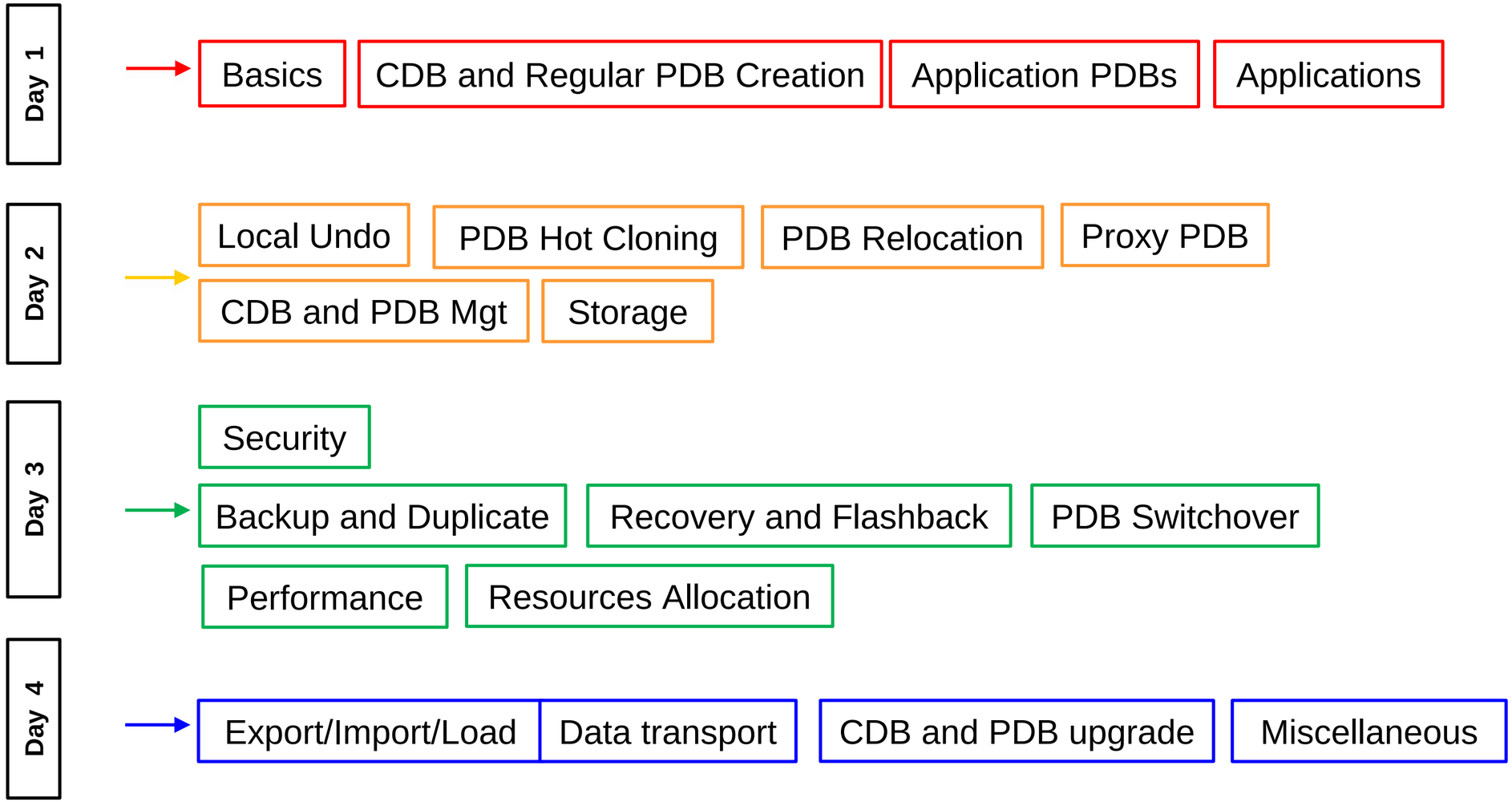
CDB Basics

Oracle Database 19c Multitenant Architecture



Objectives

After completing this lesson, you should be able to:

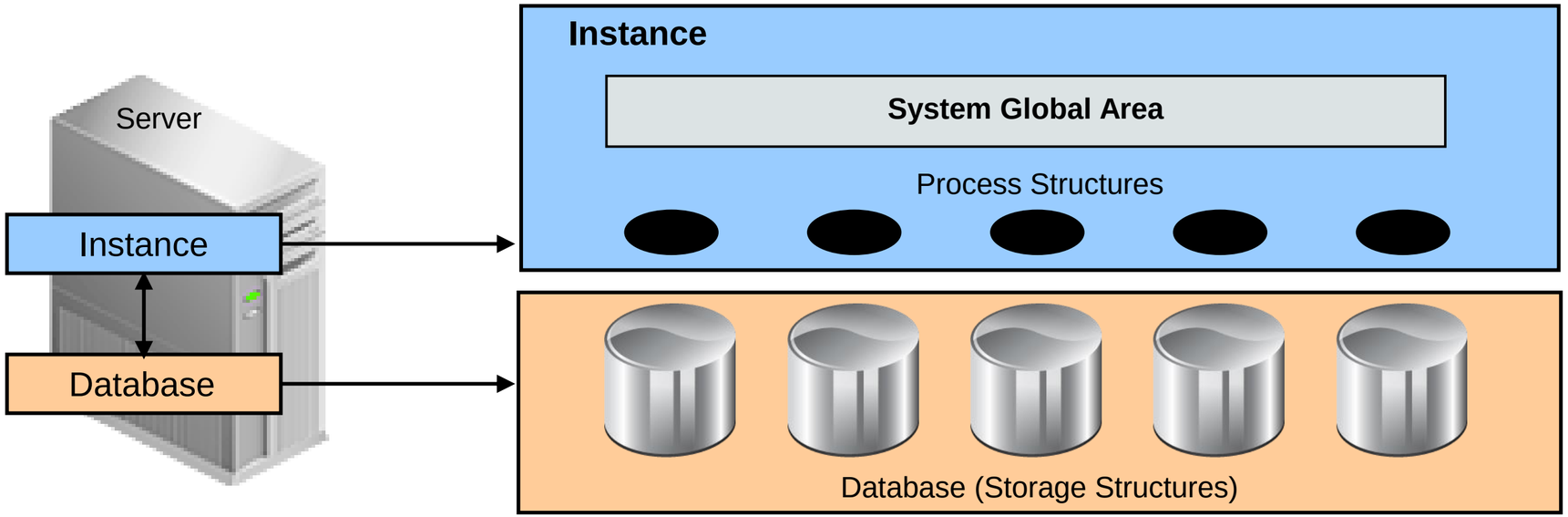
- Describe the multitenant architecture
- Describe the CDB root and pluggable database containers
- Differentiate the CDB root from a pluggable database
- Explain the terminology of commonality
- List impacts in various areas

Challenges

Many Oracle customers have large numbers of “departmental” applications built on Oracle RDBMS that:

- Do NOT use a significant percentage of the hardware on which they are deployed
- Have instance and storage overhead preventing large numbers of “departmental” databases from being placed on the same physical and storage server
- Are NOT sufficiently complex to require 100 percent of the attention of a full-time administrator
- Do require significant time to patch or upgrade all applications

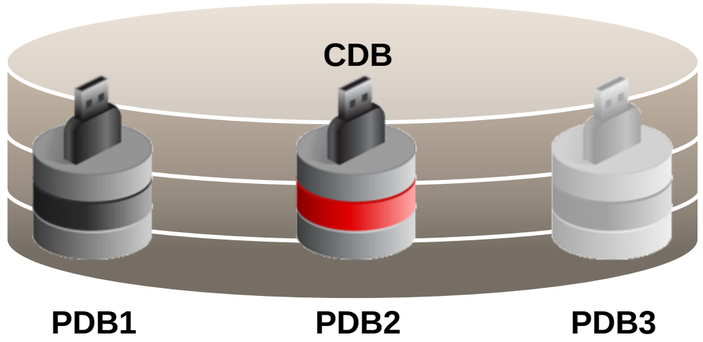
Non-CDB Architecture



- Multiple non-CDBs share nothing:
  - Too many background processes
  - High shared/process memory
  - Many copies of Oracle metadata

Multitenant Architecture: Benefits

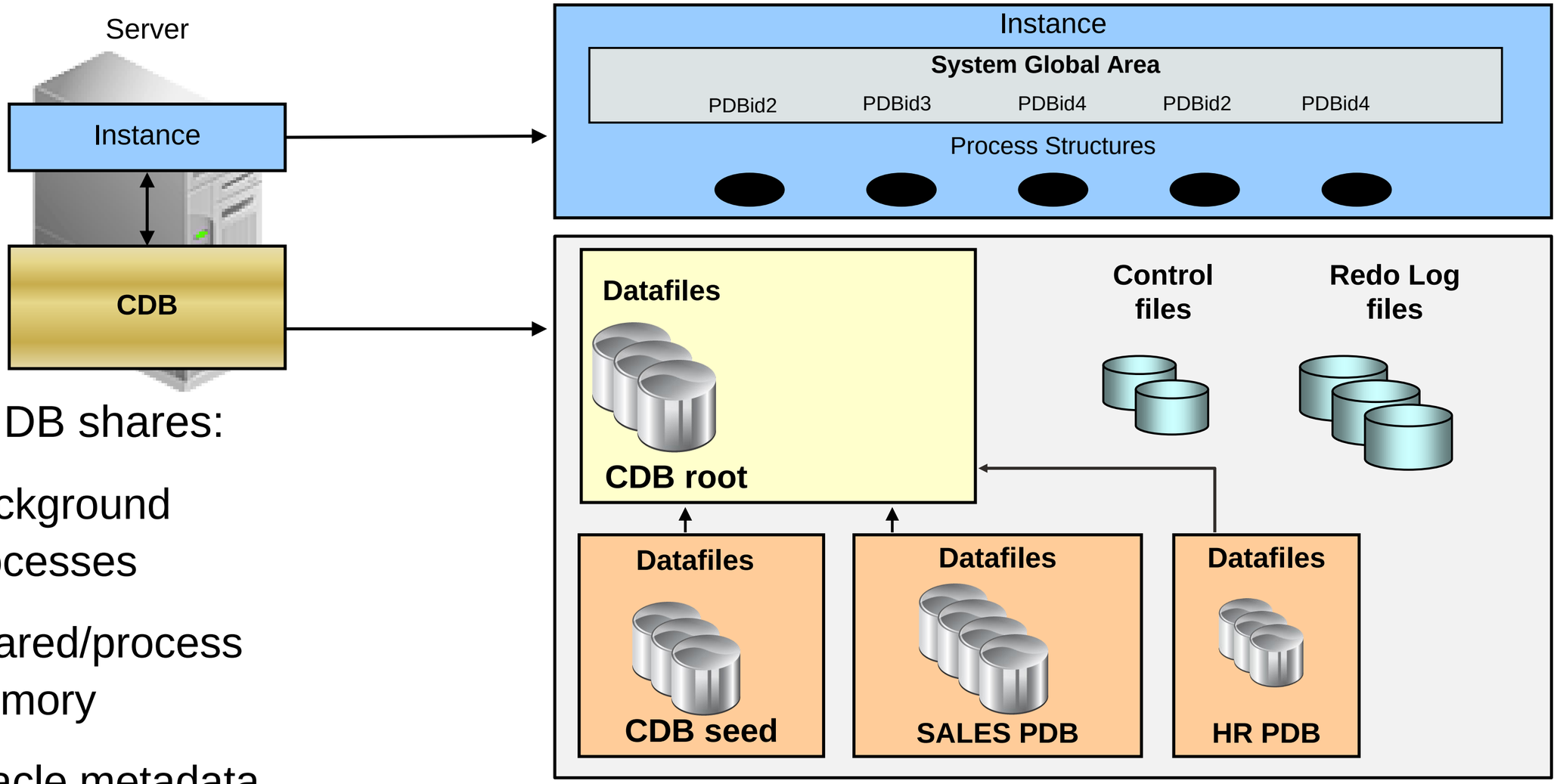
- Operates **multiple databases in a centrally managed platform** at lower costs:
  - Less instance overhead
  - Less storage cost
- Reduces DBA resources costs and maintains security
  - No application changes
  - **Fast and easy provisioning**
    - **Time saving for patching and upgrade**
    - **Separation of duties** between:
      - Different application administrators
      - Application administrators and DBA
      - Users within application
- **Provides isolation**



Other Benefits of Multitenant Architecture

- Ensures **full backward-compatibility** with non-CDBs
- Fully operates with Oracle Real Application Cluster (Oracle RAC) and Data Guard
- Is supported by Oracle Enterprise Manager
- Is integrated with Resource Manager
- Allows central management and administration of multiple databases
  - Backups or disaster recovery
  - Patching and upgrades

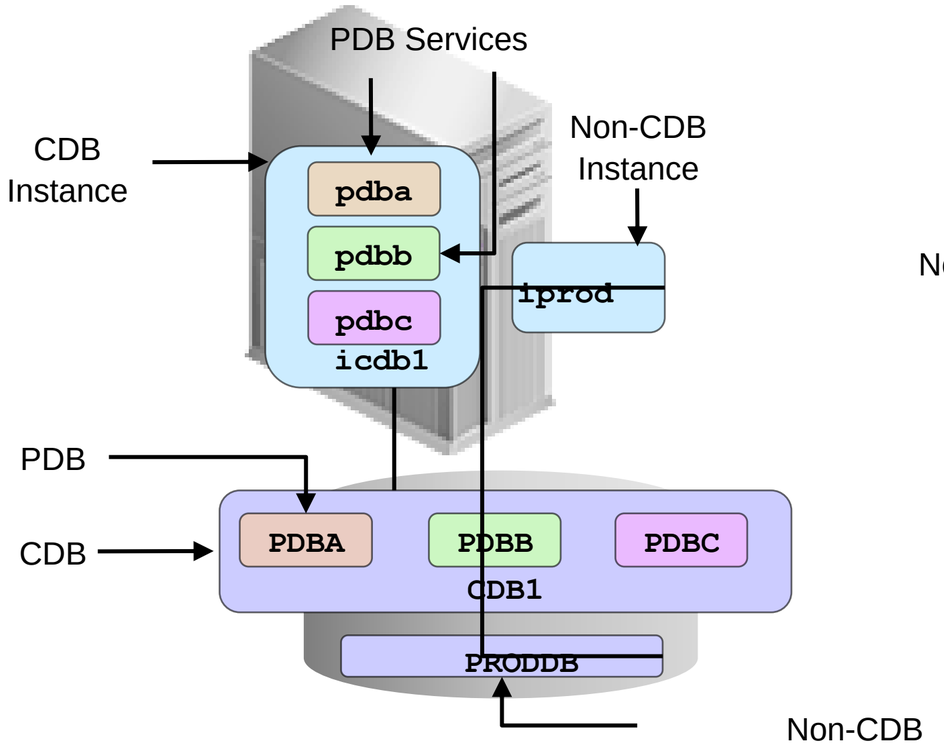
Oracle Multitenant Container Database

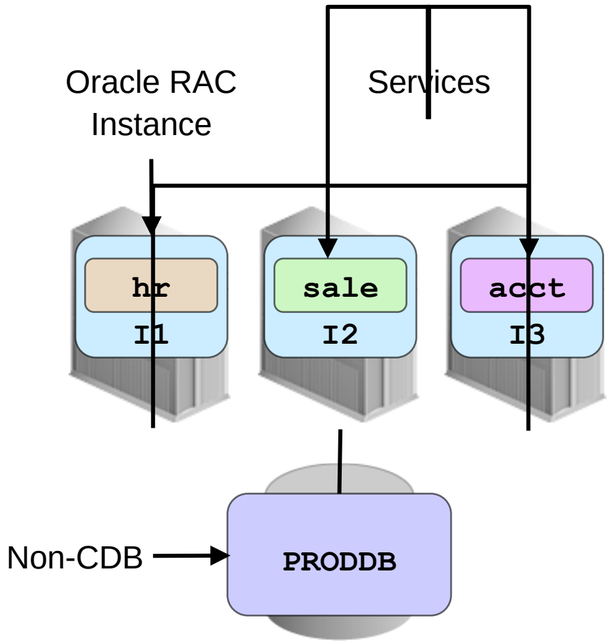


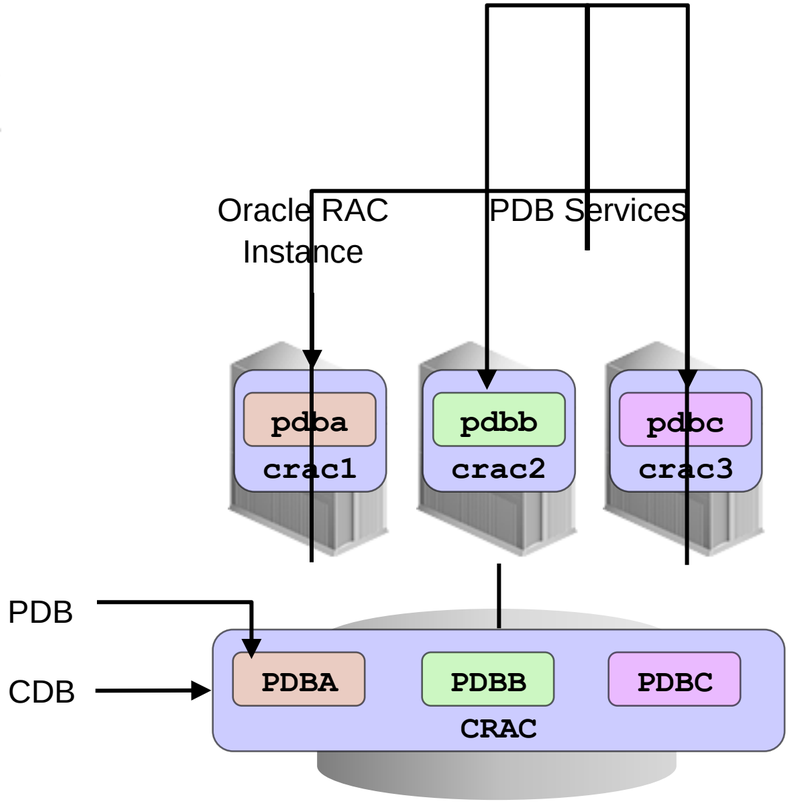
Single DB shares:

- Background processes
- Shared/process memory
- Oracle metadata

Configurations

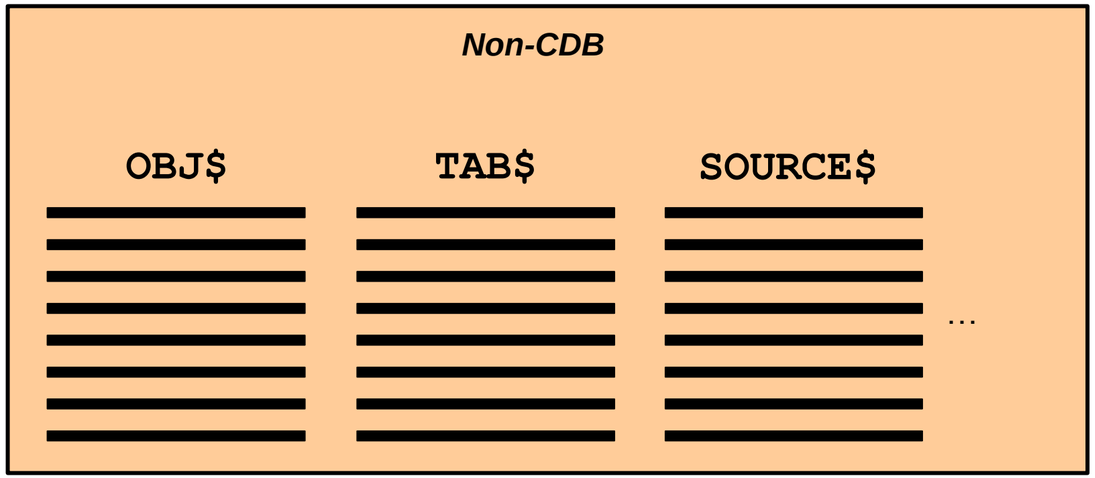






Database Objects in a Non-CDB

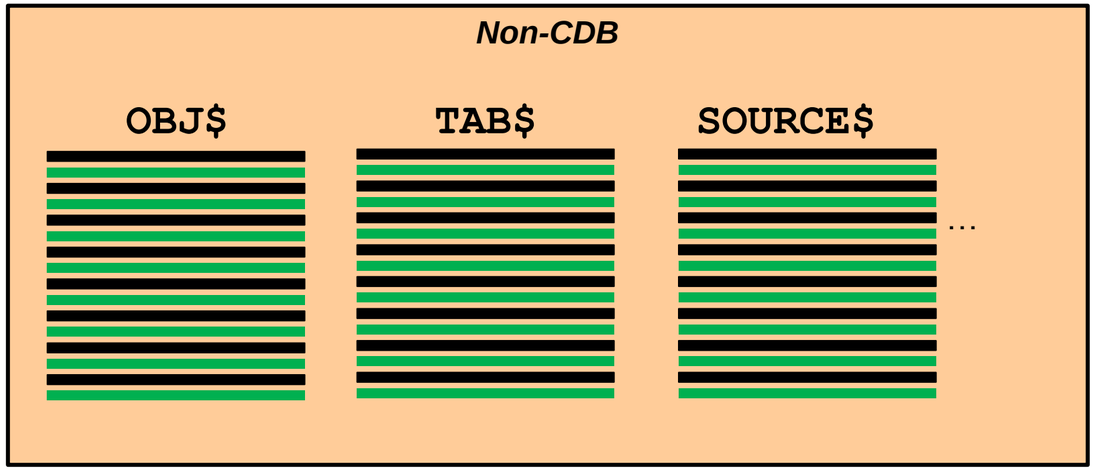
After the initial database creation, the only objects are Oracle-supplied objects.



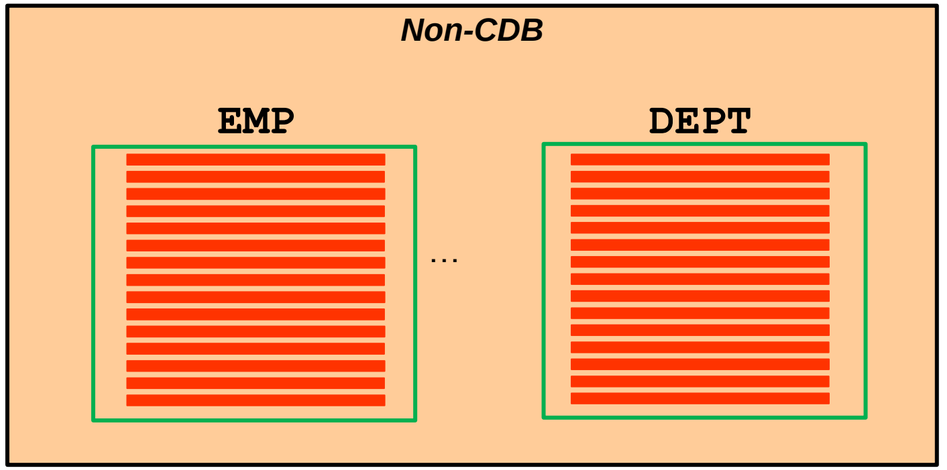
Oracle System data

User-Added Objects to a Non-CDB

In a non-CDB, user data is added: The metadata is mixed with the Oracle-supplied data in the data dictionary.

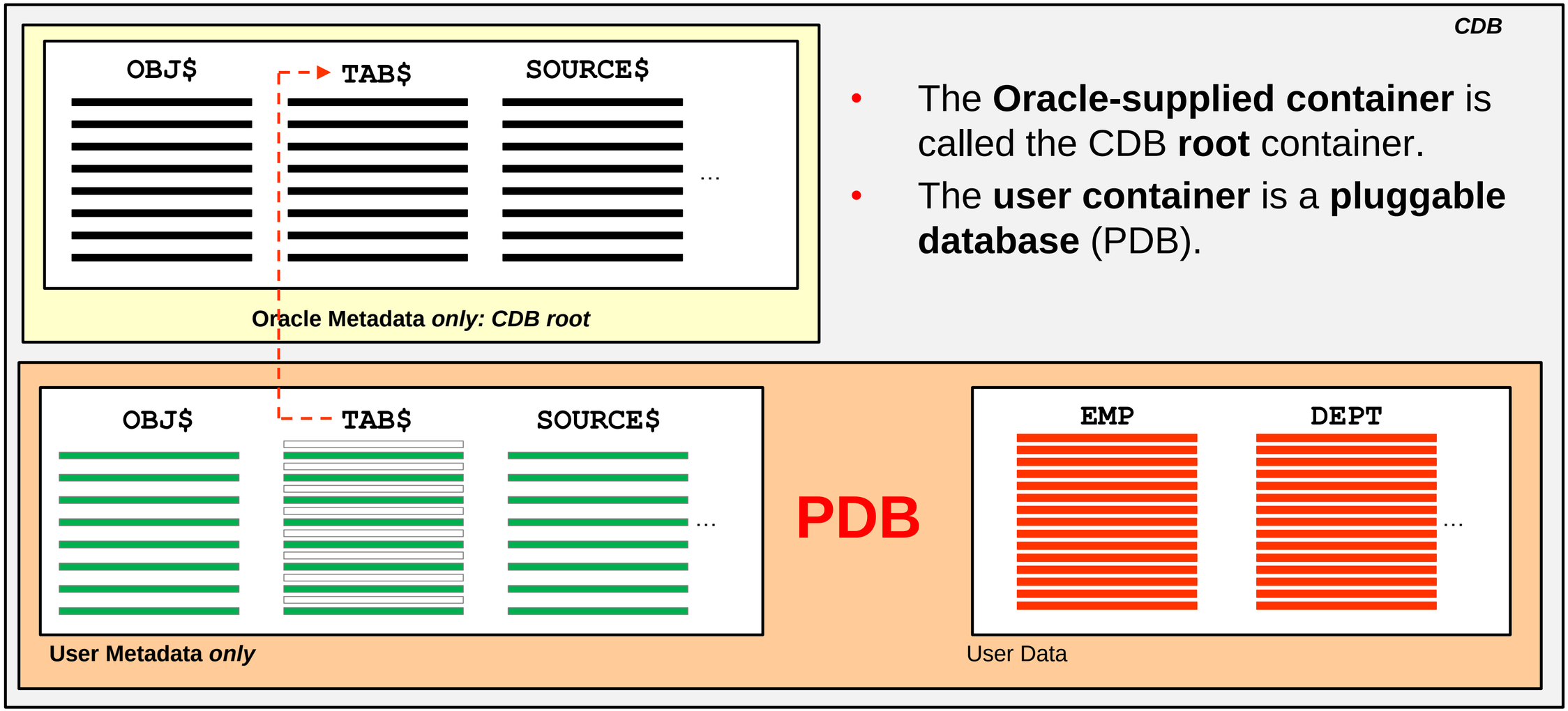


Oracle System data mixed with user metadata



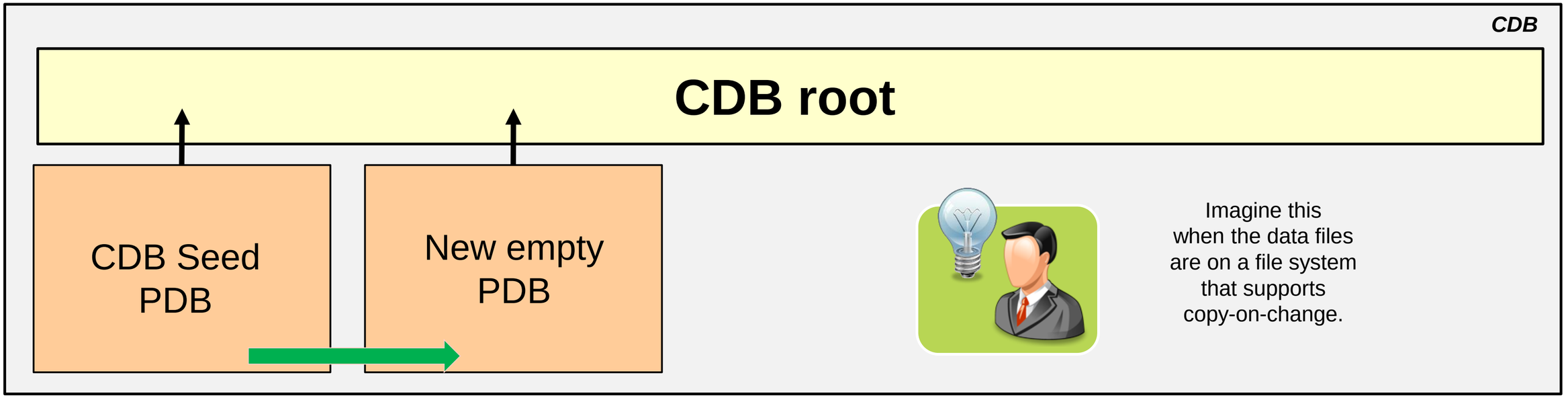
User data

SYSTEM Objects in the USER Container



- The **Oracle-supplied container** is called the CDB **root** container.
- The **user container** is a **pluggable database** (PDB).

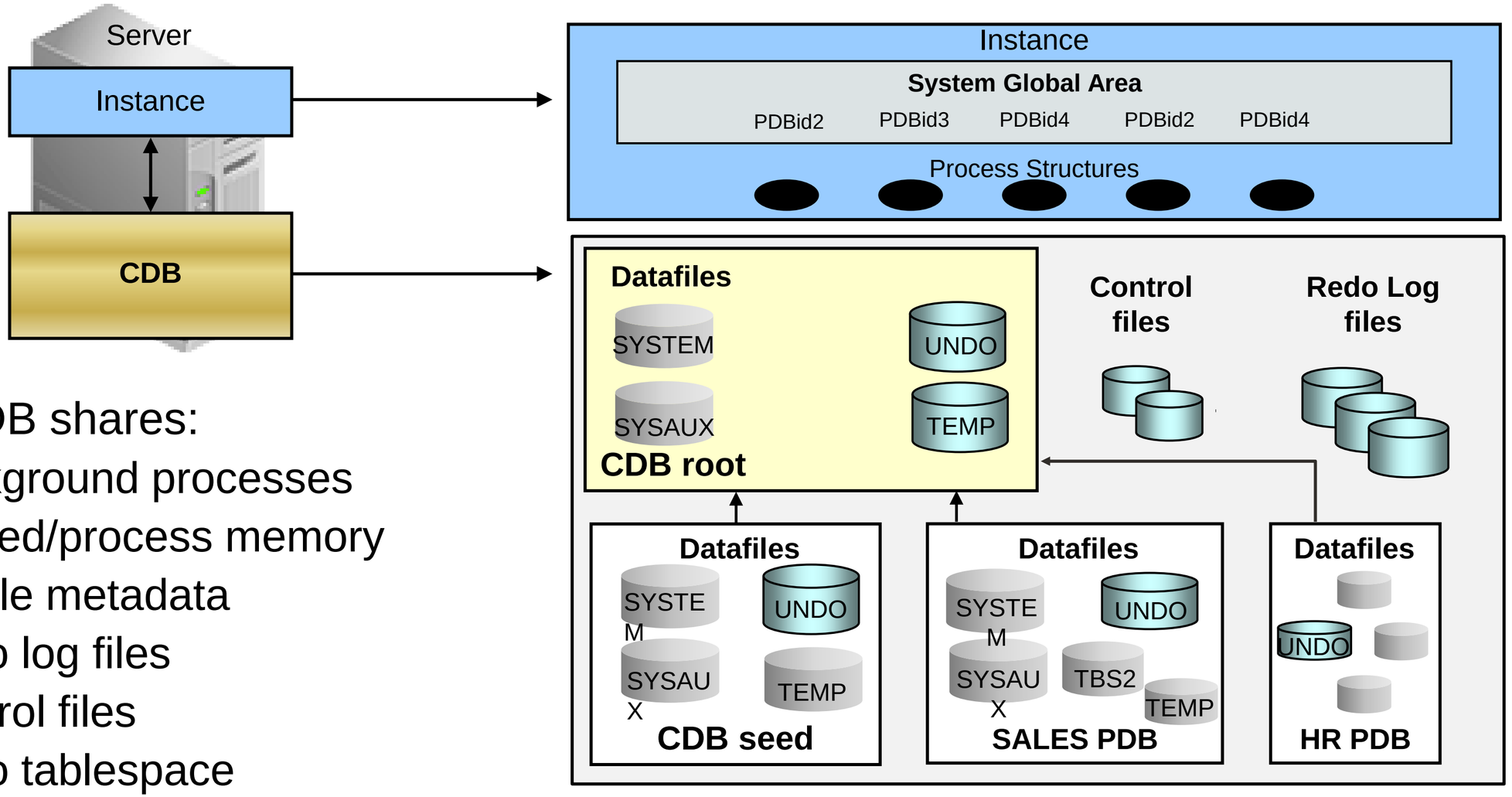
Provisioning a Pluggable Database



Different methods:

- **Create** new PDB from **CDB seed** pluggable database.
- **Plug** or **clone** a non-CDB as a PDB into a CDB.
- **Clone** or **relocate** a PDB from another PDB into the same or another CDB.
- **Plug an unplugged PDB** into the same or another CDB.
- **Create** a PDB as a **proxy** PDB to access a PDB in a remote CDB.

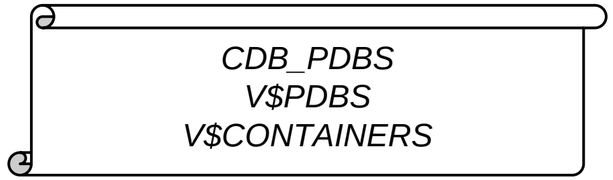
Multitenant Container Database Architecture



Single DB shares:

- Background processes
- Shared/process memory
- Oracle metadata
- Redo log files
- Control files
- Undo tablespace

Containers



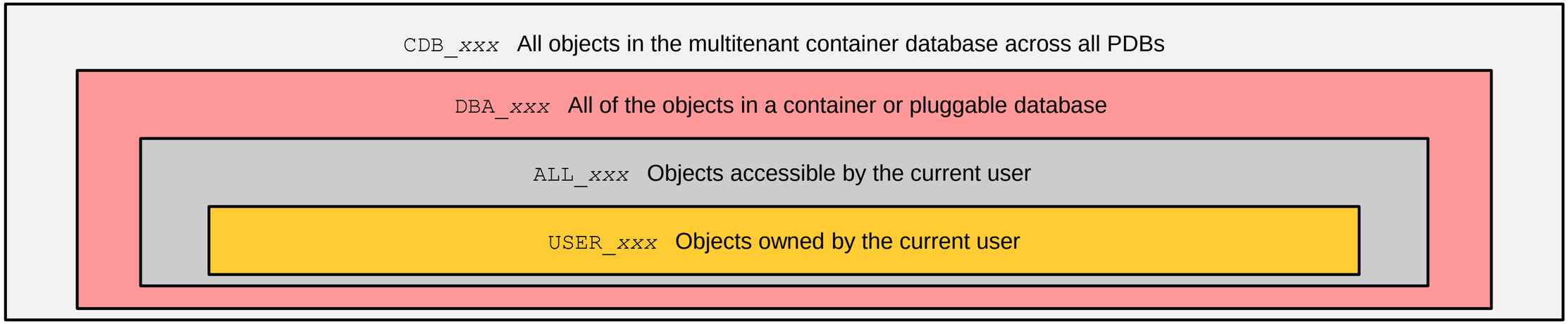
Types of containers in `V$CONTAINERS`:

- **CDB root**
  - The first mandatory container created at CDB creation
  - Oracle system–supplied common objects and metadata
  - Oracle system–supplied common users and roles
- **Pluggable database** (PDBs)
  - Tablespaces (permanent and temporary)
  - Schemas / Objects / Privileges
  - Created / cloned / unplugged / plugged / proxied
  - Particular PDB: CDB seed (`PDB$SEED`) used for fast provisioning of a new PDB

Tools

	SQL*Plus	OUI	DBCA	EM Cloud Control	EM Database Express	SQL Developer	DBUA
Create a new CDB or PDB	Yes	Yes	Yes	Yes (PDB only)	Yes (PDB only)	Yes (PDB only)	
Explore CDB instance, architecture, and PDBs	Yes			Yes	Yes	Yes	
Upgrade a 12c CDB to 18c CDB				Yes			Yes

Data Dictionary and Dynamic Views



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

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

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

Terminology

- DBA, CDB_DBA, and PDB_DBA (roles assigned to administrators at different levels)
- Common vs Local:
  - Users
  - Privileges / Roles
  - Objects
  - Profiles
- CDB vs PDB level:
  - CDB Resource Manager plan vs PDB RM plan
  - Unified audit policies at CDB or PDB level
  - Encryption master keys at CDB and PDB level
  - Database Vault realms and command rules at CDB or PDB level
  - XStream at CDB or PDB level

Impacts

- Define a character set for the CDB and per PDB.
- Define PDB initialization parameters in a single SPFILE.
- Do not use PDB-qualified database object names. Instead use database links.

```
SQL> SELECT * FROM HR:apps.tab1;        SQL> SELECT * FROM apps.tab1@HR;
```

- Implement subset standbys at the PDB level.
- Configure Oracle Database Vault per PDB and on common objects.
- Create one TDE master encryption key per PDB to encrypt PDB data.
- Configure unified audit at CDB and PDB level.
- Benefit from Heat Maps and Automatic Data Optimization.
- Use Logminer for objects at all levels.
- Configure replication at PDB and application level with XStream and Oracle GoldenGate.

Summary

In this lesson, you should have learned how to:

- Describe the multitenant architecture
- Describe the CDB root and pluggable database containers
- Differentiate the CDB root from a pluggable database
- Explain the concept of commonality
- List impacts in various areas

Practices Environment - 2

Pre-created databases and instances with their respective PDBs:

- Data files in `/u02/app/oracle/oradata/`
- CDB root data files in `/u02/app/oracle/oradata/<db_name>`
- PDB data files in `/u02/app/oracle/oradata/<db_name>/<pdb_name>`
- Control files in `/u02/app/oracle/oradata/<db_name>` and `/u03/app/oracle/fast_recovery_area/<db_name>`
- All redo log files in `/u04/app/oracle/redo/<db_name>`
- All backup files in `/u03/app/oracle/fast_recovery_area/<db_name>`

Practice 1: Overview

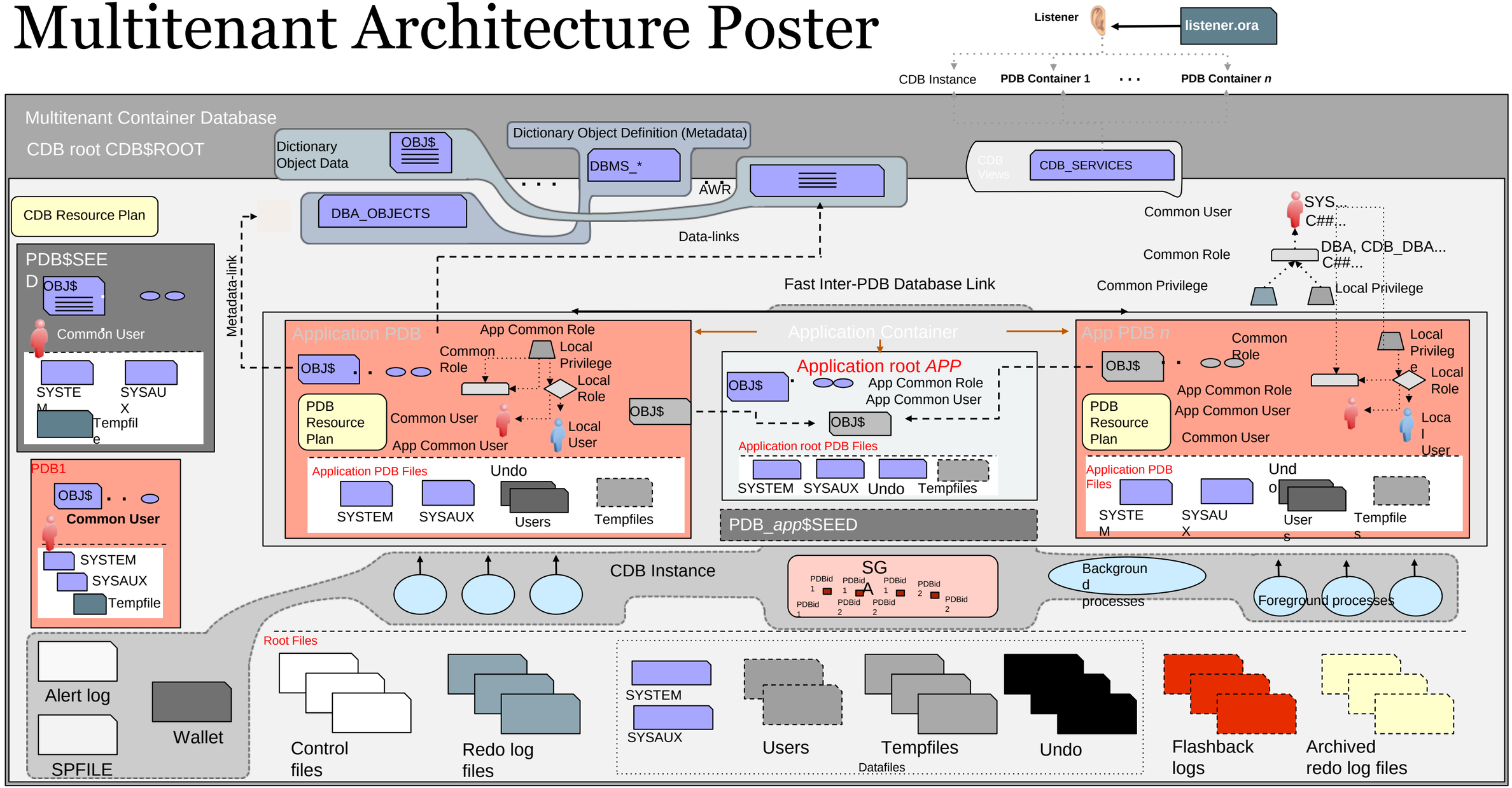
- 1-1: Discovering practices environment
- 1-2: Setting and Validating OEMCC Named Credentials
- 1-3: Exploring CDB and PDB using Enterprise Manager Cloud Control
- 1-4: Using Enterprise Manager Express

Note:

In most of the practices, you will have to execute setup and cleanup shell scripts and SQL scripts. The scripts may generate false errors in the following occurrences:

- A tablespace dropped because the tablespace was not created
- A pluggable database dropped because the pluggable database was not created
- A directory creation dropped because the directory already exists
  Do not pay attention to the errors.

Multitenant Architecture Poster



CDB and Regular PDBs

Objectives

After completing this lesson, you should be able to:

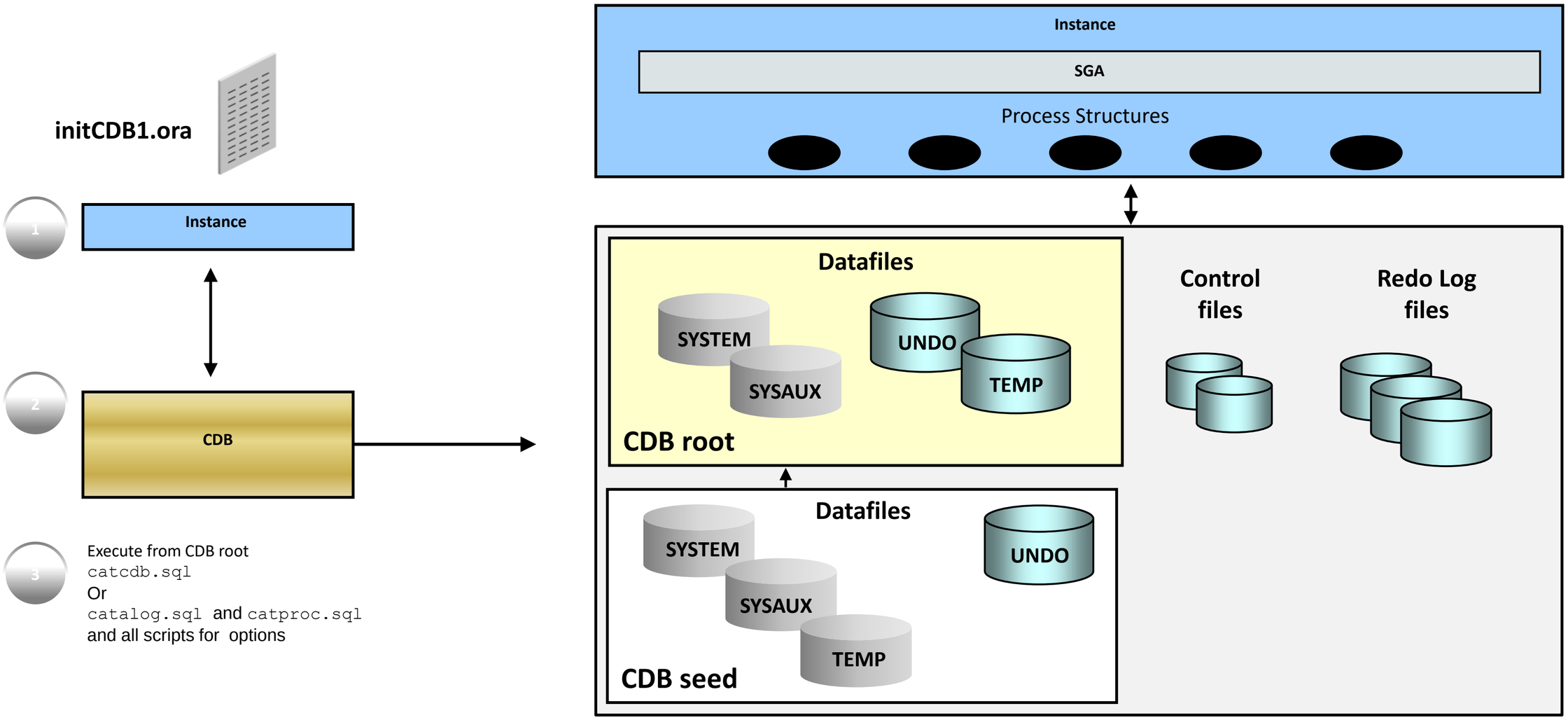
- Configure and create a CDB
- Create a new PDB from the CDB seed
- Explore the instance
- Explore the structure of PDBs
- Explore the ADR

Goals

Create a multitenant container database:

- To consolidate many pre-12.1, 12c, 18c, and 19c 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

Creating a CDB



Creating a CDB: Using SQL*Plus

1. Start up the instance :
    a. Set `ORACLE_SID=CDB1`.
    b. Create the `initCDB1.ora` file and set parameters:
        - `CONTROL_FILES` to CDB control file names
        - `DB_NAME` to a CDB name
        - `ENABLE_PLUGGABLE_DATABASE` to `TRUE`

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

2. Create the database:

```
SQL> CREATE DATABASE cdb1 ENABLE PLUGGABLE DATABASE …
            SEED FILE_NAME_CONVERT = ('/oracle/dbs','/oracle/seed');
```

➔ `CDB$ROOT` + `PDB$SEED` created

3. Execute the $`ORACLE_HOME/rdbms/admin/catcdb.sql` SQL script.

Clause: SEED FILE_NAME_CONVERT

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

Clause: ENABLE PLUGGABLE DATABASE

Without **SEED FILE_NAME_CONVERT**:

- OMF: **DB_CREATE_FILE_DEST=**'/u02/app/oradata'

```
SQL> CONNECT / AS SYSDBA
SQL> STARTUP NOMOUNT
SQL> CREATE DATABASE cdb2
      USER SYS IDENTIFIED BY p1 USER SYSTEM IDENTIFIED BY p2
      EXTENT MANAGEMENT LOCAL
      DEFAULT TEMPORARY TABLESPACE temp
      UNDO TABLESPACE undotbs
      DEFAULT TABLESPACE users
      ENABLE PLUGGABLE DATABASE;
```

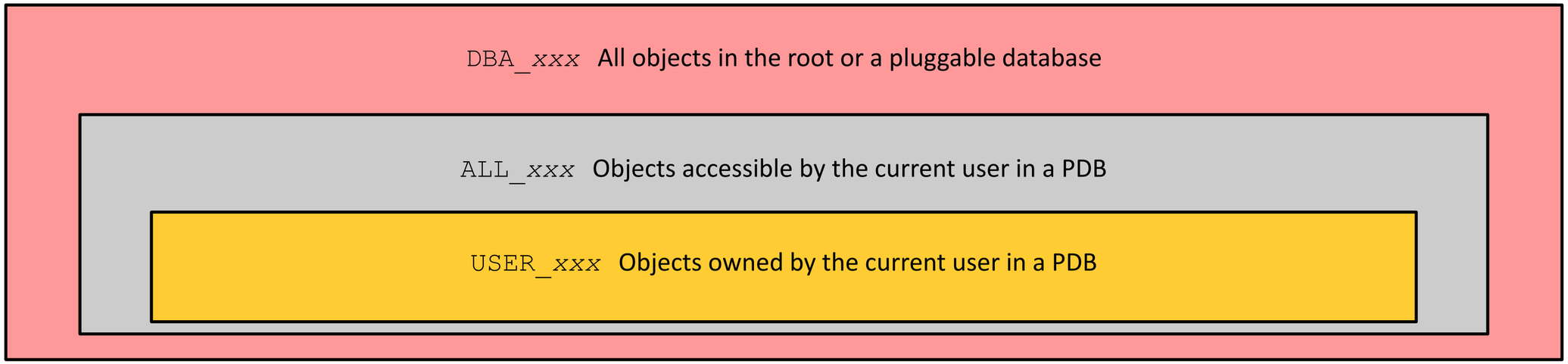
- Or instance parameter: **PDB_FILE_NAME_CONVERT =**
  '/u02/app/oradata/CDB2','/u02/app/oradata/seed'

After CDB Creation: What's New in CDB

A CDB has new characteristics compared to non-CDBs:

- **Two containers:**
  - The CDB **root** (`CDB$ROOT`)
  - The CDB **seed** (`PDB$SEED`)
- **Several services**: One per container
  - Name of CDB root service = name of the CDB (`cdb2`)
  - Maximum number of services: 10000
  - Max nb of services per PDB<= max nb of services in CDB
- **Common users** in CDB root and CDB seed: `SYS`,`SYSTEM` …
- **Common privileges** granted to common users
- **Predefined common roles**
- Tablespaces and data files associated with each container:
  - `SYSTEM`, `SYSAUX`, and `UNDO`

Data Dictionary Views: DBA_xxx

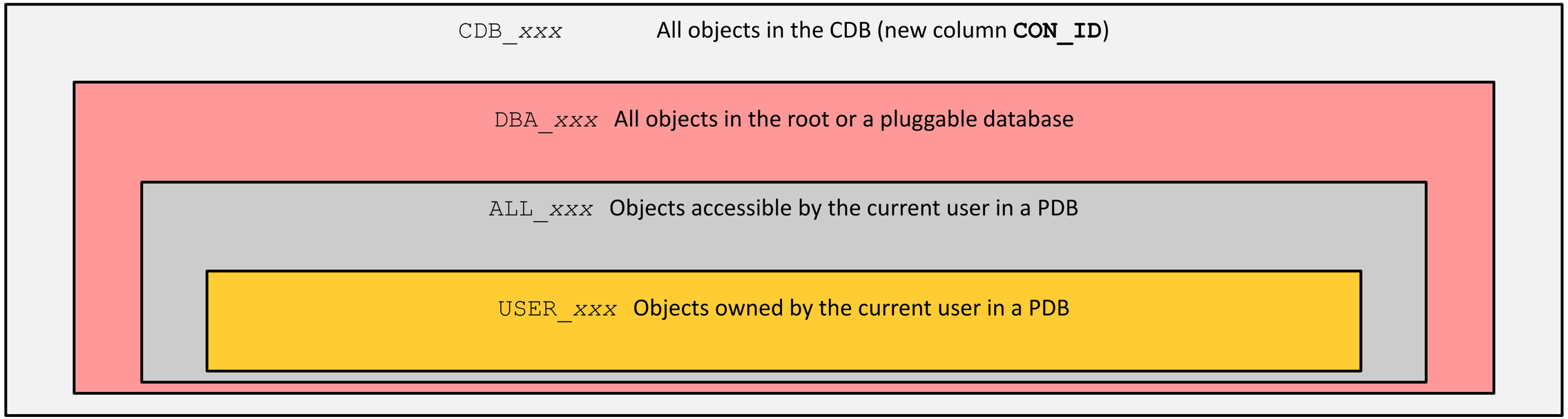


DBA dictionary views providing information within PDB:

```
SQL> SELECT table_name FROM dict 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 dictionary views provide information across PDBs:

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

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

Data Dictionary Views: Examples

- Comparisons:

1
2

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

3
4

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

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

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

NAME             OPEN_MODE
---------------- ----------
PDB$SEED             READ ONLY
PDB1                 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   ------------>   CDB root
free              1
xcur              1
xcur              2   ------------>   CDB seed
cr                3   ------------>   PDB1 PDB
```

```
SQL> select OBJECT_ID, ORACLE_USERNAME, LOCKED_MODE, CON_ID 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

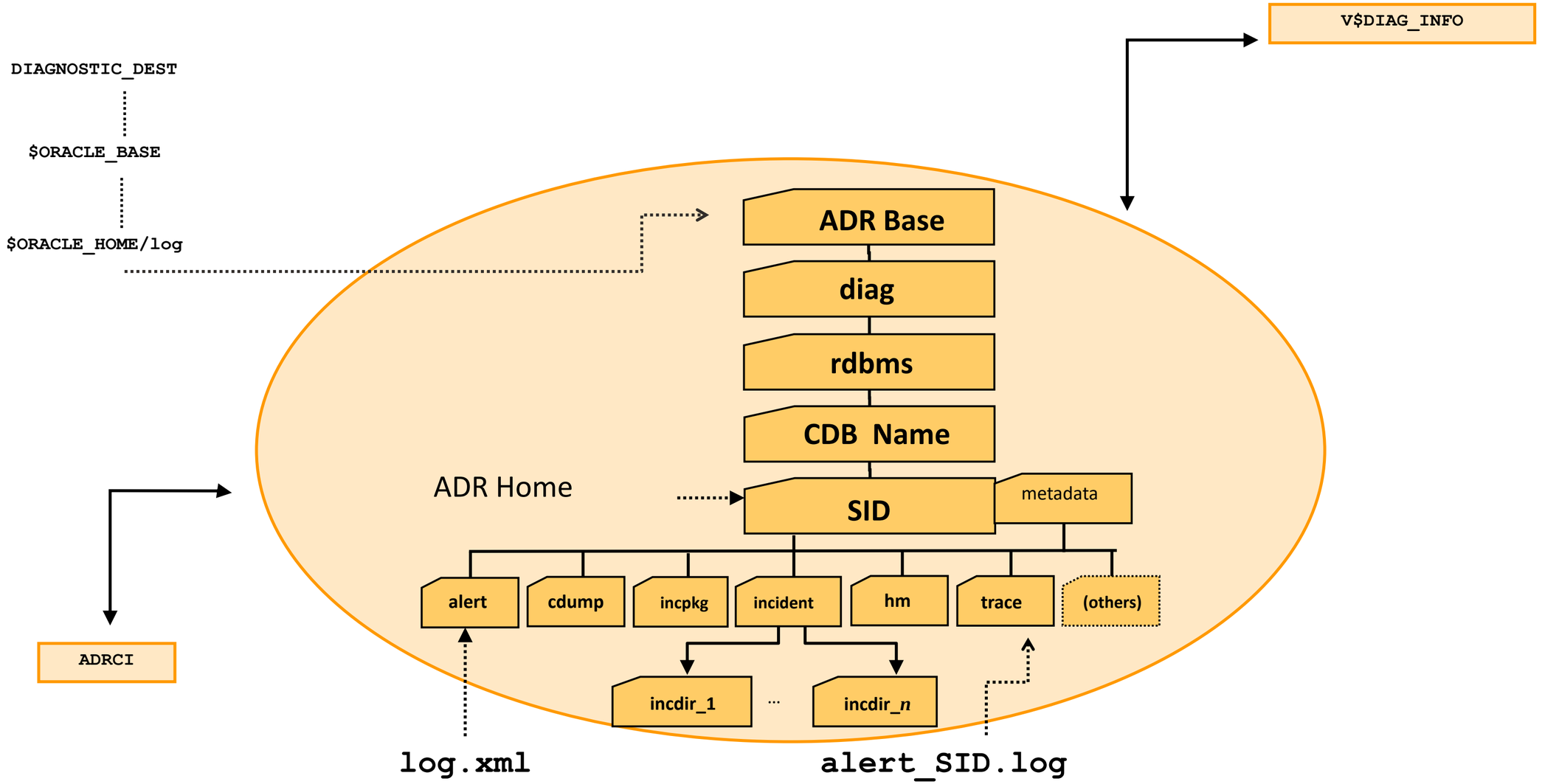
After CDB creation, the CDBA has to:

- Create the SPFILE from the PFILE
- Execute the `$ORACLE_HOME/rdbms/admin/utlrp.sql` script
- Optionally plug non-CDBs and create new PDBs
- Test startup/shutdown procedures
- Automate PDBs opening
- Create backup and recovery procedures

After PDB creation, each PDBA in its own PDB has to:

- Set a default tablespace
- Optionally create additional temporary tablespaces

Automatic Diagnostic Repository



Automatic Diagnostic Repository: alert.log File

The alert_CBD1.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 PDB$SEED AS CLONE USING …
CREATE PLUGGABLE DATABASE pdb1 … ;
ALTER PLUGGABLE DATABASE pdb1 UNPLUG INTO … ;
ALTER PLUGGABLE DATABASE ALL OPEN ;
ALTER PLUGGABLE DATABASE pdb2 CLOSE IMMEDIATE ;
```

Provisioning New Pluggable Databases

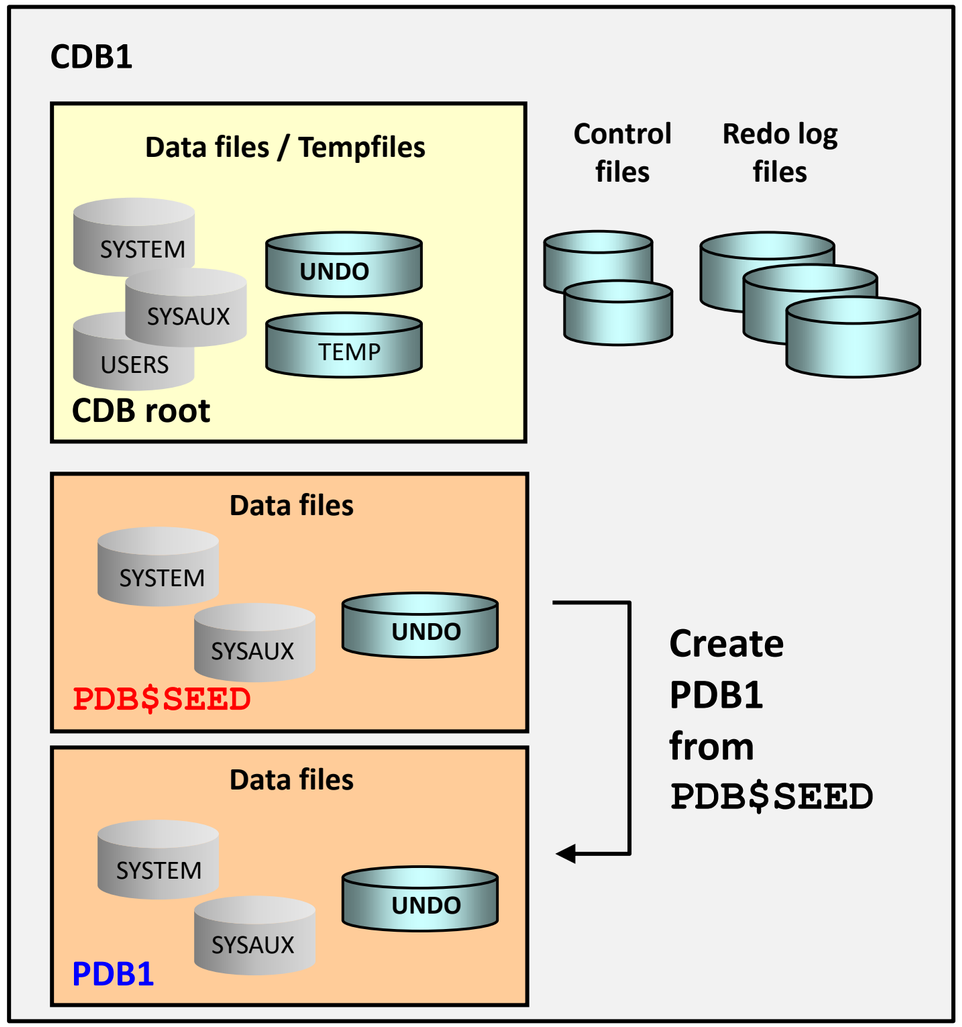
- Create a new PDB from the CDB seed.
- Plug an unplugged PDB into the same CDB or into another CDB.
- Plug a non-CDB in a CDB as a PDB.
- Clone a PDB from another PDB (local or remote CDB, hot or cold).
- Relocate a PDB from a CDB into another CDB.
- Proxy a PDB from another PDB.

Tools

To provision new PDBs, you can use:

- SQL*Plus
- SQL Developer
- Enterprise Manager Cloud Control
- Enterprise Manager Database Express
- Database Configuration Assistant (DBCA)
  - Clone from CDB seed
  - Clone from an existing PDB
  - Plug an unplugged PDB
  - Clone a remote PDB
  - Relocate a PDB from a remote CDB to a local CDB

Create New PDB from PDB$SEED



- Copies the data files from PDB$SEED data files
- Creates tablespaces SYSTEM, SYSAUX, UNDO
- Creates a full catalog including metadata pointing to Oracle- supplied objects
- Creates common users:
  - 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 CDB root as a common user with the `CREATE PLUGGABLE TABASE` system privilege:

```
SQL> CREATE PLUGGABLE DATABASE pdb1
     ADMIN USER admin1 IDENTIFIED BY p1 ROLES=(CONNECT)
     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**:

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

Or

- Use the instance parameter: **PDB_FILE_NAME_CONVERT =** '/u01/app/oradata/CDB1/seed','/u01/app/oradata/CDB1/**pdb1**'

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

Or

- Use the clause in the CREATE PLUGGABLE DATABASE command:
  CREATE_FILE_DEST = '/u01/app/oradata/CDB1/**pdb1**'

Summary

In this lesson, you should have learned how to:

- Configure and create a CDB
- Create a new PDB from the CDB seed
- Explore the instance
- Explore the structure of PDBs
- Explore the ADR

Application PDBs and Application Installation

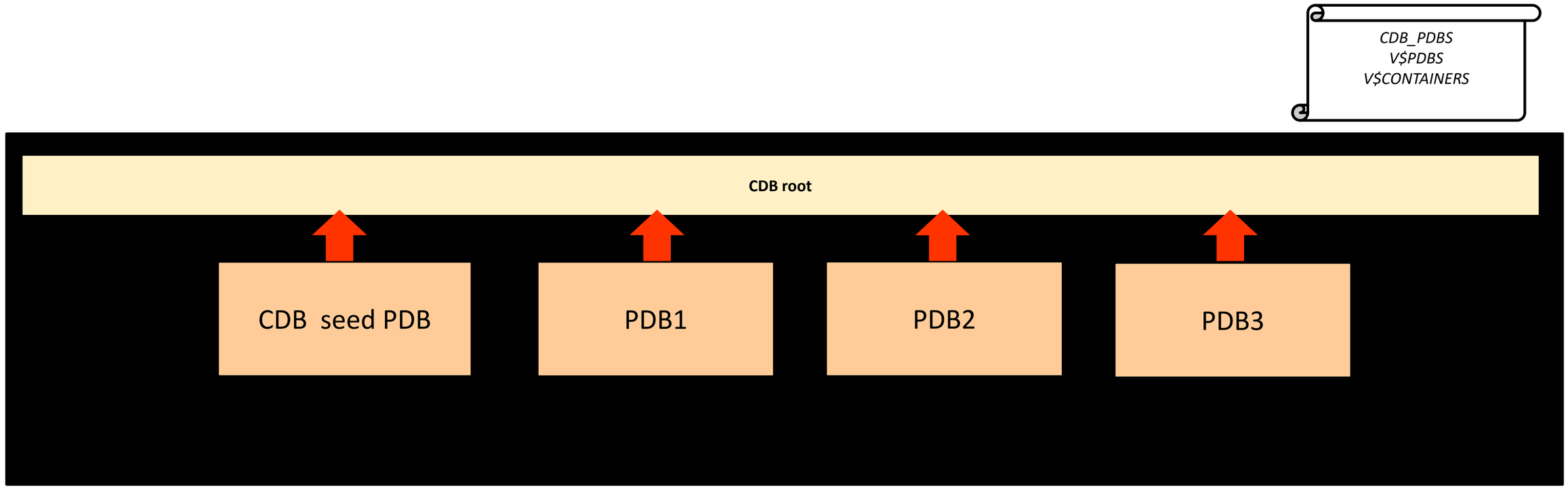
Objectives

After completing this lesson, you should be able to:

- Describe application containers in CDBs
- Explain the purpose of application root and application seed
- Define application PDBs
- Create application PDBs
- Explain application installation on top of application containers
- Install an application
- Upgrade and patch applications
- Describe the commonality concept in application contexts
- Use a dynamic container map
- Describe enhancements in various areas

Regular PDBs

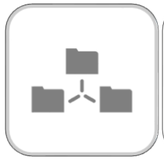
- A regular PDB is a PDB within a CDB, storing data in objects independently of other PDBs.
- A regular PDB can be created from the CDB seed or from another PDB (*cloning* or *unplugging/plugging*).



PDBs and Applications

Applications in regular PDBs need to be upgraded or patched in the same CDB or across many CDBs.

The upgrade script has to be executed in all regular PDBs individually.



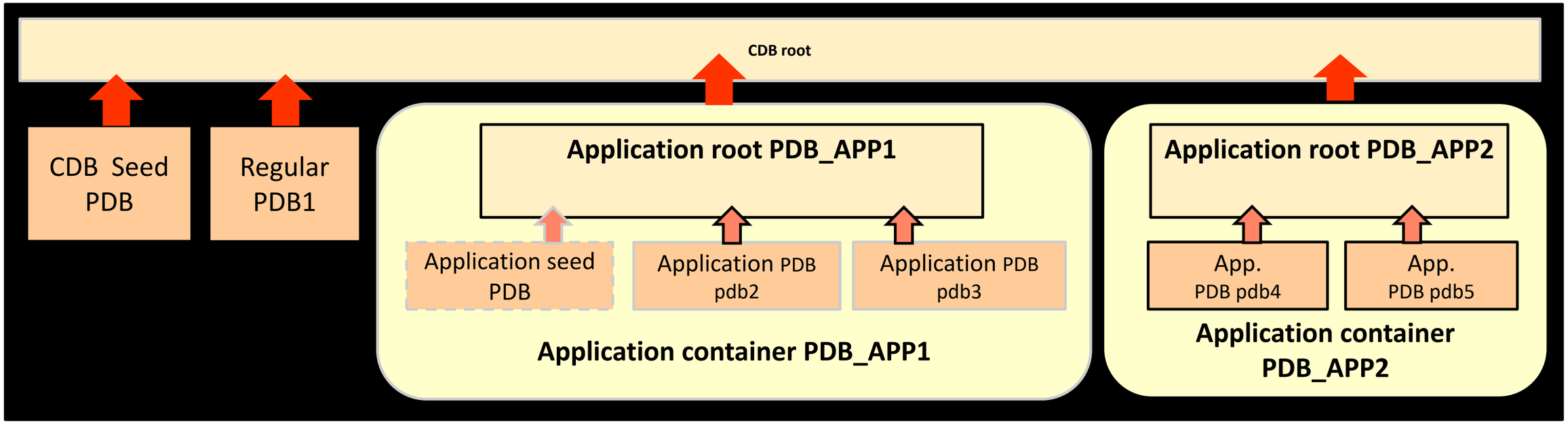
No single master definition of application

Application Containers

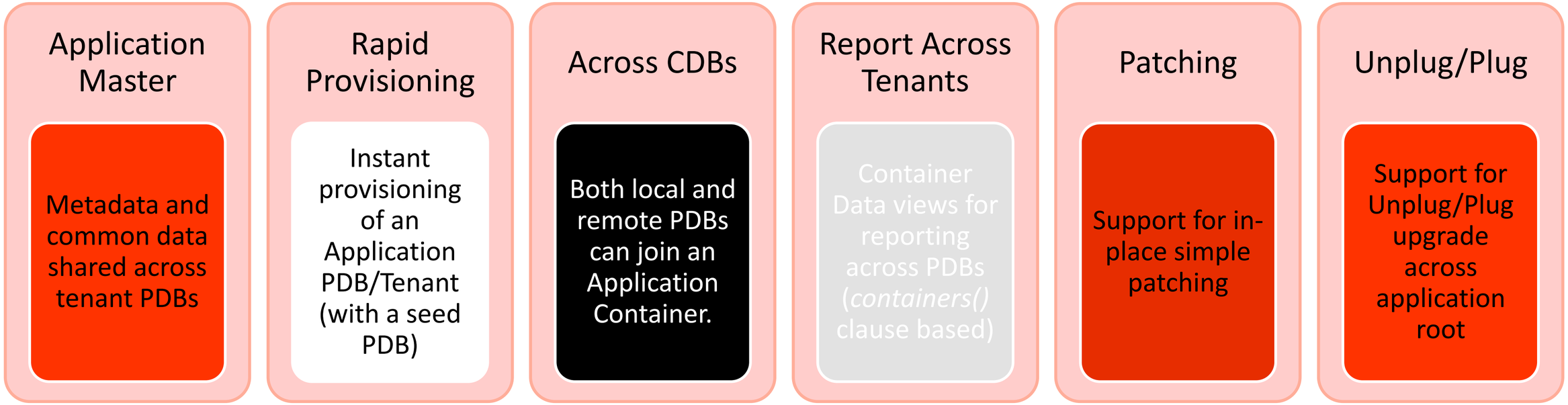
An application container is a collection of PDBs grouped together within a CDB to store data for an application.

- The application root
- An optional application seed
- Application PDBs associated with the application root

Columns in CDB_PDBS
V$PDBS
V$CONTAINERS

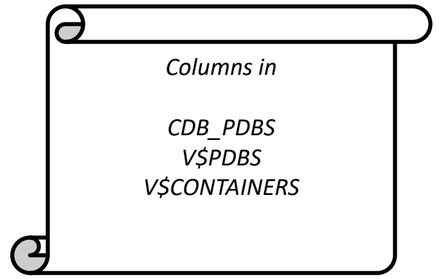


Application Containers: Other Features

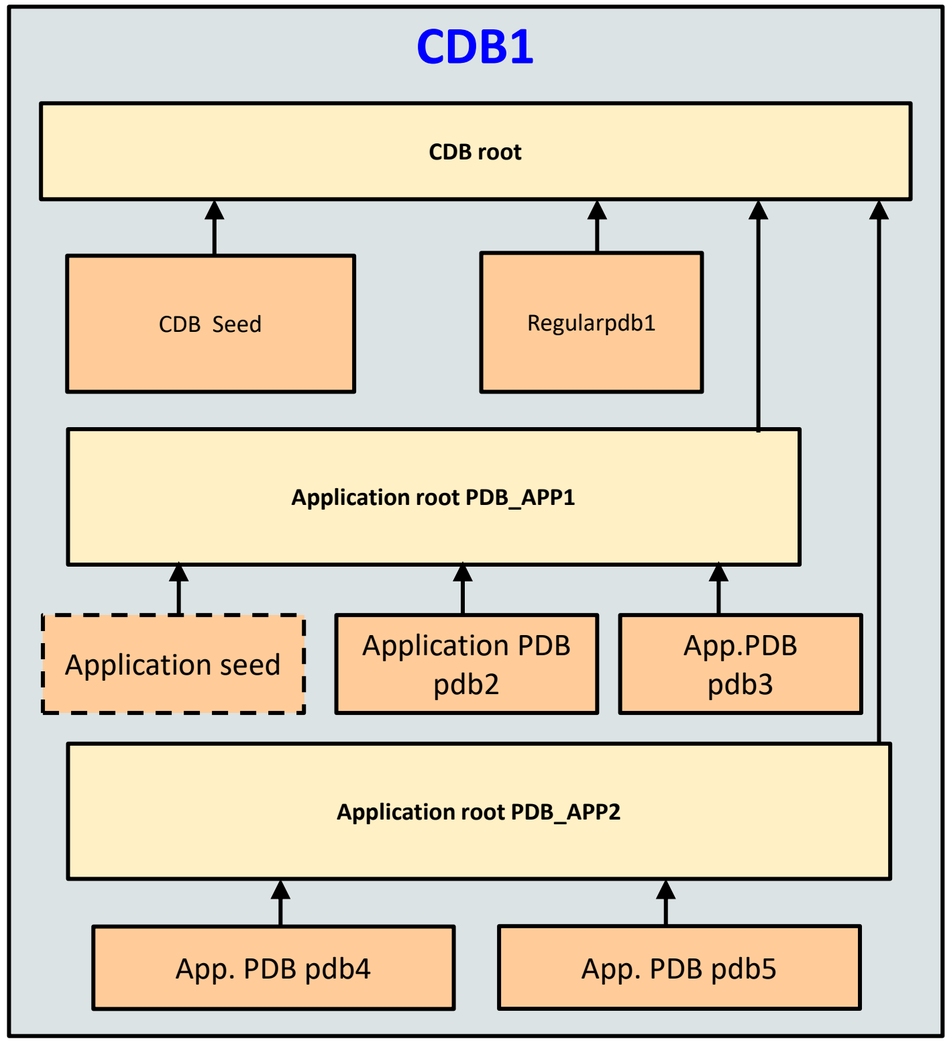


Types of Containers

- The **CDB root container** (`CDB$ROOT`)
  - The first mandatory container created at CDB creation
  - Oracle system–supplied common objects and metadata
  - Oracle system–supplied common users and roles
- **Pluggable database containers** (PDBs)
  - The CDB seed (`PDB$SEED`)
    - The second mandatory container created at CDB creation
    - Oracle system–supplied common entities for new PDBs
  - Regular PDBs
  - Application containers
    - Application root PDB
    - Optional application seed PDB (*`application_container_root_name`*`$SEED`)
    - Application PDBs



Creating Application PDBs



1. Connect to the **CDB1** CDB root.
2. Create the **PDB_APP1** PDB as the application root.

```
SQL> CONNECT / AS SYSDBA
SQL> CREATE PLUGGABLE DATABASE pdb_app
                AS APPLICATION CONTAINER …;
```

3. Connect to the **PDB_APP1** application root.
4. Install the application.
5. Optionally, create the application seed for the application PDBs in the application root.
6. Create the **PDB2** PDB as an application PDB within the **PDB_APP1** application root.
7. Create other application PDBs if required.
8. Synchronize all application PDBs with the application installed if step 5 was not completed.

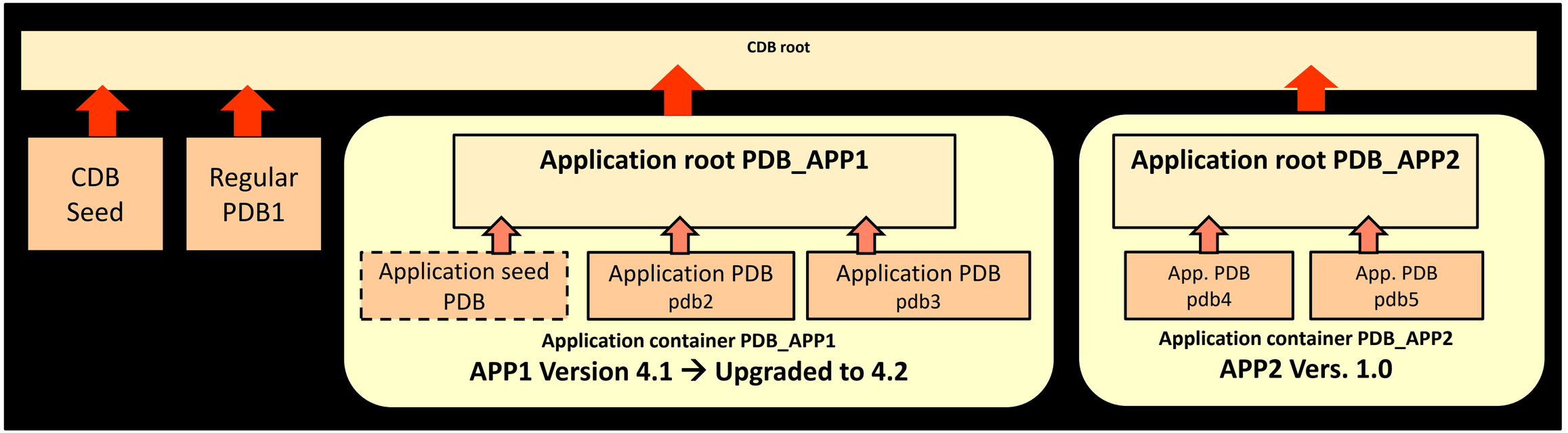
Application Name and Version

An application container can be tagged with:

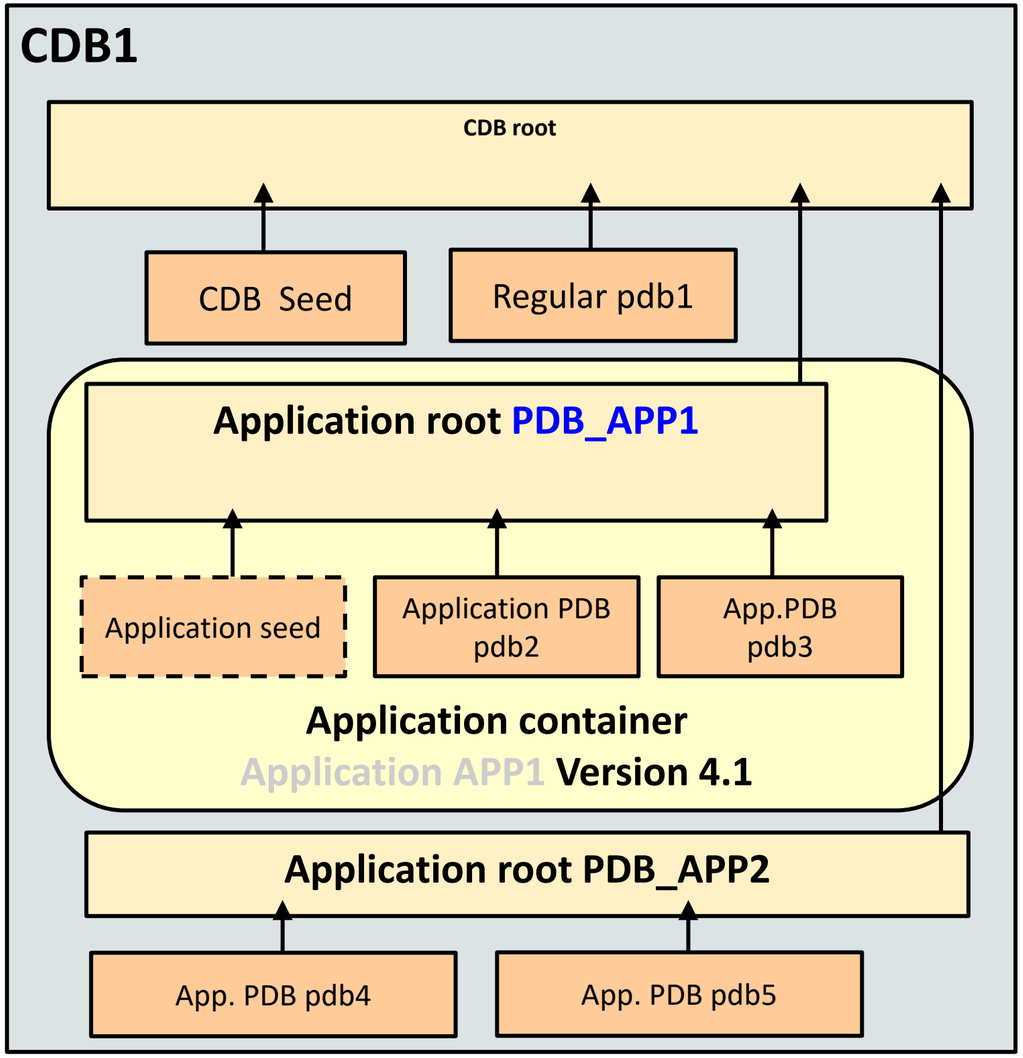
- An application name
- An application version

An application can be patched, upgraded, or uninstalled.

DBA_APPLICATIONS
DBA_APP_VERSIONS
DBA_APP_PATCHES
DBA_APP_ERRORS
DBA_APP_STATEMENTS



Installing Applications



1. Connect to the **PDB_APP1** application root.
2. Assign an application name and version to the new **APP1** application that is being installed.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION app1
             BEGIN INSTALL '4.1';
```

3. Execute the user-defined scripts.

```
SQL> @scripts
```

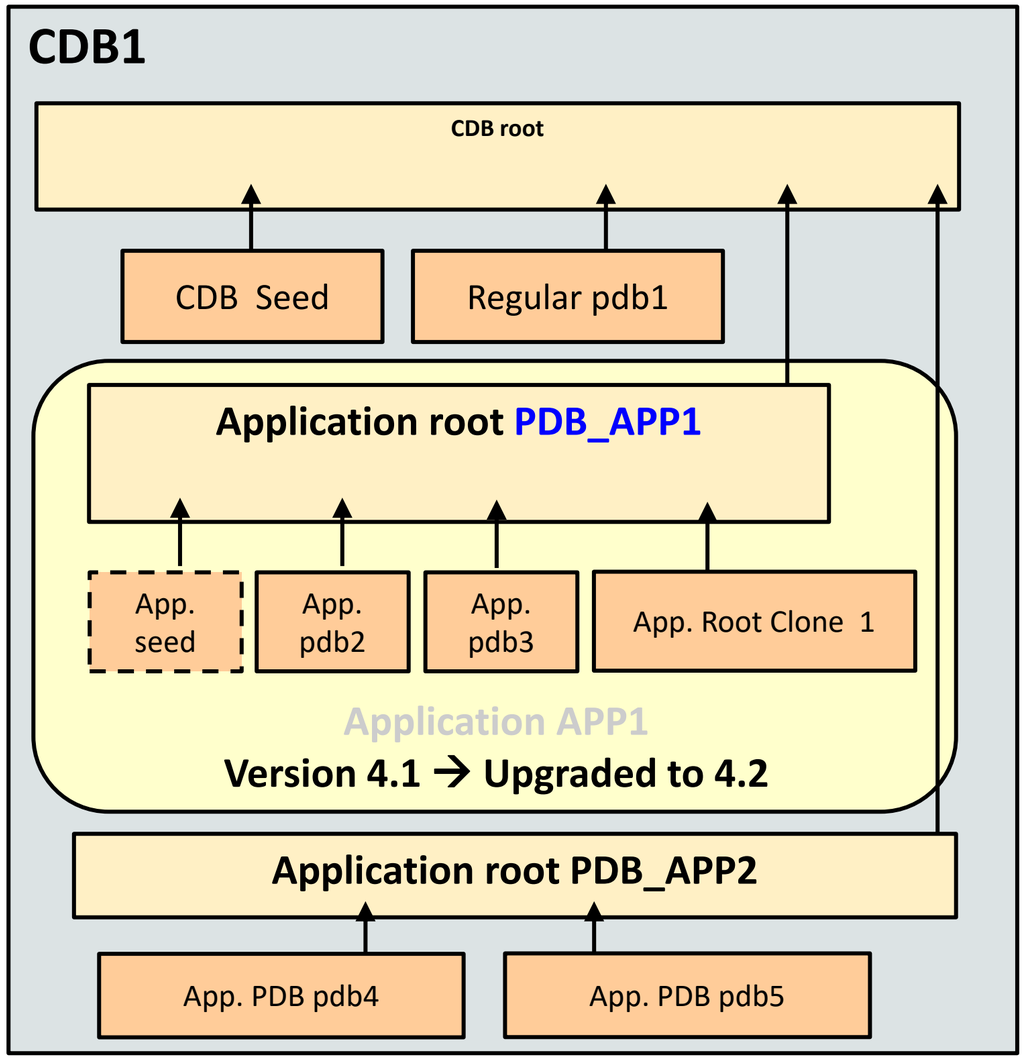
4. Finish the application installation.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION app1
             END INSTALL '4.1';
```

5. Synchronize each application PDB

```
SQL> CONNECT sys@pdb2
SQL> ALTER PLUGGABLE DATABASE APPLICATION app1
             SYNC;
```

Patching and Upgrading Applications



1. Connect to the **PDB_APP1** application root of the **APP1** application.
2. Check the current version of the **APP1** application before starting the upgrade.
3. Start the application upgrade to a higher version.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION app1
             BEGIN UPGRADE '4.1' TO '4.2';
```

4. Complete the application upgrade.

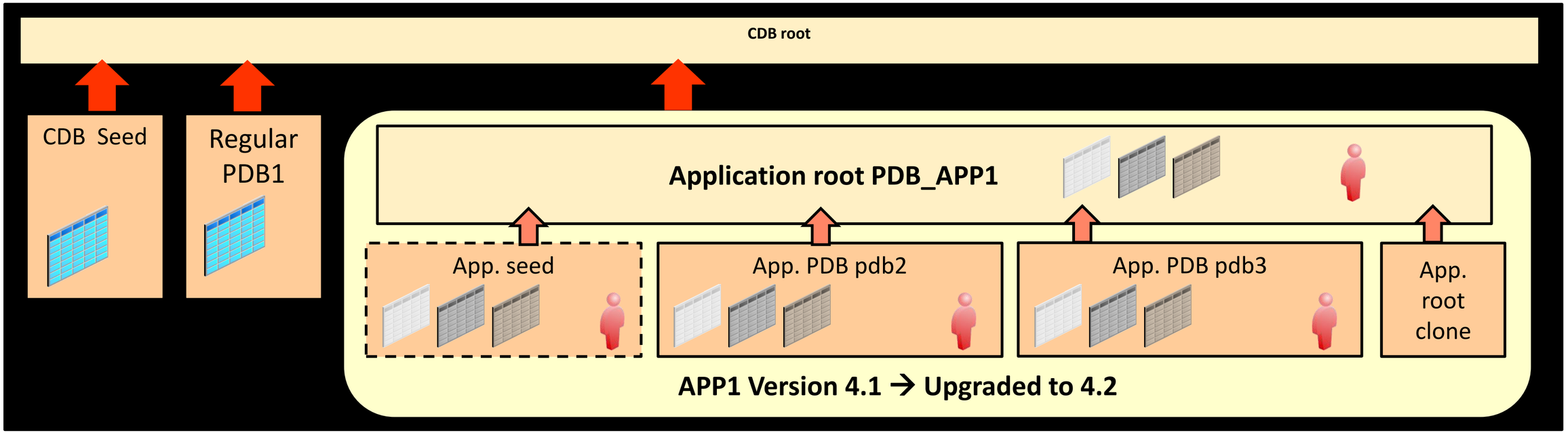
```
SQL> @scripts
SQL> ALTER PLUGGABLE DATABASE APPLICATION app1
             END UPGRADE TO '4.2';
```

5. Synchronize each application PDB.

```
SQL> CONNECT sys@pdb2
SQL> ALTER PLUGGABLE DATABASE APPLICATION app1
             SYNC;
```

Application Common Objects

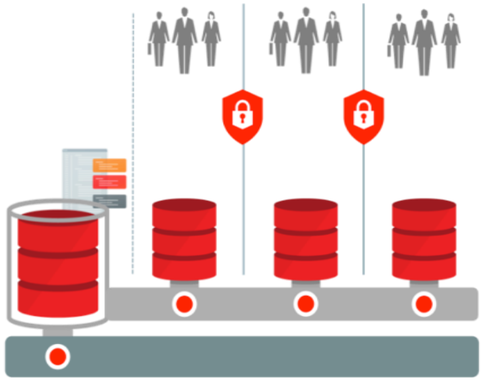
- The application root holds the common objects:
  - Users, roles, granted privileges, profiles, tables, views, and so on
- Synchronization of application PDBs with the application root is required.
- If an application is patched or upgraded, resynchronization of application PDBs is required.



Use Cases for Application Containers

Pure SaaS

- Each customer’s data resides in an individual PDB.
- All PDB-level operations are applicable on individual customer data.
- Customer data can be securely managed.
- Thousands of tenants can be handled.

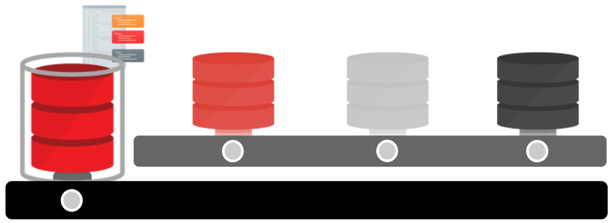


Hybrid SaaS

- Large customers reside in individual PDBs.
- Smaller customers share a PDB.
- It is suitable for applications with a high density of customers.
- Similar types of customers can be grouped in a PDB.
- Hundreds of thousands of tenants can be handled.

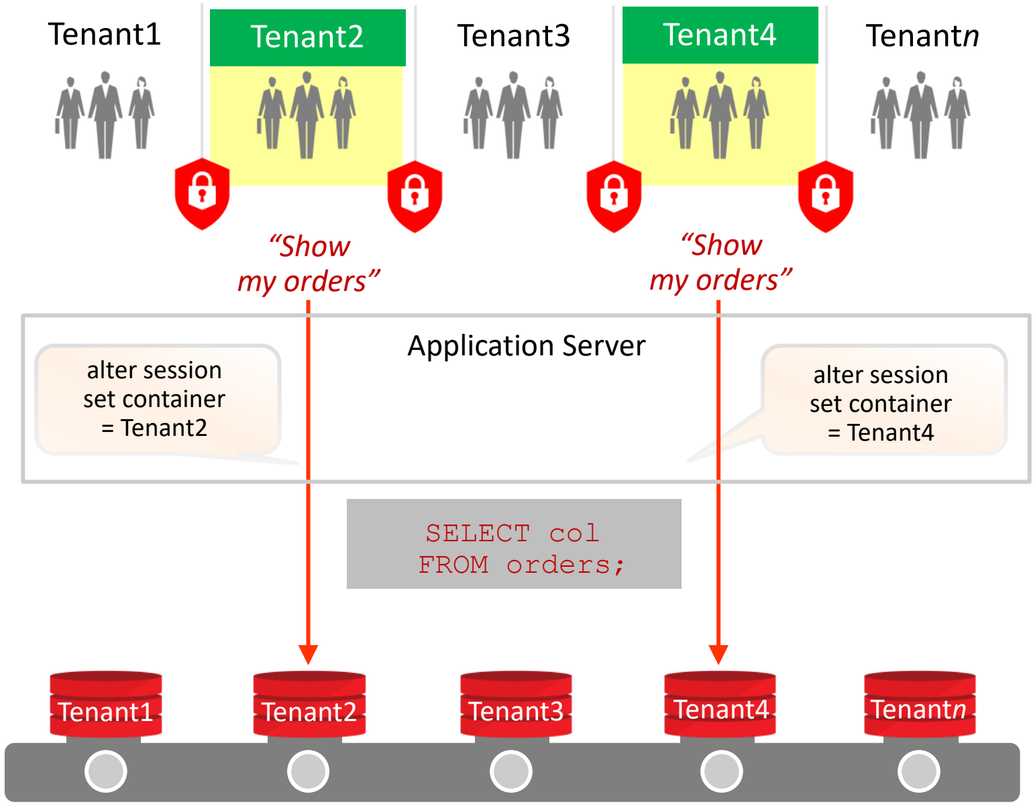
Logical DW

- Customers may address data sovereignty issues: *Country or region data will be segregated into a separate PDB.*
- There is efficient execution of ETLs for every region without impacting each other.
- The best execution plans are based on actual data distribution.

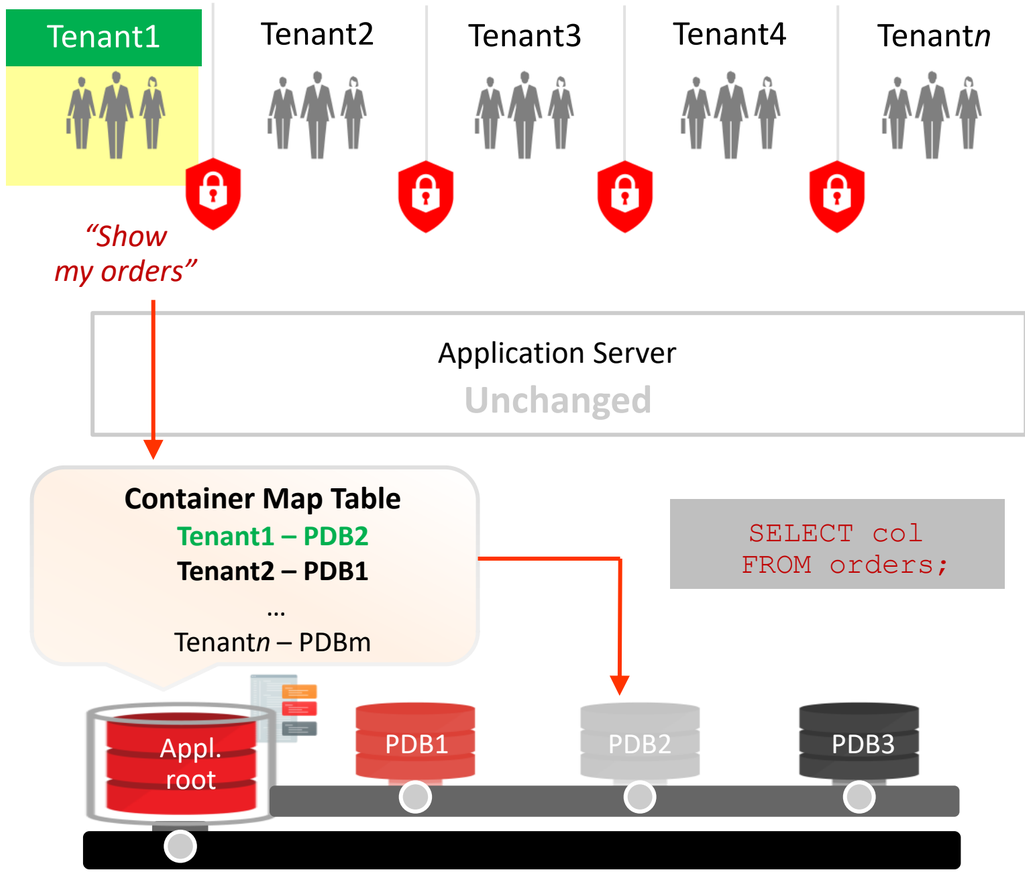


Use Case: Pure PDB-Based Versus Hybrid Model

Pure PDB-Based Tenancy

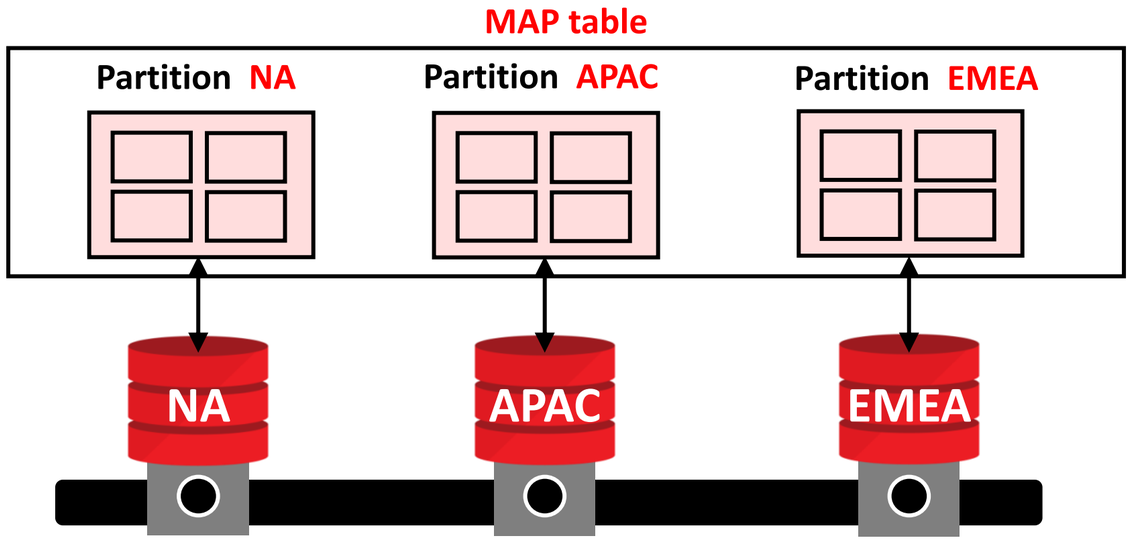


Hybrid Model: Container Map



Container Map

- Define a PDB-based partition strategy based on the values stored in a column.
- Select a column that is commonly used and never updated.
  - Time Identifier (versus creation_date) / Region Name
- Set the database property `CONTAINER_MAP` in the application root.

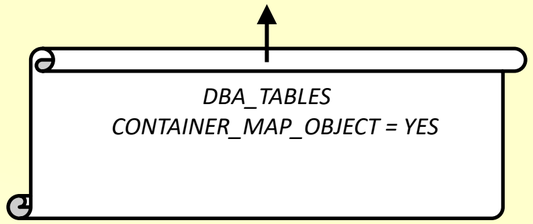


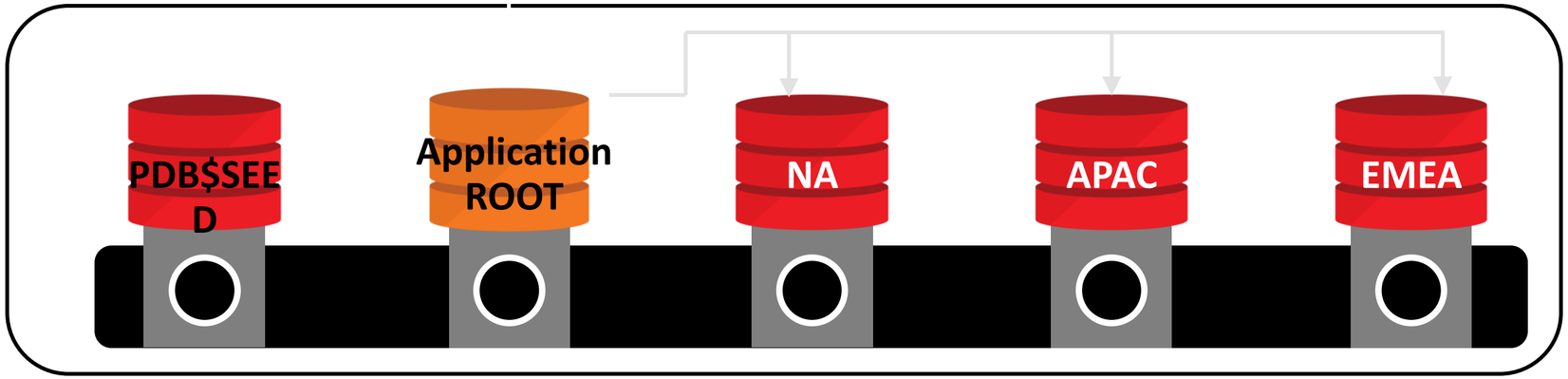
Each PDB corresponds to data for a particular partition.

DATABASE_PROPERTIES
PROPERTY_NAME = CONTAINER_MAP
PROPERTY_VALUE = app.tabapp
DESCRIPTION = value of container mapping table

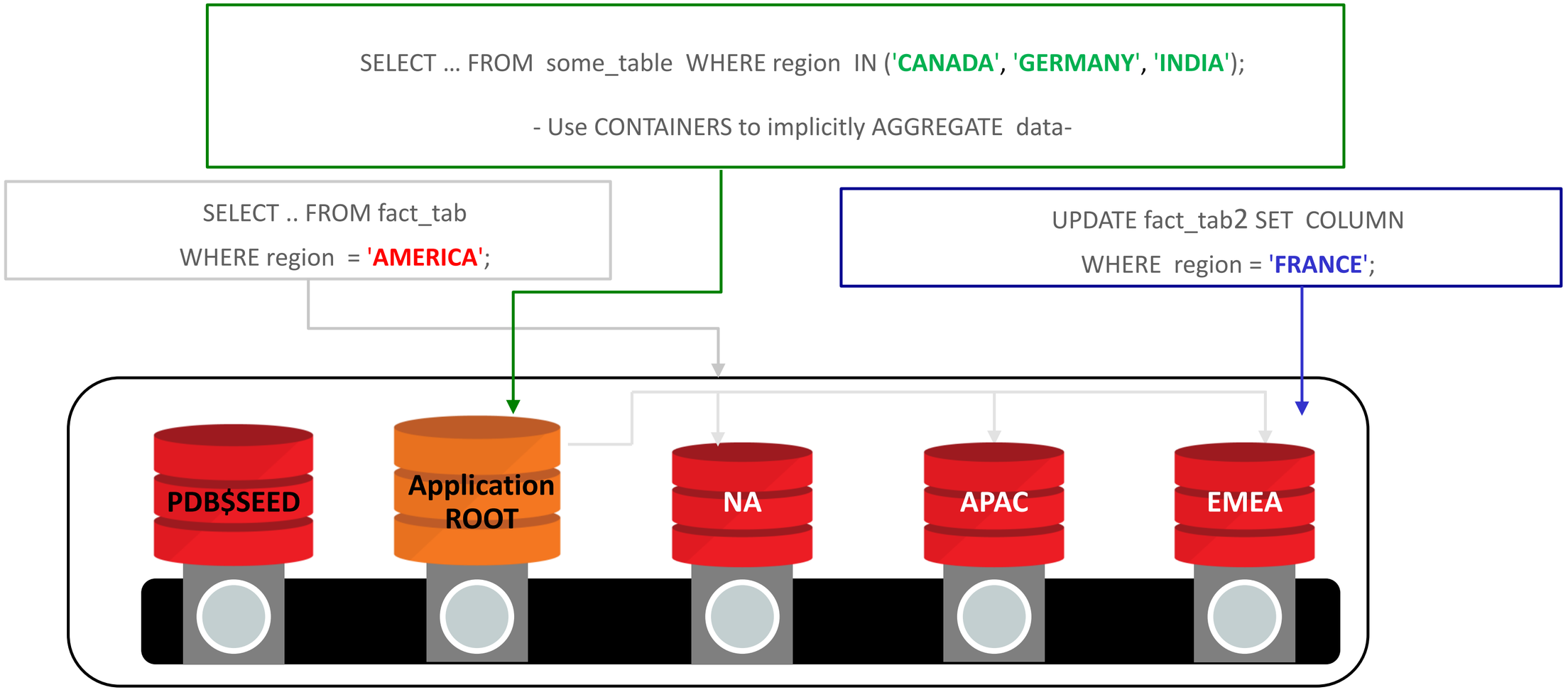
Container Map: Example

```
CREATE TABLE tab1 (region …, …);
CREATE TABLE tab2 (…, region …);

CREATE TABLE app1.app_map ( columns …, region  VARCHAR2(20))
PARTITION BY LIST (region)
 (PARTITION NA VALUES ('AMERICA', 'MEXICO','CANADA'),
  PARTITION EMEA VALUES ('UK', 'FRANCE', 'GERMANY'),
  PARTITION APAC VALUES ('INDIA', 'CHINA', 'JAPAN'));

ALTER PLUGGABLE DATABASE SET CONTAINER_MAP = 'app1.app_map';
ALTER TABLE tab1 ENABLE container_map;
```



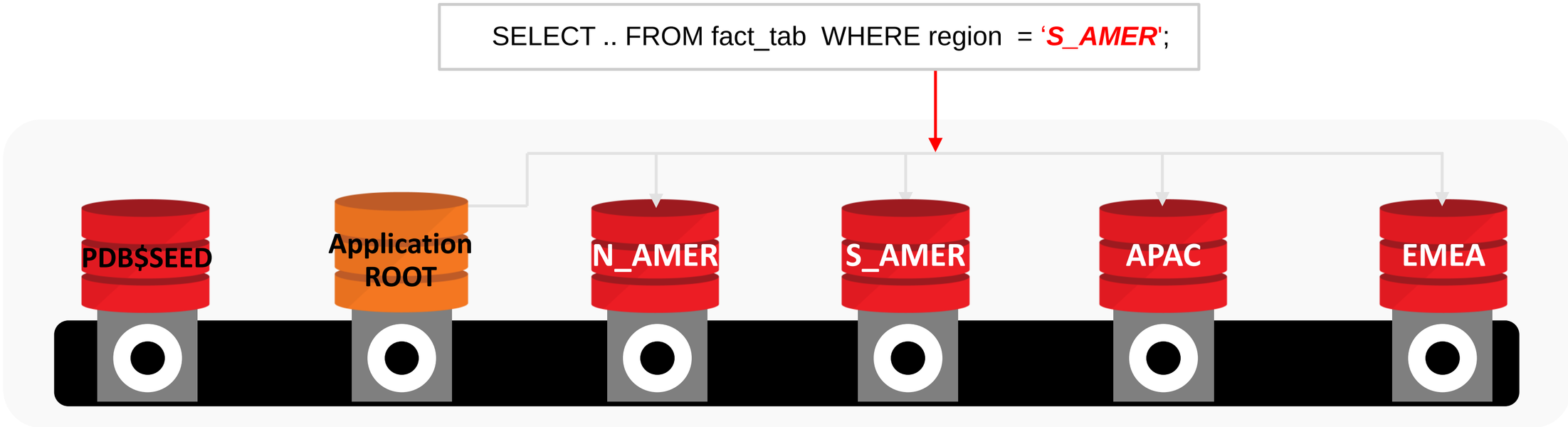


Query Routed Appropriately



Dynamic Container Map

```
CREATE PLUGGABLE DATABASE s_amer …
       CONTAINER_MAP UPDATE (ADD PARTITION s_amer VALUES ('PERU','ARGENTINA'));
```




```
CREATE PLUGGABLE DATABASE s_amer_peru …
       CONTAINER_MAP UPDATE (SPLIT PARTITION s_amer
                     INTO (partition s_amer ('ARGENTINA'), partition s_amer_peru));
```

Container Map and Containers Default

CONTAINERS_DEFAULT allows you to wrap the CONTAINERS() clause around any table.

CONTAINER_MAP, when used in conjunction with CONTAINERS_DEFAULT, prunes the partitions (PDBs) based on the key passed to the query.

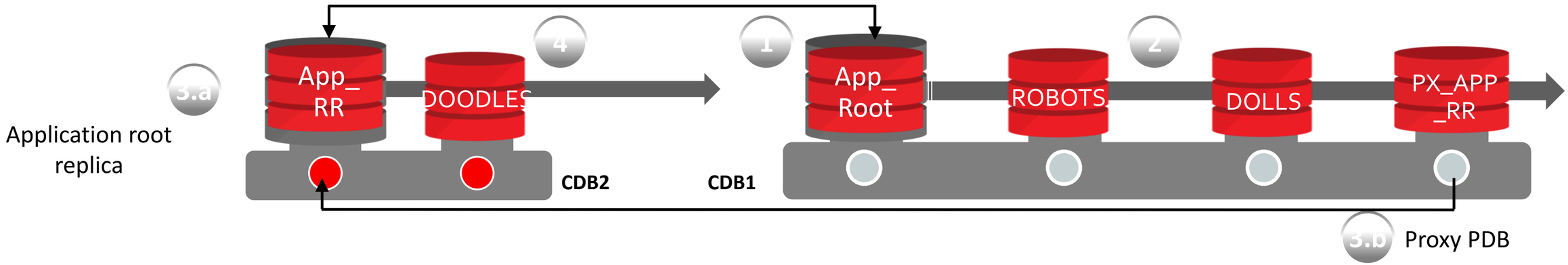
```
SELECT EMPNO
FROM CONTAINERS(EMP)
WHERE CON_ID = 6 ;
```

DBA_TABLES
CONTAINERS_DEFAULT= YES
CONTAINER_MAP = YES

```
SELECT EMPNO
  FROM EMP
WHERE DEPT = 'HR' ;
```

Query Across CDBs Using Application Root Replica

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



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

5

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

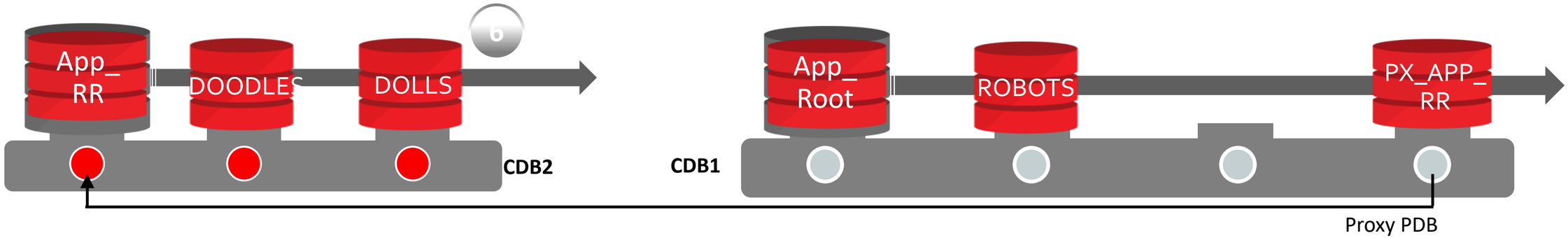
Durable Location Transparency

Load balance by relocating one of the application PDBs:

The query still retrieves all the rows from the shared table in all the PDBs under the application roots in the CDBs.

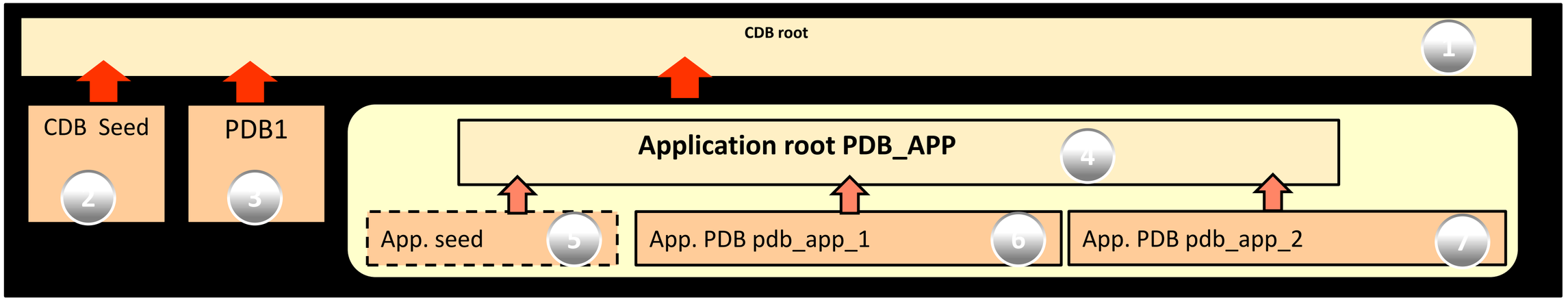
The application code is unchanged.

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



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

Data Dictionary Views




```
SQL> SELECT name, con_id, application_root "APP_ROOT", application_seed "APP_Seed",
            application_pdb "APP_PDB", application_root_con_id "APP_ROOT_CONID"
     FROM   v$containers order by con_id;

NAME         CON_ID APP_ROOT  APP_Seed  APP_PDB  APP_ROOT_CONID
------------ ------ -------- --------  -------- --------------
CDB$ROOT          1 NO       NO        NO
PDB$SEED          2 NO       NO        NO
PDB1              3 NO       NO        NO
PDB_APP           4 YES      NO        NO
PDB_APP$SEED      5 NO       YES       YES             4
PDB_APP_1         6 NO       NO        YES             4
PDB_APP_2         7 NO       NO        YES             4
```

Terminology in Application Container Context

Common versus Local:

- Users
- Privileges / Roles
- Objects
- Profiles
- Auditing policies and FGA policies
- Application context and VPD policies
- Transparent sensitive data protection (TSDP) policies
- Database Vault realms and common command rules

Note: Any statement that can be issued in a CDB root can also be issued in an application root.

Commonality in Application Containers

In an application root, statements to create common entities can be issued only as part of an application operation.

Application Operation	Common Entity
BEGIN INSTALL / END INSTALL BEGIN UPGRADE / END UPGRADE BEGIN PATCH / END PATCH	Create, alter, or drop a common user. Create, alter, or drop a common role. Create, alter, or drop a common profile. Commonly grant privileges or roles to or revoke them from a common user or common role. Create, alter, and drop common objects.

Impacts

- Per PDB character set:
  - Enables storing multilingual data
  - Facilitates conversion of existing non-CDBs to PDBs
  - Facilitates fast and seamless unplug/plug of PDBs across CDBs that have different compatible character sets
  - Is the same for all PDBs in an application container
  - Is supported with the LogMiner data dictionary
- Common unified and FGA policies in application containers
- Database Vault common realms and command rules at CDB level
- Common objects in application PDBs supported by LogMiner

Summary

In this lesson, you should have learned how to:

- Describe application containers in CDBs
- Explain the purpose of application root and application seed
- Define application PDBs
- Create application PDBs
- Explain application installation on top of application containers
- Install an application
- Upgrade and patch applications
- Describe the commonality concept in application contexts
- Use a dynamic container map
- Describe enhancements in various areas