
ORACLE®

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

ORACLE

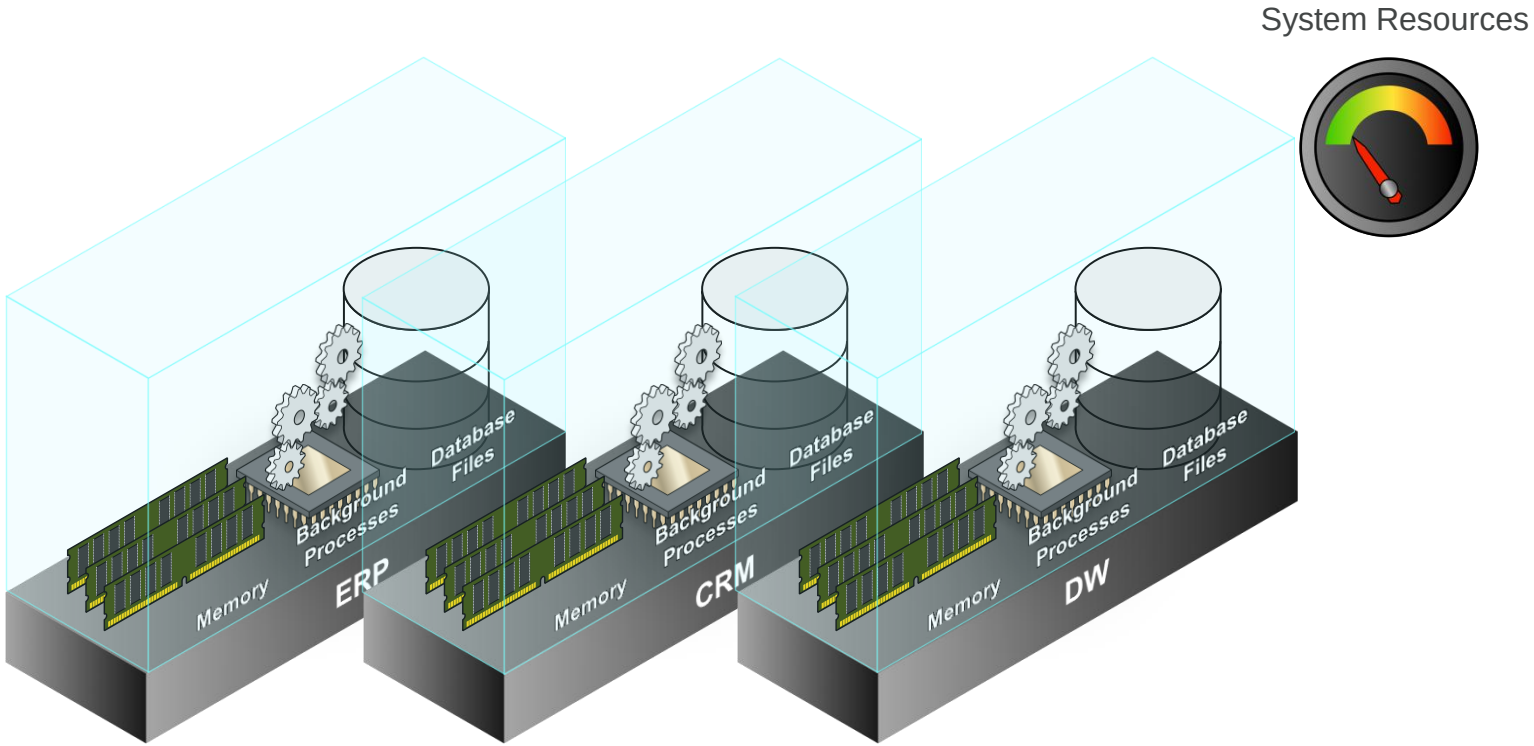
Oracle Multitenant

Simplify Consolidation with Oracle Database 12c

Morana Kobal Butković
Principal Sales Consultant
Oracle Hrvatska

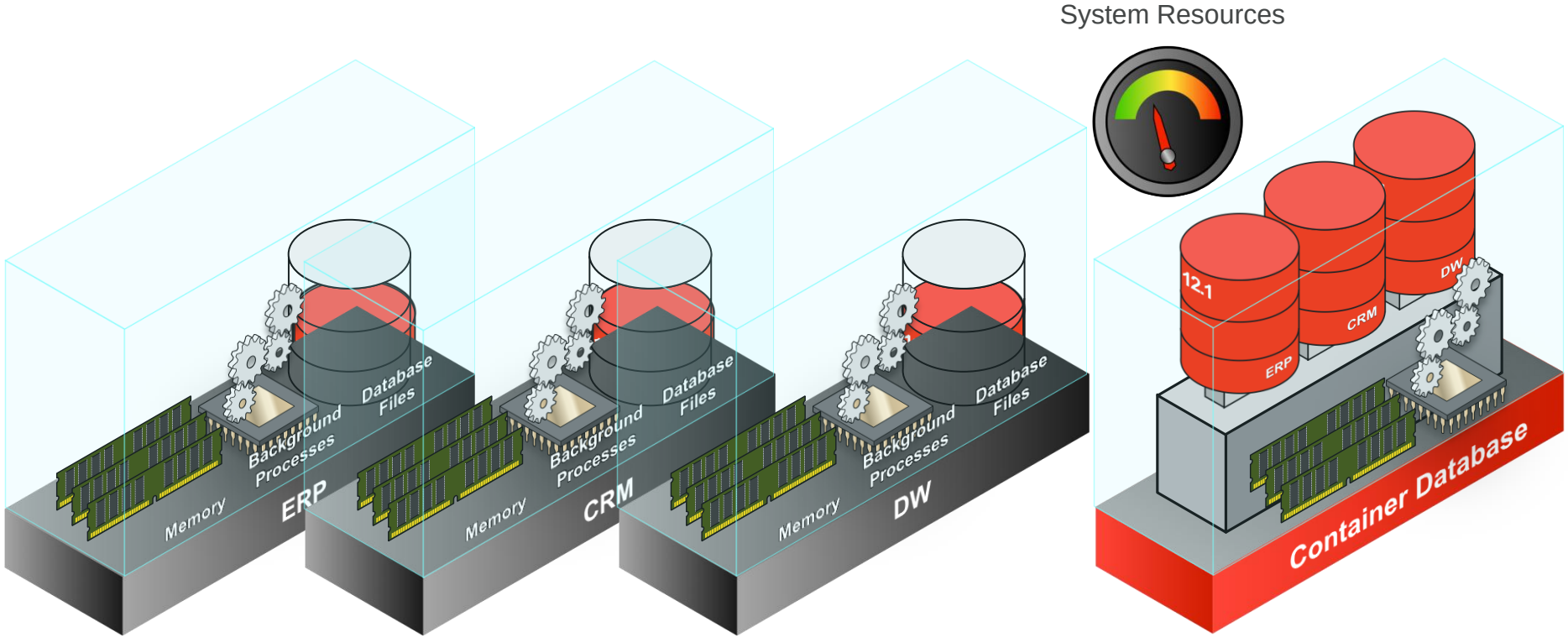
Oracle Database Architecture

Requires memory, processes and database files



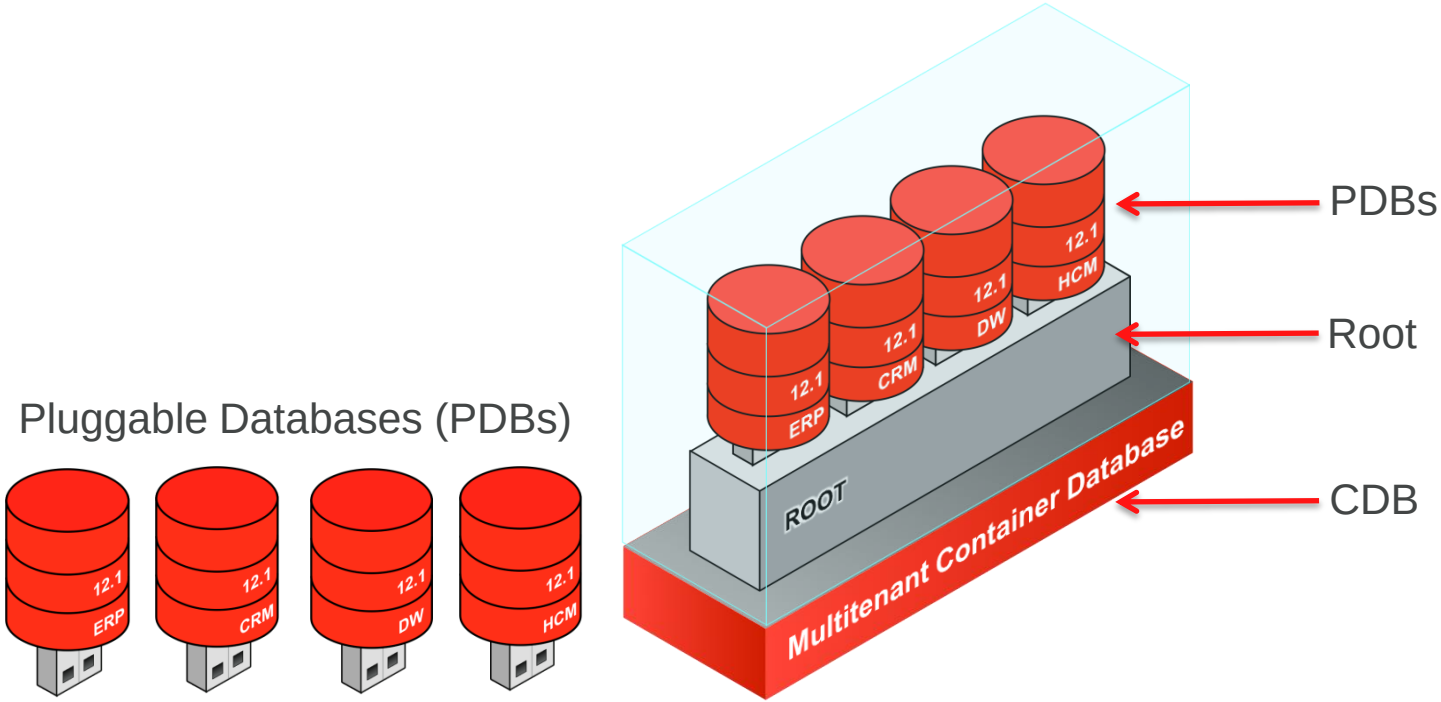
New Multitenant Architecture

Memory and processes required at multitenant container level only

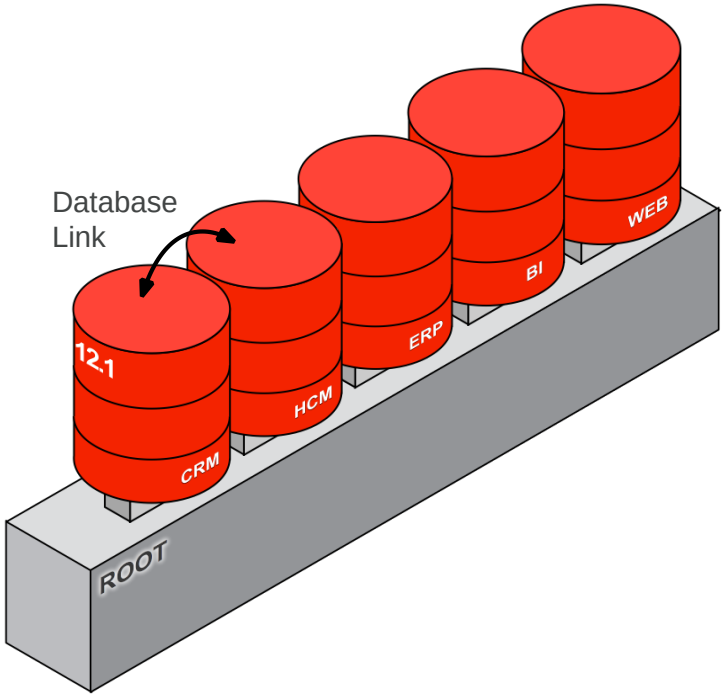


Multitenant Architecture

Components of a Multitenant Container Database (CDB)



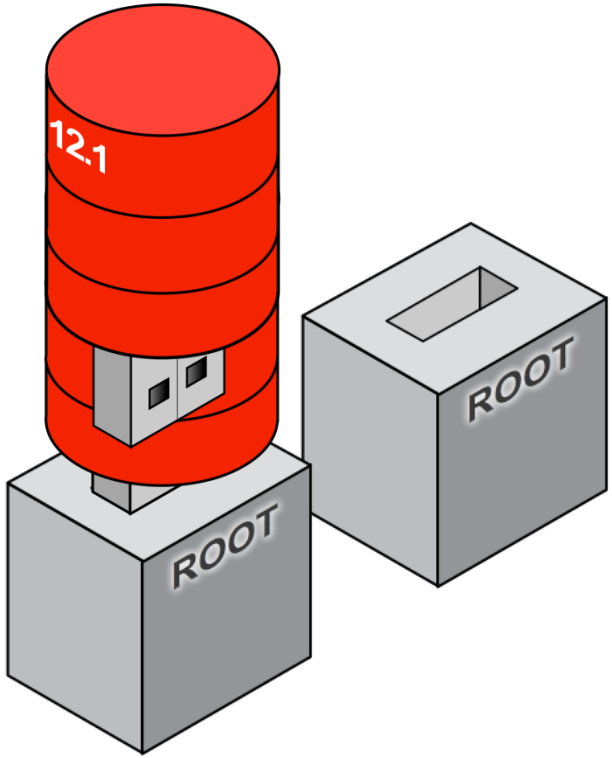
Multitenant Architecture



- Multitenant architecture can currently support up to 252 PDBs
- A PDB feels and operates identically to a non-CDB
- You cannot tell, from the viewpoint of a connected client, if you're using a PDB or a non-CDB

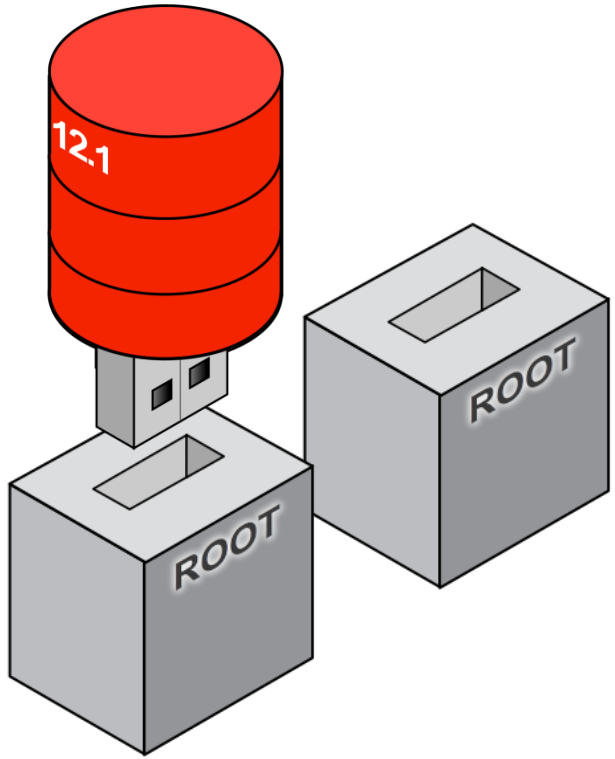
Unplug / plug

Simply unplug from the old CDB…



Unplug / plug

…and plug in to the new CDB…



- Moving between CDBs is a simple case of moving a PDB's metadata
- An unplugged PDB carries with it lineage, opatch, encryption key info etc

Unplug / plug

Example

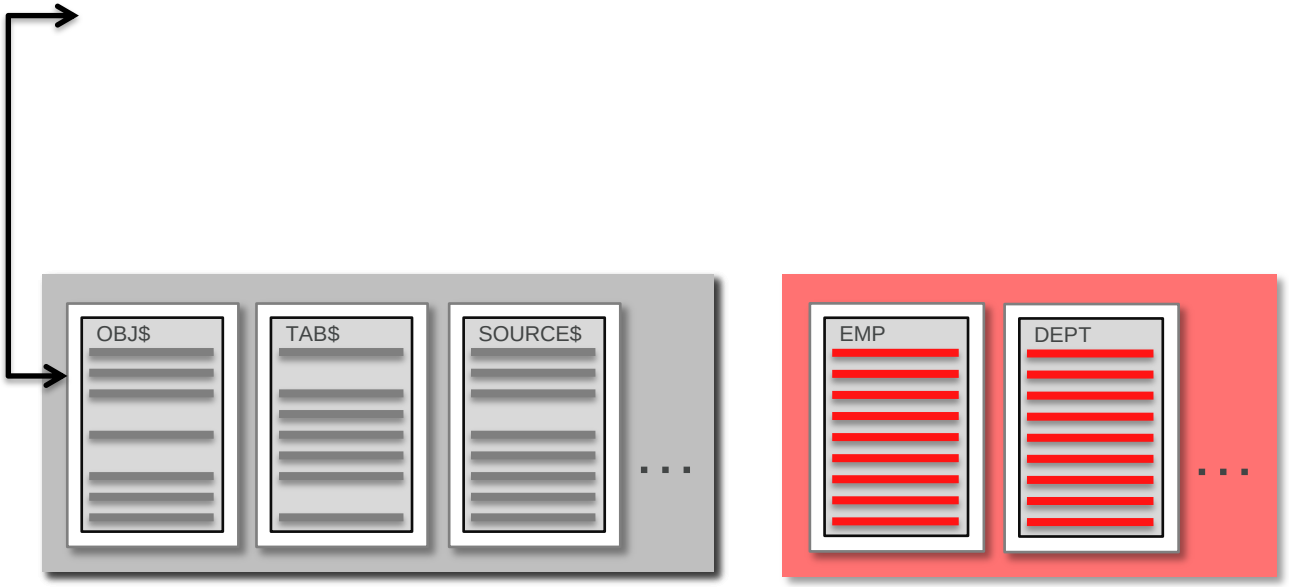
Unplug

```
alter pluggable database HCM
unplug into '/u01/app/oracle/oradata/…/hcm.xml'
```

Plug

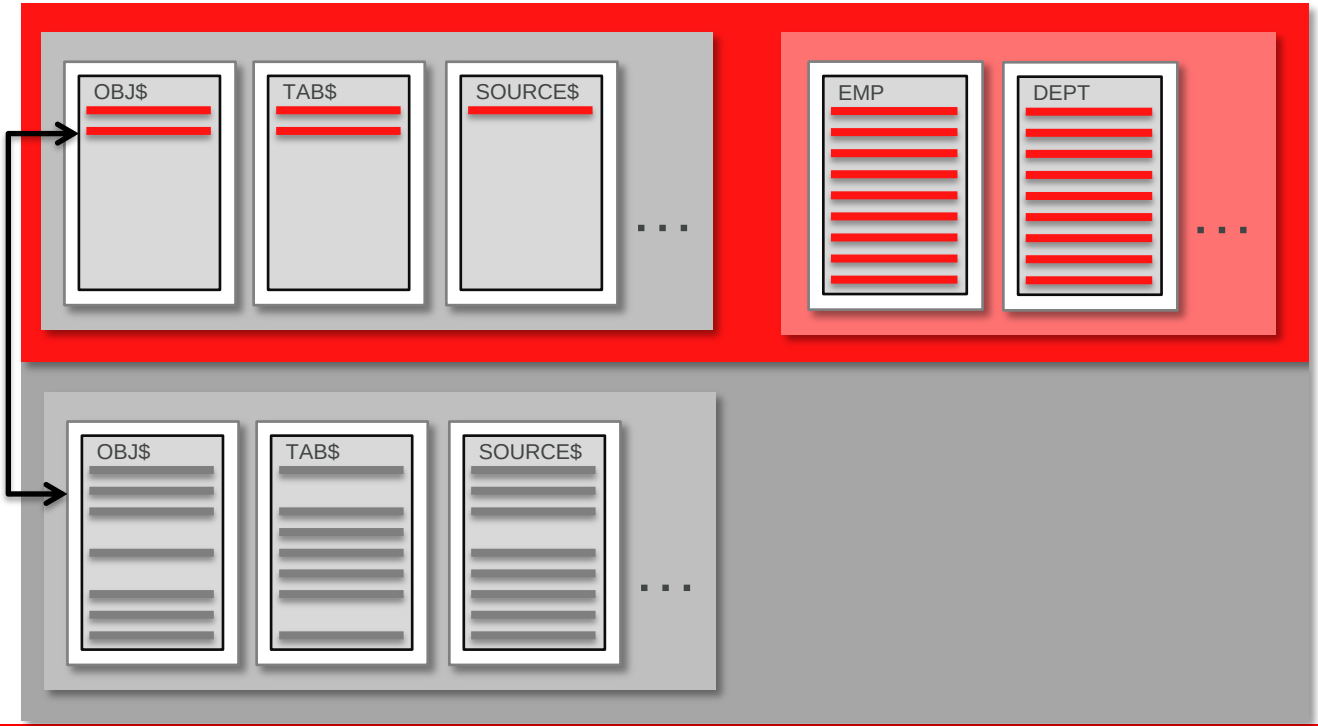
```
create pluggable database My_PDB
using '/u01/app/oracle/oradata/…/hcm.xml'
```

Oracle Data and User Data



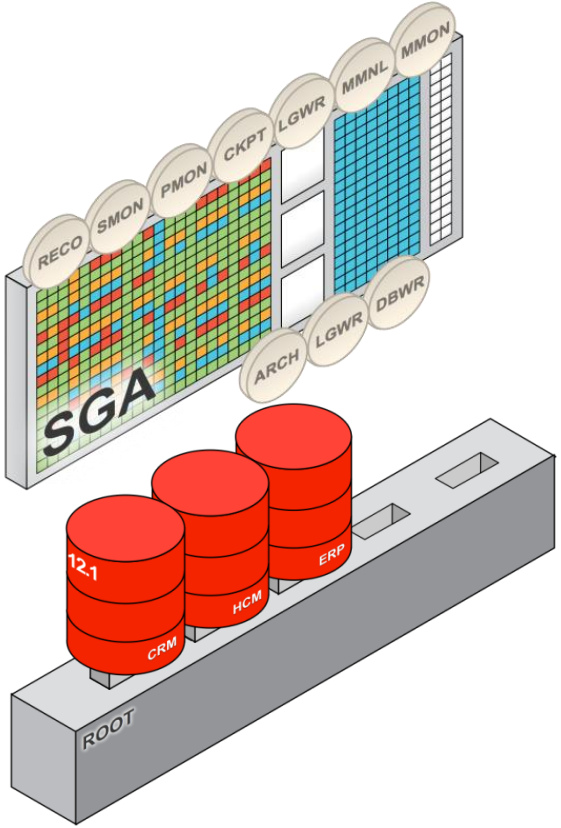
- Multitenant fix: Horizontally-partitioned data dictionary
- Only Oracle system definition remains
- Data dictionary is diluted by customer's metadata

Horizontally Partitioned Data Dictionary



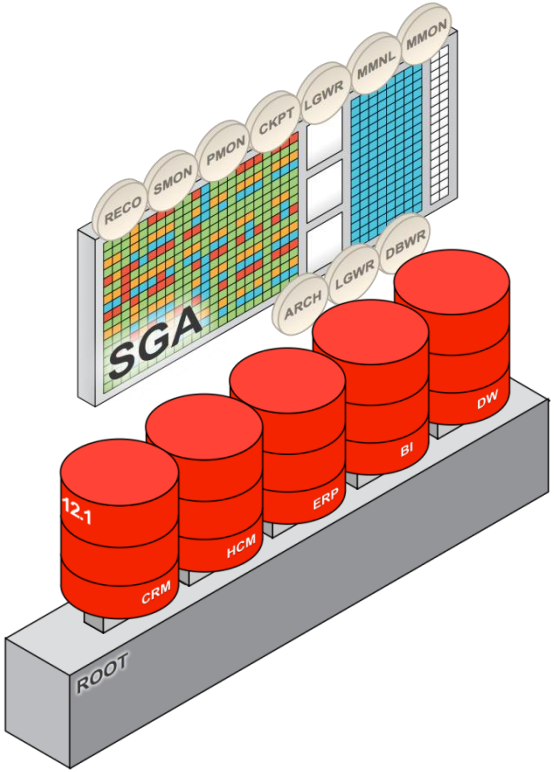
- Oracle-supplied objects such as views, PL/SQL, etc., are shared across all PDBs using object “stubs”
- In-database virtualization

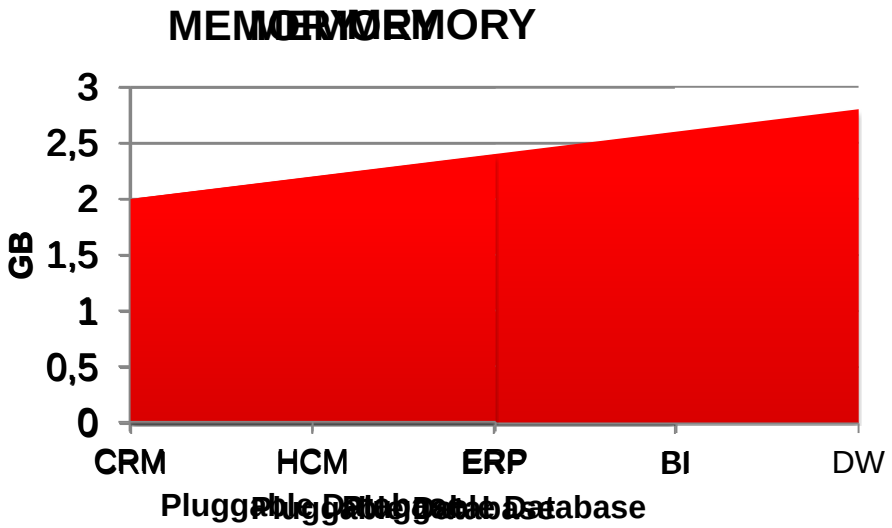
Multitenant Architecture – Dynamics



- PDBs share common SGA and background processes
- Foreground sessions see only the PDB they connect to

Multitenant Scalability

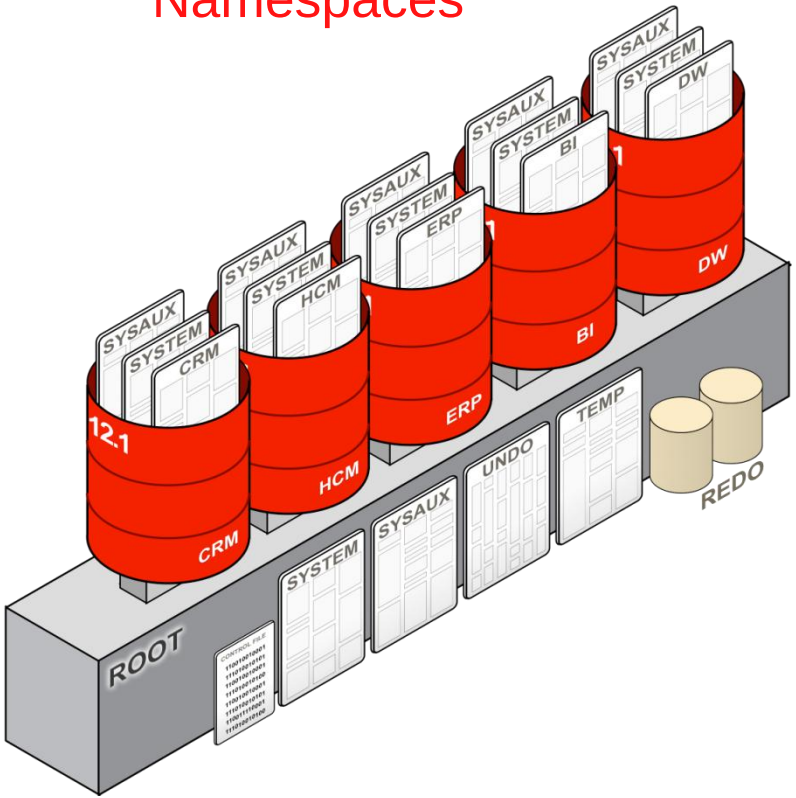




- Only small increments in memory as additional PDBs are added

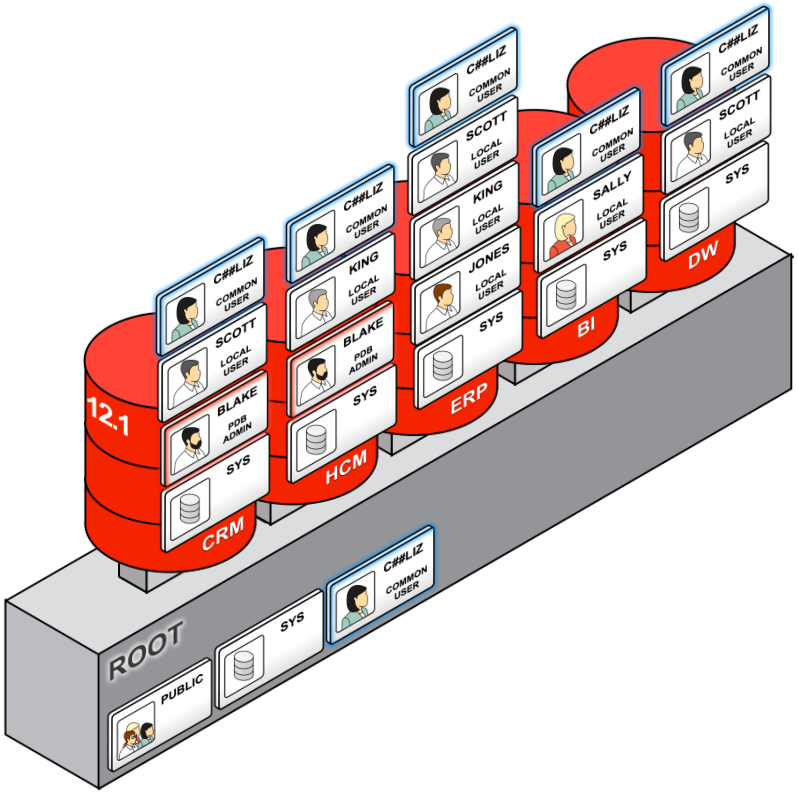
Files in the CDB

Namespaces



- Each PDB has its own set of tablespaces including SYSTEM and SYSAUX
- PDBs share UNDO, REDO and control files, (s)pfile
- By default the CDB has a single TEMP tablespace but PDBs may create their own

Users



- Local users are the successors for customer-created users in a non-CDB
- A local user is defined only in a PDB
- A local user can administer a PDB
- A common user is defined in the root and is represented in every PDB
- A common user can log into any PDB where it has “Create Session” and can therefore administer a PDB
- The Oracle system is owned by common users

Common Users and Privileges

Authorization is checked in the same way as as pre-12.1

- A common user can be granted privileges locally in a PDB (or root) and therefore differently in each container
- A common user can, alternatively, be granted a system privilege *commonly* – the grant is made in root and every PDB, present and future
- You can create a common role
- A common role can be granted to a common user commonly
- Authorization is checked in the container where the SQL is attempted considering *only* the privileges that the user has in that container

Creating Common Users

Example

```
create user c##Über_Administrator
identified by pwd container=all
```

Services and Sessions

The “five facts” have the same result as pre-12.1

- Listener location, listener port, service name, username, password
- Now a service has a new property: the initial PDB
- A session is created in essentially the same as before
- You can connect to a PDB *only* by using network authentication
- A local user can create a session only in the PDB where it is defined, if it has **`Create Session`** there
- A common user can create a session in any PDB where it has **`Create Session`**

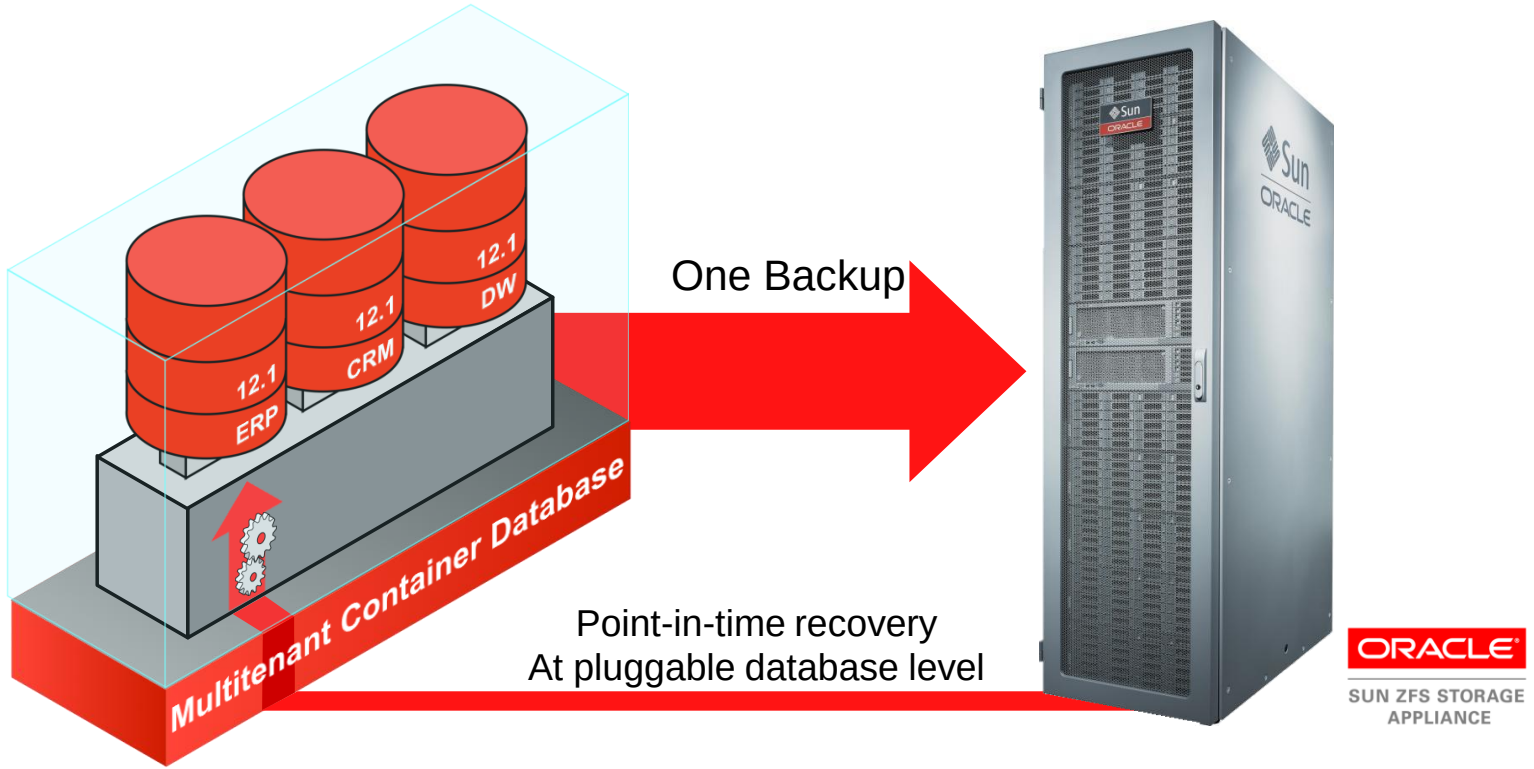
Dictionary and Performance views

PDB-to-PDB privacy; overall system image

- The performance views have a new **`Con_ID`** column
- There's a new **`CDB_`** dictionary view family with a **`Con_ID`** column
- These are so-called container-data objects
- When queried from a PDB, the results show only **`Con_ID`** for that PDB
- When queried from the root, the results *may* show all **`Con_ID`** values
- The containers you see depend on an attribute of the user that lists these

Manage Many as One with Multitenant

Backup databases as one; recover at pluggable database level



PDB Point-in-time Recovery (PITR)

Recovery isolation at granularity of PDB

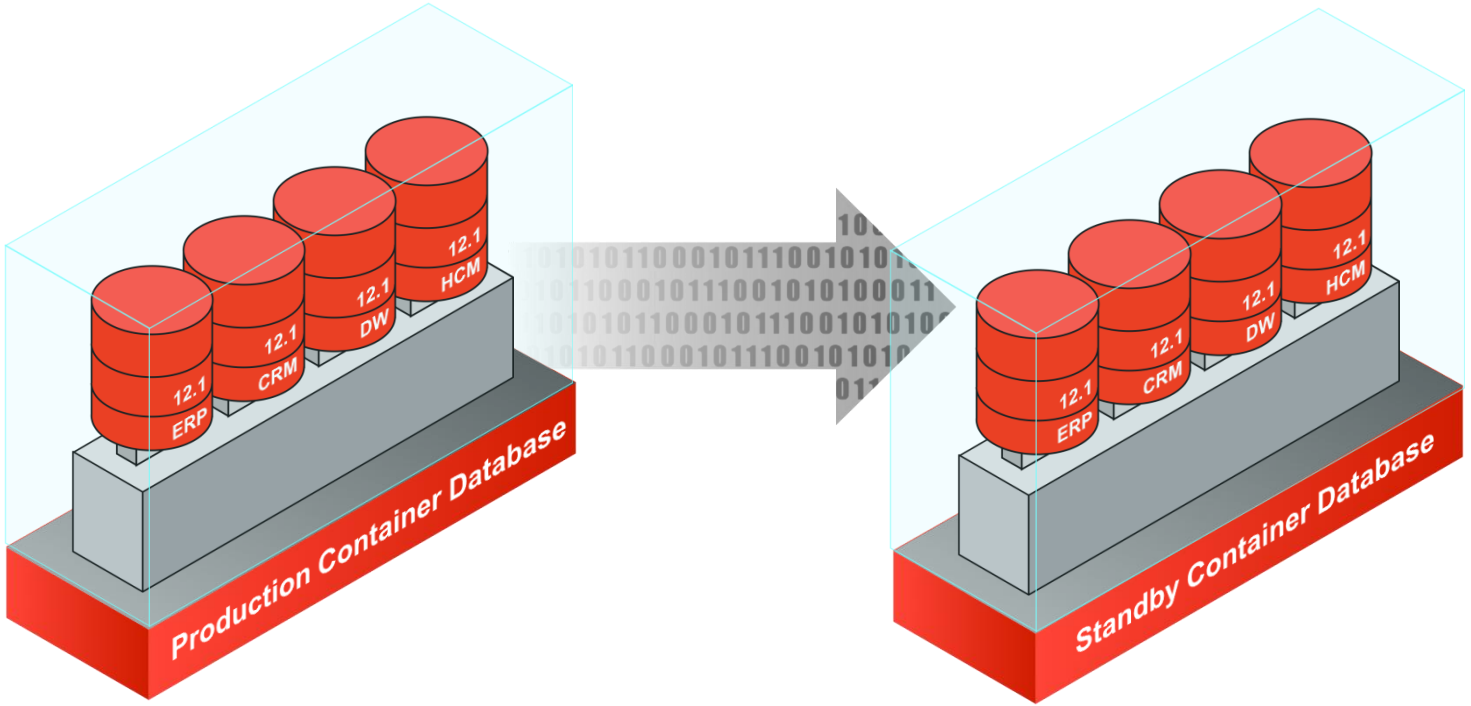
- PDB PITR consists of 3 simple steps from RMAN:

```
RMAN> restore pluggable database my_db;
RMAN> recover pluggable database my_db until SCN 411010;
RMAN> alter pluggable database my_db open resetlogs;
```

- All datafiles are recovered in-place, undo is recovered out of place and applied in-place during open resetlogs.
- Old backups of the PDB will continue to be valid.

Manage Many as One with Multitenant

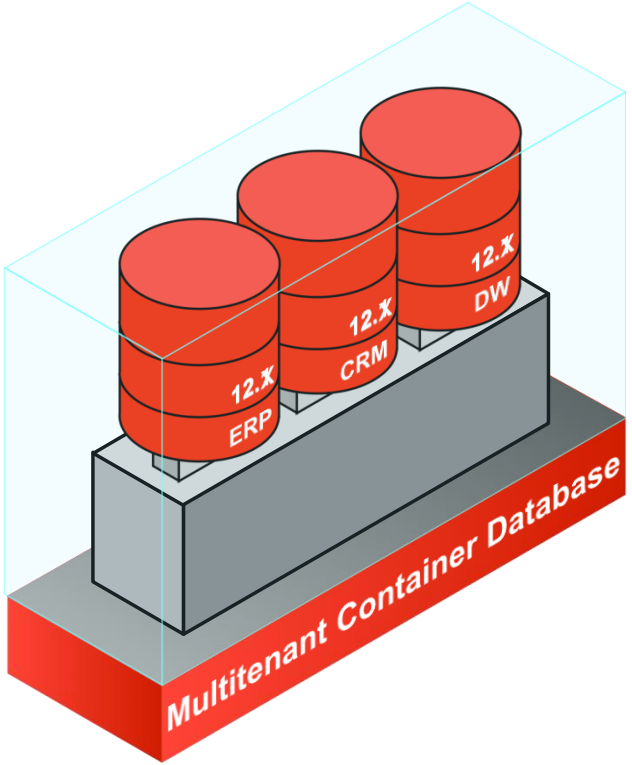
One standby database covers all pluggable databases



Multitenant for Simplified Patching

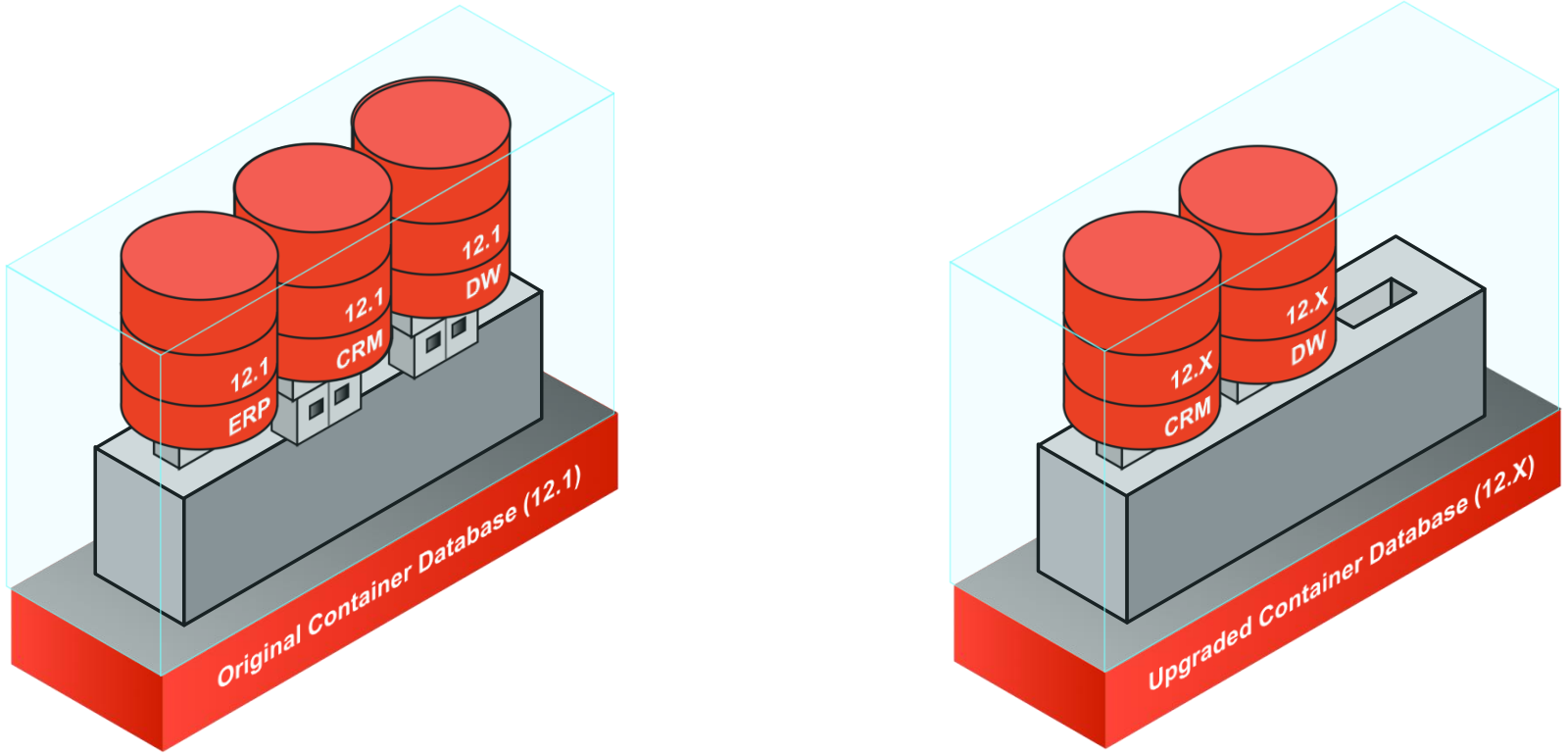
Apply changes once, all pluggable databases updated

Upgrade in-place



Multitenant for Upgrades

Flexible choice when patching & upgrading databases



Patching via Unplug/Plug

Details

- Plug compatibility
- Xml manifest
- Things to check
  - Options
  - Character set
  - Parameters
  - Endianness
- Post-plug, run any necessary SQL patch script in PDB
  - Nothing required for binary patches
- This method can be used for upgrades to next patchset

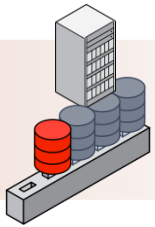
Unprecedented Agility with Pluggable Portability

PDB migrates through SLAs as it becomes more mission critical

GOLD RAC, Data Guard, Daily Incrementals

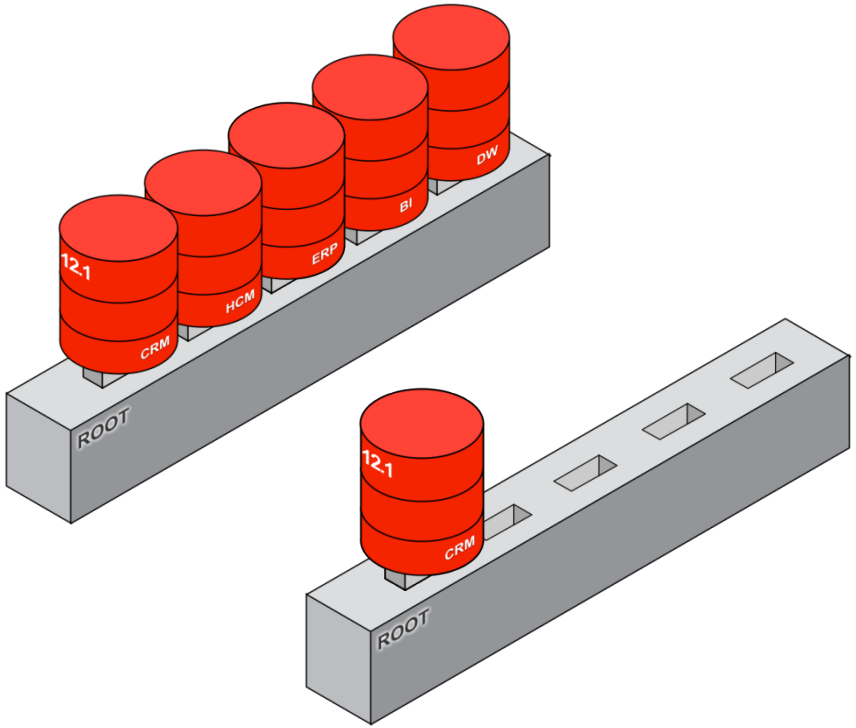
SILVER Data Guard, Daily Incrementals

BRONZE Weekly Full Backups



Multitenant for Provisioning

Fast cloning of PDBs



- PDBs can be cloned from within the same CDB
- PDBs can be cloned from remote CDBs

Cloning a PDB

Example

Local

```
create pluggable database HCMBI from HCM
```

Remote (DB Link)

```
create pluggable database HCMBI from HCM@us.acme.db1
```

Multitenant Cloning with Full Copy

Efficient, simple & reliable process

- Orchestration entirely from SQL
  - DBA skills only
  - No o/s access required
- Parallel Execution Servers
  - Fast and efficient
- Clones have unique GUIDs
- Source PDB must be open read only

Multitenant Cloning with Snapshot Copy

Thin provisioning of database clones in seconds

```
create pluggable database DEV44
from CRM_MASTER snapshot copy
```

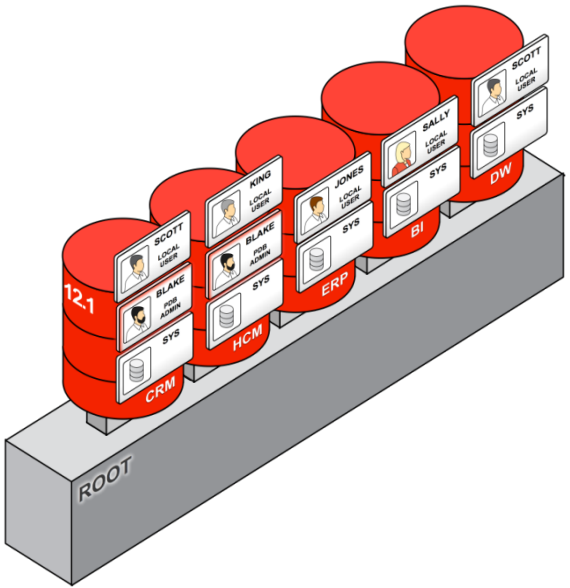
- Still SQL with new “as snapshot” clause
- Where not supported receive error
  ORA-17517: Database cloning using storage snapshot failed
- Built on copy-on-write capability of underlying file system
- Storage admin’s credentials stored in secure wallet once per CDB
- No subsequent requirement for intervention of storage administrator

Snapshot Clone Restrictions

- Local clones only
- Source PDB cannot be dropped
- Source PDB cannot be unplugged
- Both source and cloned PDBs may be opened read-write

- Supported platforms
  - Sun ZFS Storage Appliance (ZFSSA)
  - Oracle ASM Cluster File System (ACFS)
  - NetApp ™

PDB Provisioning



- Copy files from seed db
- Copy files from another pdb locally
- Copy files from another pdb remotely
- Plug in a unplugged pdb
- Plug in a non-cdb as pdb

Provisioning a PDB

Example

```
create pluggable database CRM
admin user crmadm identified by CRMpass123
roles = (DBA)
```

PDB Thin Provisioning

- Copy files from another pdb locally using snapshot copy option

```
create pluggable database test_hr from prod_hr snapshot copy;
```

- Makes use of storage level snapshots
- Volume tag used for each snapshot is unique
- Appears as regular copied files
- Snapshot files linked to file name provided by user using file_name_convert clause
- Monitor PDB clones using V$DBA_PDB_STORAGE_CLONE
- Supported on ZFS, ACFS, NetApp

PDB Thin Provisioning using ACFS

- Provision new PDB as snapshot clone

```
alter pluggable database hr_pdb1 open read only;

create pluggable database hr_pdb2 from  hr_pdb1
file_name_convert=('hrpdb1', 'hrpdb2') snapshot copy;
```

- Create ACFS clone by calling acfsutil

```
/sbin/acfsutil snap create <voltag> /orcl/myacfs/hrpdb1
```

- ACFS snapshots are created on the same file system
  - /orcl/myacfs/hrpdb1
  - No separate mount performed

```
<source_mntpt>/.ACFS/snaps/<voltag>
```

PDB Thin Provisioning

Best Practices

- Source and copy PDB should be in same CDB
- Source PDB cannot be dropped when copy PDB is still in use
- PDB created using snapshot copy cannot be unplugged
- Source PDB should be open in read-only mode
- Storage credentials for ZFS / NetApp in DB Keystore
- Enable dNFS, SNMP for ZFS / NetApp storage
- If using NetApp storage download vol_clone binary from NetApp.com

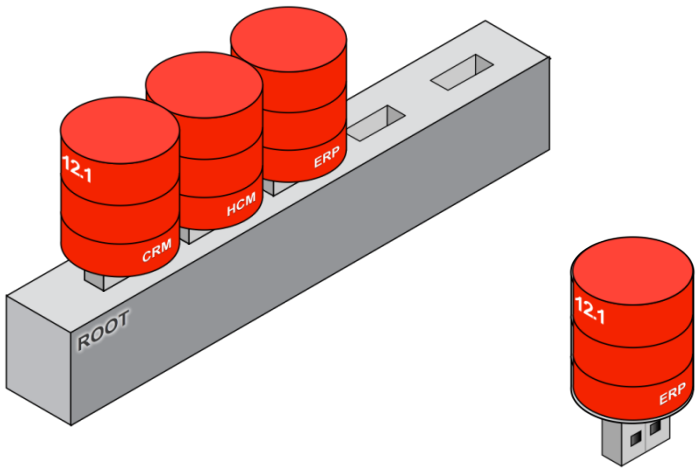
Per PDB *vs* per CDB

Common operations on CDB with granular control where appropriate

Per CDB	Per PDB
Single Oracle Software Version	RMAN point-in-time recovery
Data Guard	Ad hoc RMAN backups
Scheduled RMAN Backups	Flush shared pool
Some parameters/properties e.g. homogeneous character set	Parameters where `IsPDB_Modifiable = 'TRUE'`
Redo and Undo	

Upgrade to Multitenant from Oracle Database 11g

Upgrade 11g database and plug in



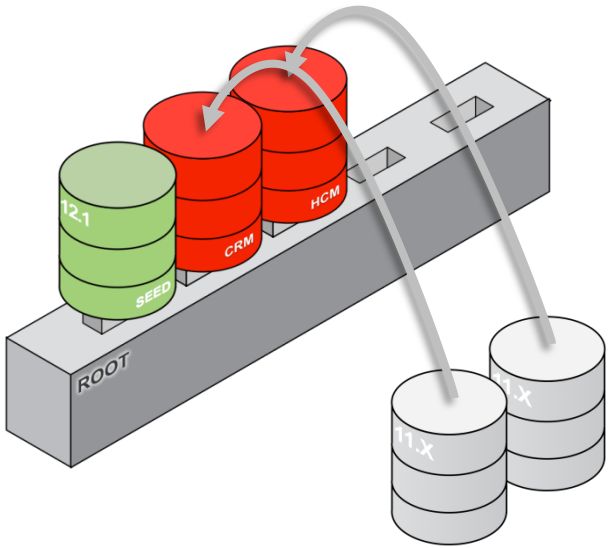
1. Upgrade 11.2 database to 12.1 in place
2. Place the non-CDB into read-only mode
3. Connect to non-CDB and generate a description file (manifest)
4. Shutdown the non-CDB
5. Plug in non-CDB to CDB
6. Post-plug script to remove redundant metadata for the Oracle system

Generate the PDB Manifest

Example

```
begin
  DBMS_PDB.Describe(
    PDB_Descr_File => '/oracle/home/hcmmeta.xml');
end;
```

Migrate using Replication



1. Provision new PDB from Seed
2. Replicate using technologies such as Oracle GoldenGate or Data Pump

New in 12.1, you ask that full database export and full database import make maximum use of transportable tablespaces in the single *expdb* and *impdb* commands.

(Backported to 11.2.0.3.)

Over the wire migration via Data Pump

Full Transportable Tablespace – new in 12.1, backported to 11.2.0.3

```
impdp system/manager123@cdbdatabase/hcm
FULL=y
NETWORK_LINK=com.us.acme.db1
DIRECTORY=dmpdir
```

Use Cases

1. Development / Testing – *start here!*
2. Consolidation of Disparate Applications
3. Database as a Service (DBaaS) on Private Cloud
4. Software as a Service (SaaS)
5. ISVs: Distribution of Packaged Apps and Data
6. …many others!

Multitenant Demo