
ORACLE®

Oracle Multitenant Best Practices

Deba Chatterjee
Senior Principal Product Manager

Giridhar Ravipati
Principal Member of Technical Staff

November 21, 2015

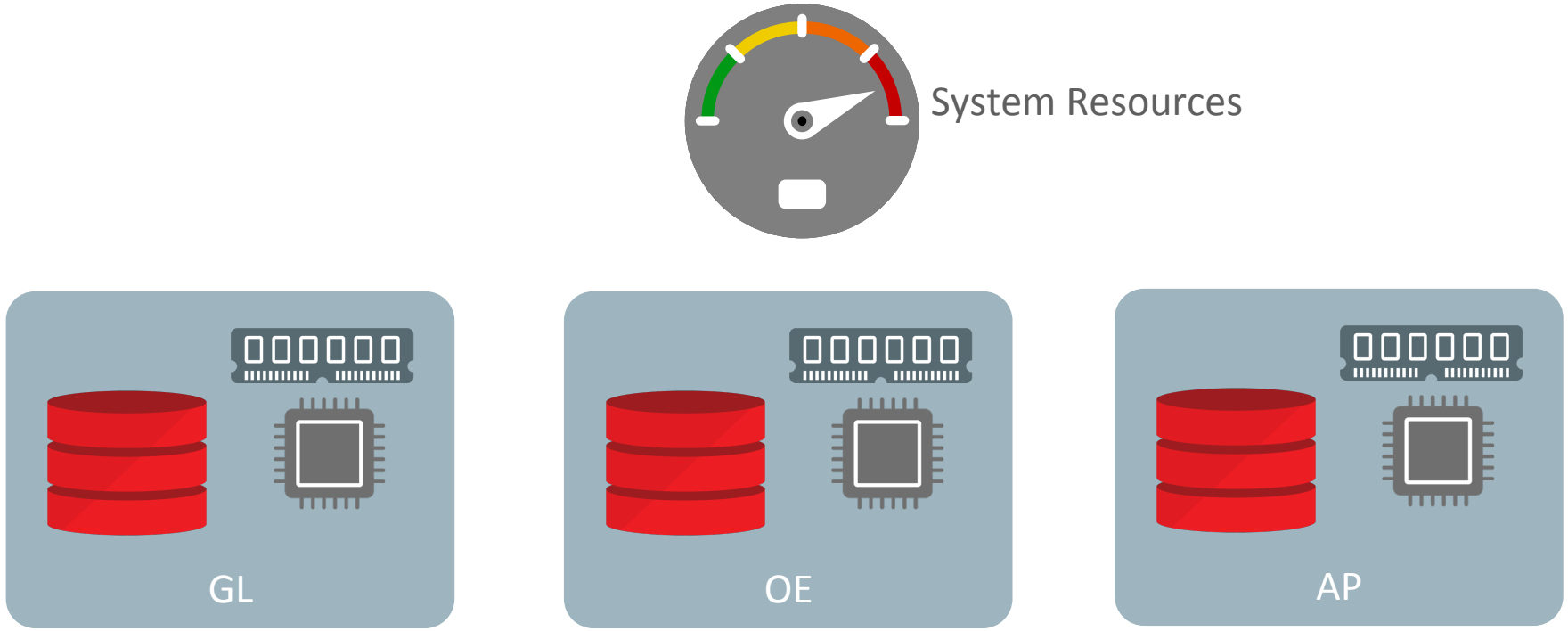
ORACLE

Safe Harbor Statement

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.

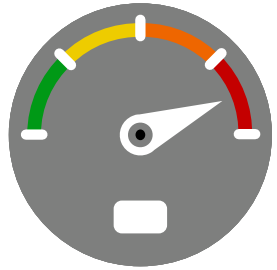
Oracle Database Architecture

Requires memory, processes and database files

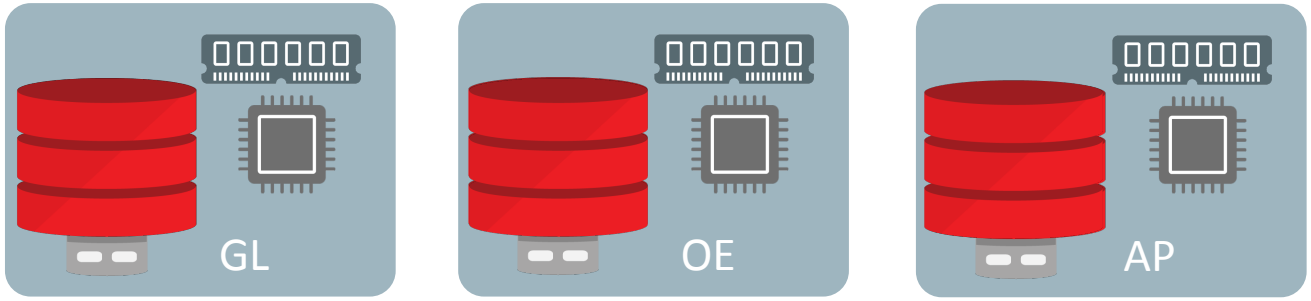


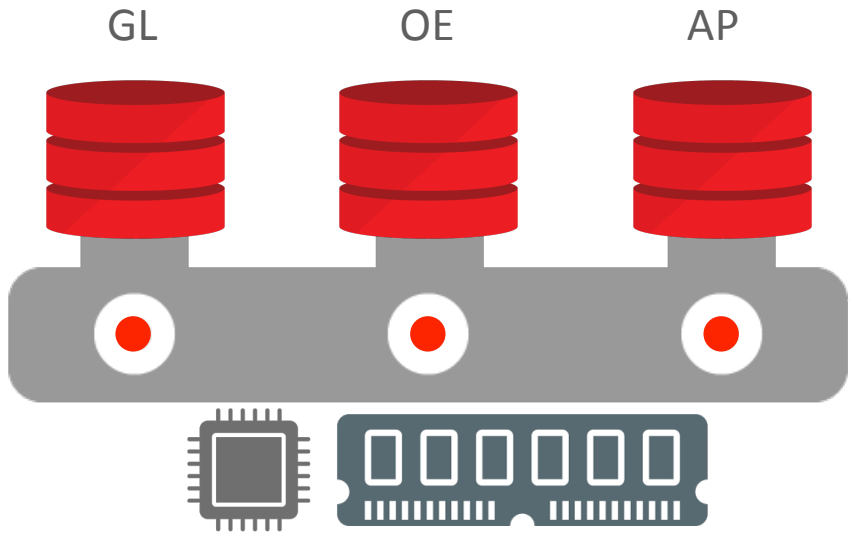
New Multitenant Architecture

Memory and processes required at container level only



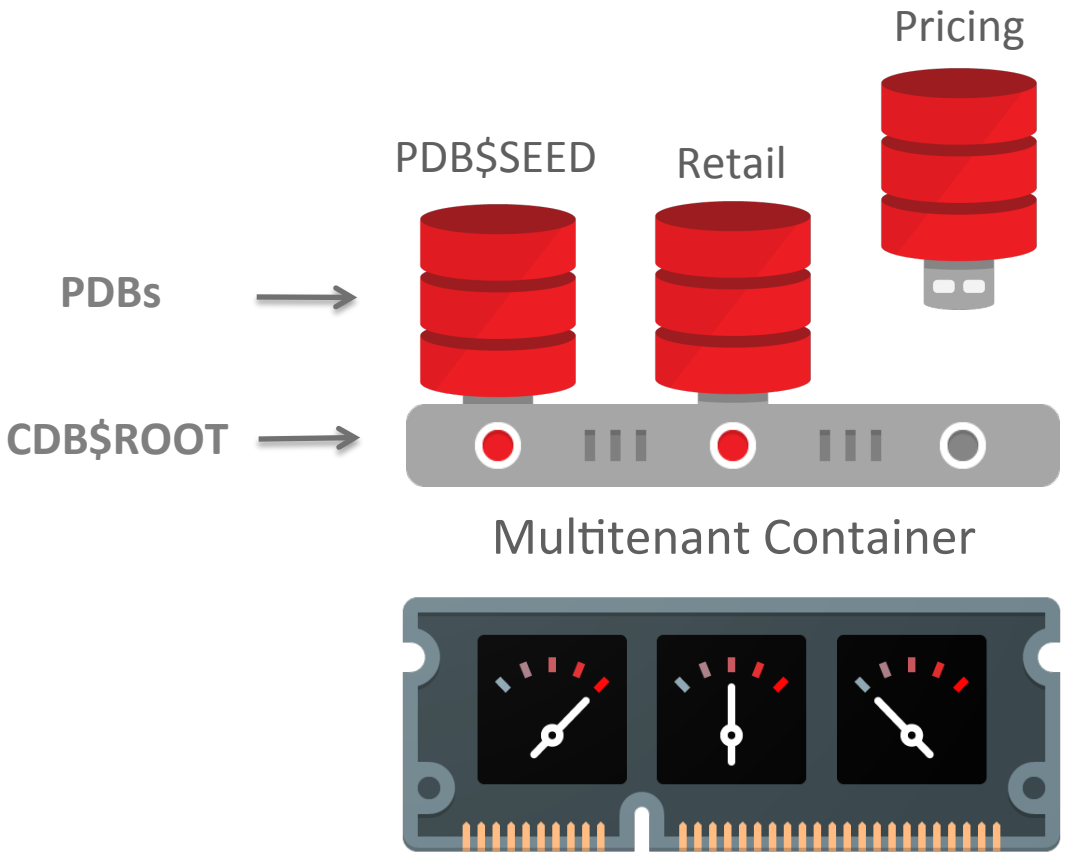
System Resources





Oracle Multitenant

New architecture for consolidating databases and simplifying operations



Self-contained PDB for each application

- Portability (via pluggability)
- Rapid provisioning (via clones)
- Applications run unchanged

Common operations performed at Root level

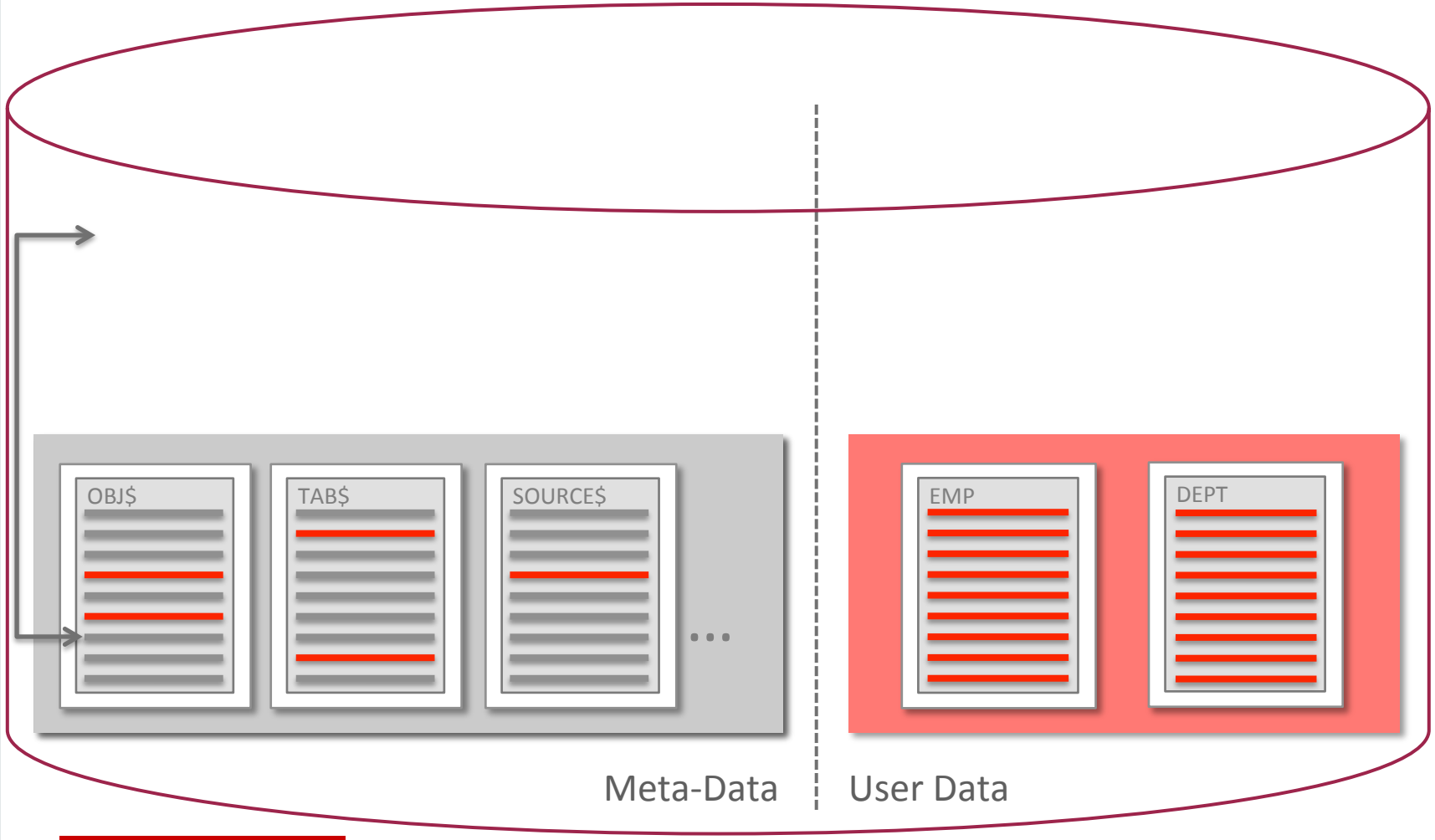
- Manage many as one (upgrade, backups, HA)
- Granular control when appropriate

Shared memory and background processes

- More applications per server

Oracle Data and User Data

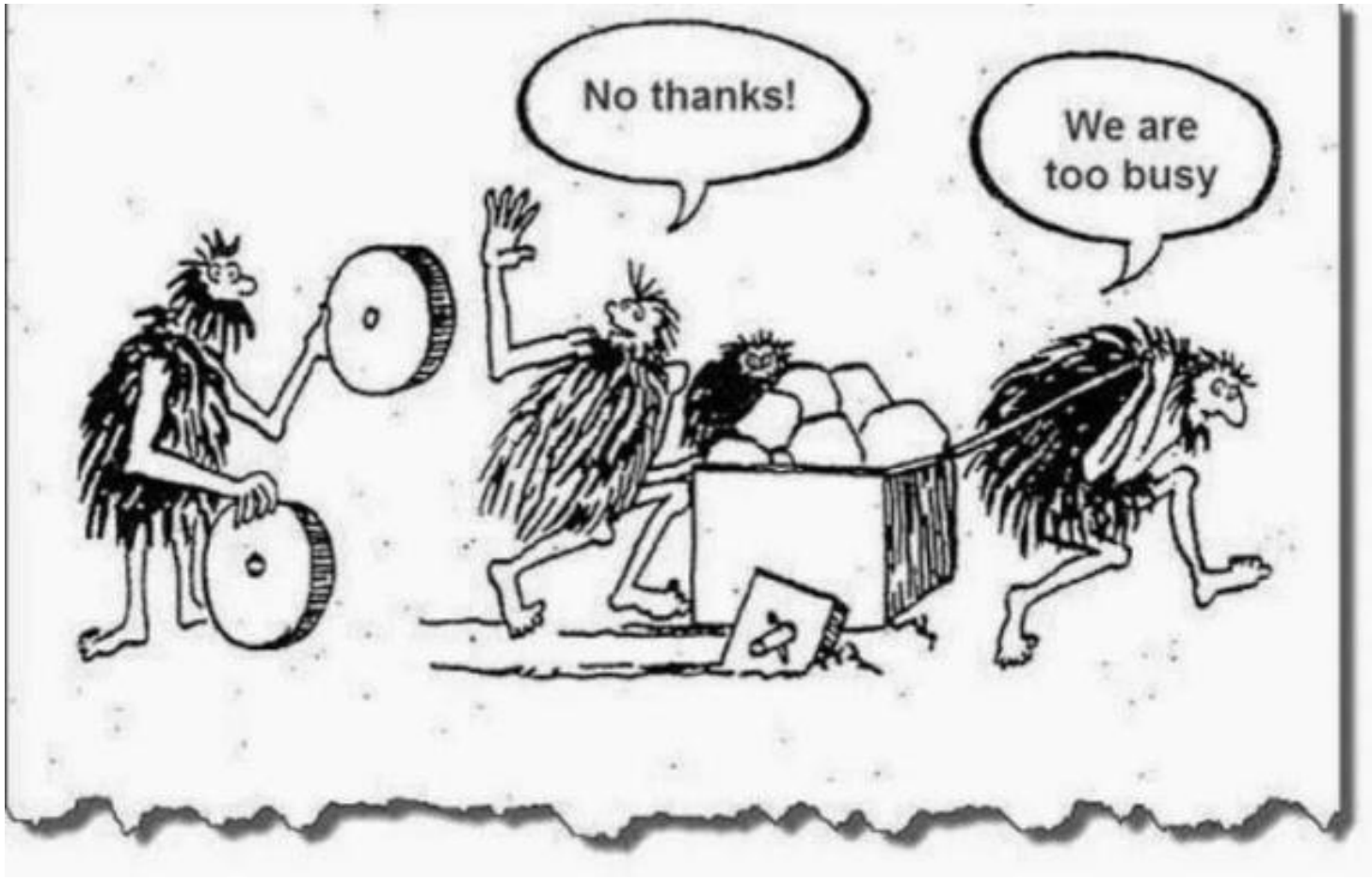
Before 12.1: Oracle and user data intermingle over time



- New database contains Oracle meta-data only
- Populate database with user data
  - Oracle and customer meta-data intermingled
  - Portability challenge!
- Multitenant fix: *Horizontally-partitioned data dictionary*
  - Only Oracle-supplied meta-data remains in root

Planning & Consolidation Target

Are You Too Busy to Plan ?



Planning(1/2)

Identify – Who are my candidates ?

- Define consolidation criteria
  - SLAs – Performance and/or availability
    - Workload
    - Line of Business
    - Character set and/or Time zone
- Positions you to more easily define a DBaaS Service Catalog

How do I size my server ?

- Consider
  - Aggregate compute resource requirements (*CPU, Memory, IO, Network, Storage*)
  - Workload and targeted consolidation density
  - Estimate and Validate

Planning (2/2)

Define Management Policies

- Where/when to Manage Many as One
- SLAs and HA solution
- Provisioning policies
- DB Resource Management Policies

Evaluate PDB Candidate Uncertainties

- Deploy as Single Tenant
- Position the PDB for Multitenant
  - Clone and unplug/ plug features available

Analyze the Workloads

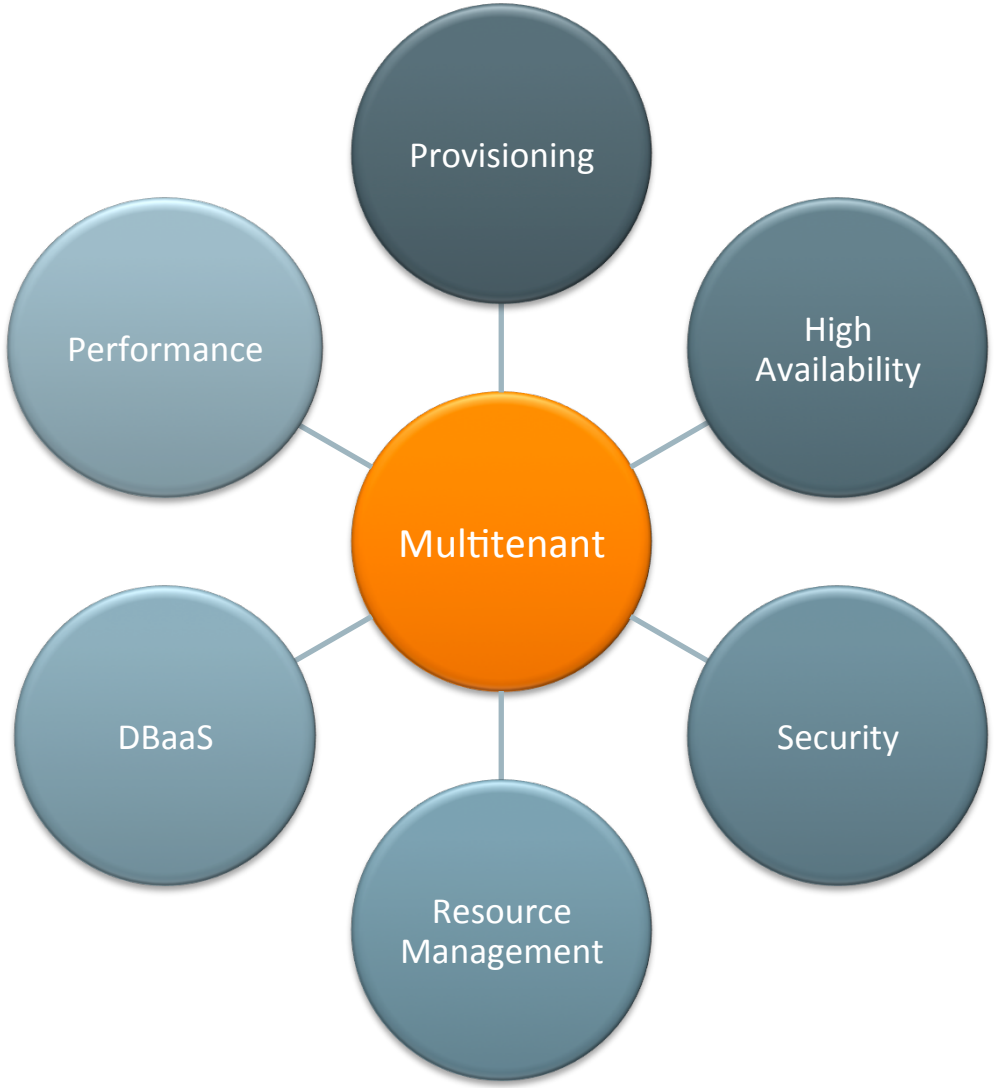
- Identify compatibility and conflicts
- Identify oversubscription opportunities

Cigna Health – CON6379: Oracle Multitenant Customer Success Story *(class of 2014)*

	Post Consolidation
Server Configs	10x reduction
Servers	3x reduction
Oracle Versions	3 → 1
OS Versions	8 → 1
Hardware Type	1 Standard Version
HA Solution	1 Standard HA Solution
Physical Servers	50% reduction
Databases	7x reduction
DB Character Set	1 Standard Character Set

Tactical View of Multitenant

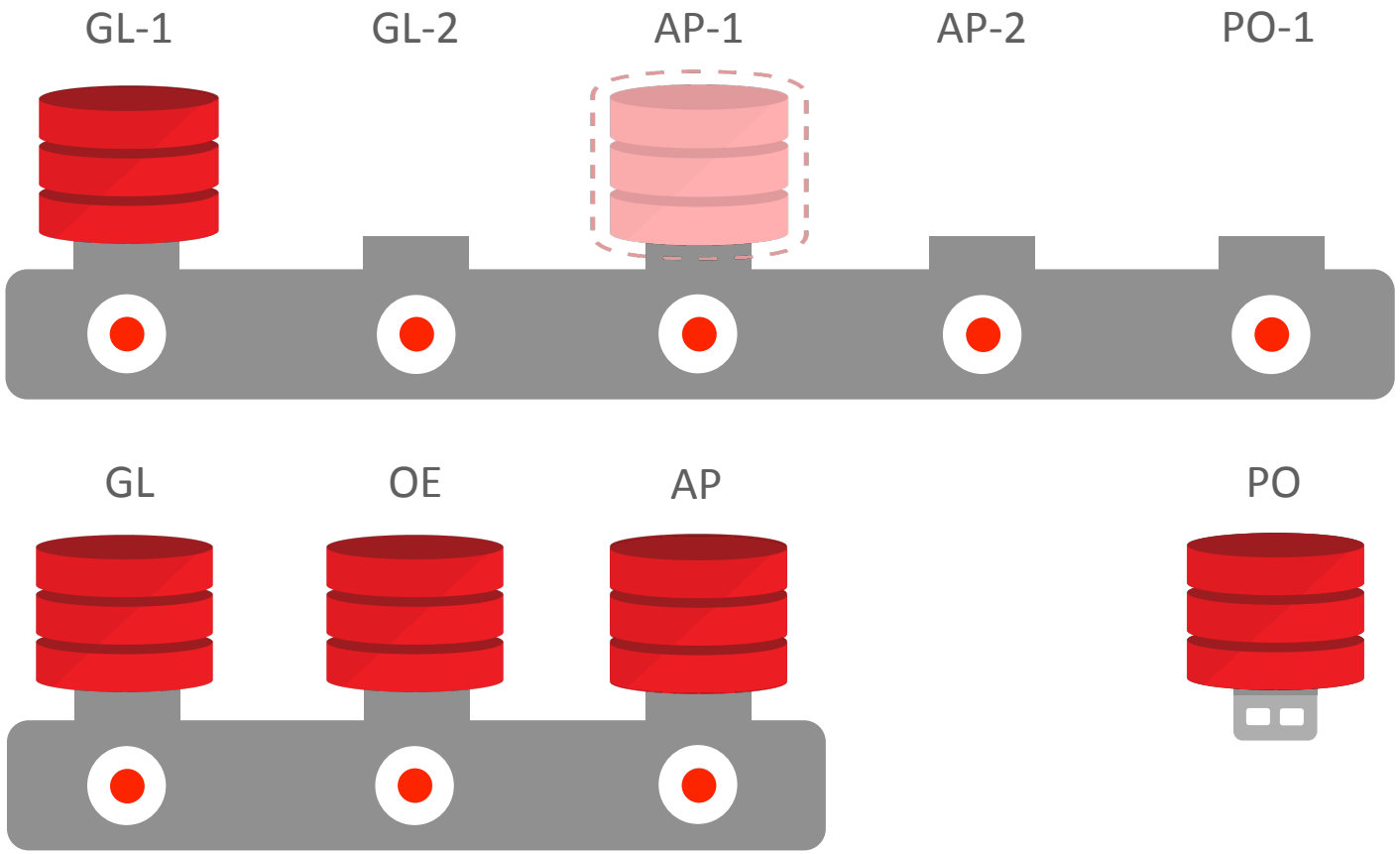
Theme	
Provisioning	Creating a CDB and Snapshot clones
DBaaS	Provisioning, Self-Service, Adopt to changing workloads
Security, Isolation	Configuring Multitenant security for various use cases
Resource Manager	Shares, caps, dynamic capabilities
RAC	PDB/Node Affinity, portability, scale out
Performance	Manage Performance



Provisioning & Cloning

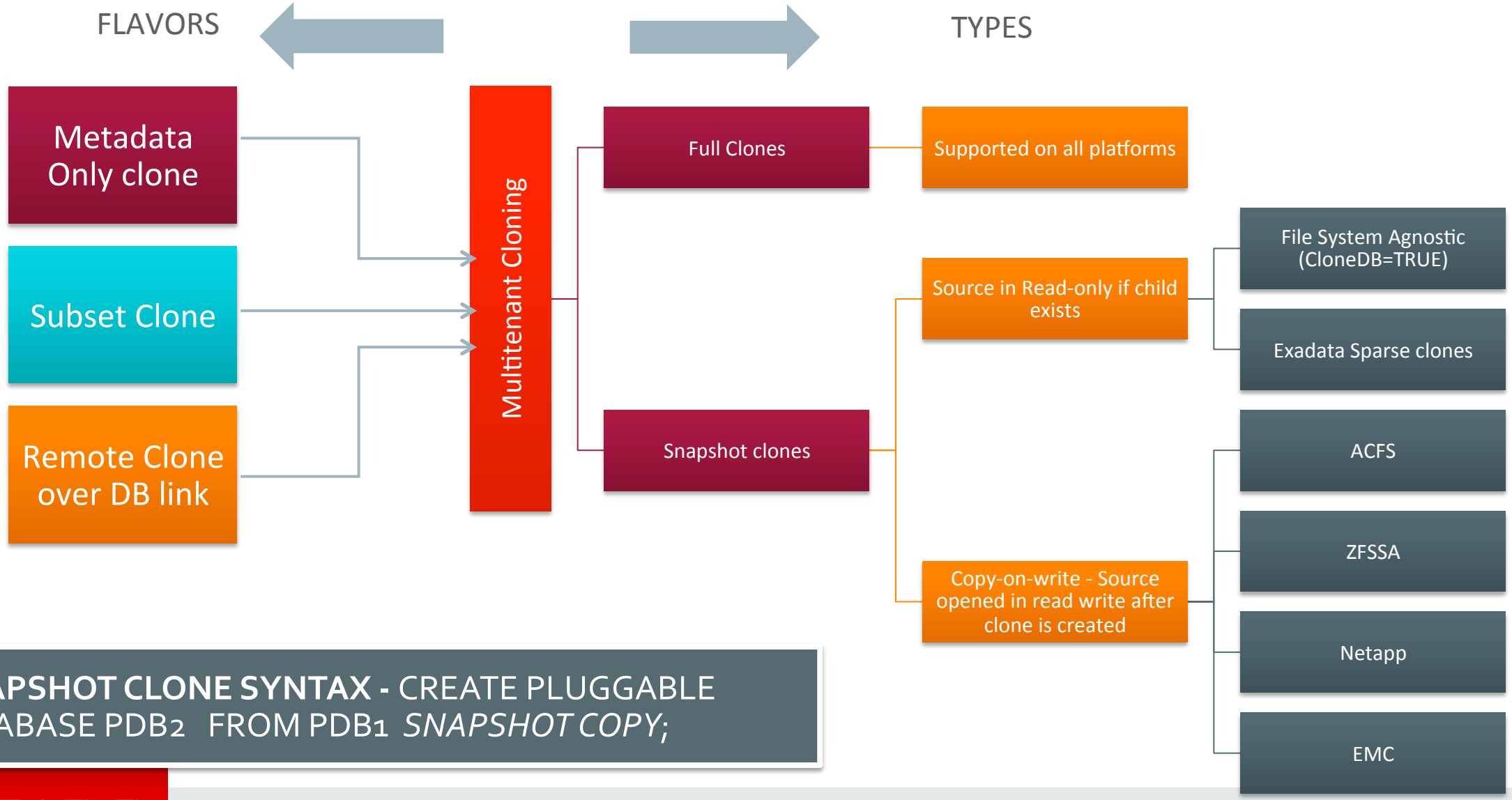
Multitenant for Provisioning

Fast cloning of PDBs



- PDBs can be cloned from within the same CDB
- PDBs can be cloned from remote CDBs
- PDBs can be cloned from non-CDBs
- Thinly provision *snapshot* clones in seconds

Multitenant Cloning – Types and Platforms



SNAPSHOT CLONE SYNTAX - CREATE PLUGGABLE DATABASE PDB2 FROM PDB1 *SNAPSHOT COPY*;

Deployment

CDB Creation and Configuration

Use DBCA

- GUI or Silent Mode
- Customize CDB Options
  - Customization of Database Options in a Multitenant Setup(**1616554.1**)

Standardize

- Character set
  - AL32UTF8
- Use OMF
- Operations
  - e.g. Patching schedule

Deployment - Configurations

- Size your SGA – the Divide by 2 rule of thumb
- Configure Huge Pages if SGA > 30GB
- Active process count should be ~5 x physical core
- Set SGA_TARGET to 60% of physical memory
  - Let Oracle Manage the Memory Pools
- Automatic PGA memory management
  - 20% of SGA

- Set SGA_MAX_SIZE to a high value
- Set DB_FILES, PROCESSES to anticipate growth
- We use *GLOBAL UNDO*, size your undo tablespace with consideration
- Redo
  - Minimum 4GB and size to switch max <= 10-20 mins
  - 3-4 redo groups
- Archive
  - Calibrate retention/deletion policies for backups/archive logs
- Monitor temp usage

Initialization Parameters & Miscelleneous

- For Multitenant, there are two groups of parameters:
  - Those that can be modified within a PDB
  - Those that can only be set on the CDB level
- In order to find the possible parameters you need to run:

  SELECT NAME FROM V$SYSTEM_PARAMETER WHERE ISPDB_MODIFIABLE='TRUE' ORDER BY NAME;
- PDB level parameters persist across database restarts and also across unplug/plug and clone.

- Evaluate and adjust any parameter that affects application performance
  - OPEN_CURSORS
  - OPTIMIZER settings
  - CURSOR_SHARING
  - Evaluate as consolidation density increases
- Configure clone quotas and storage limits
- Don’t modify PDB$SEED
  - Create and customize your own SEED

Management

Operational

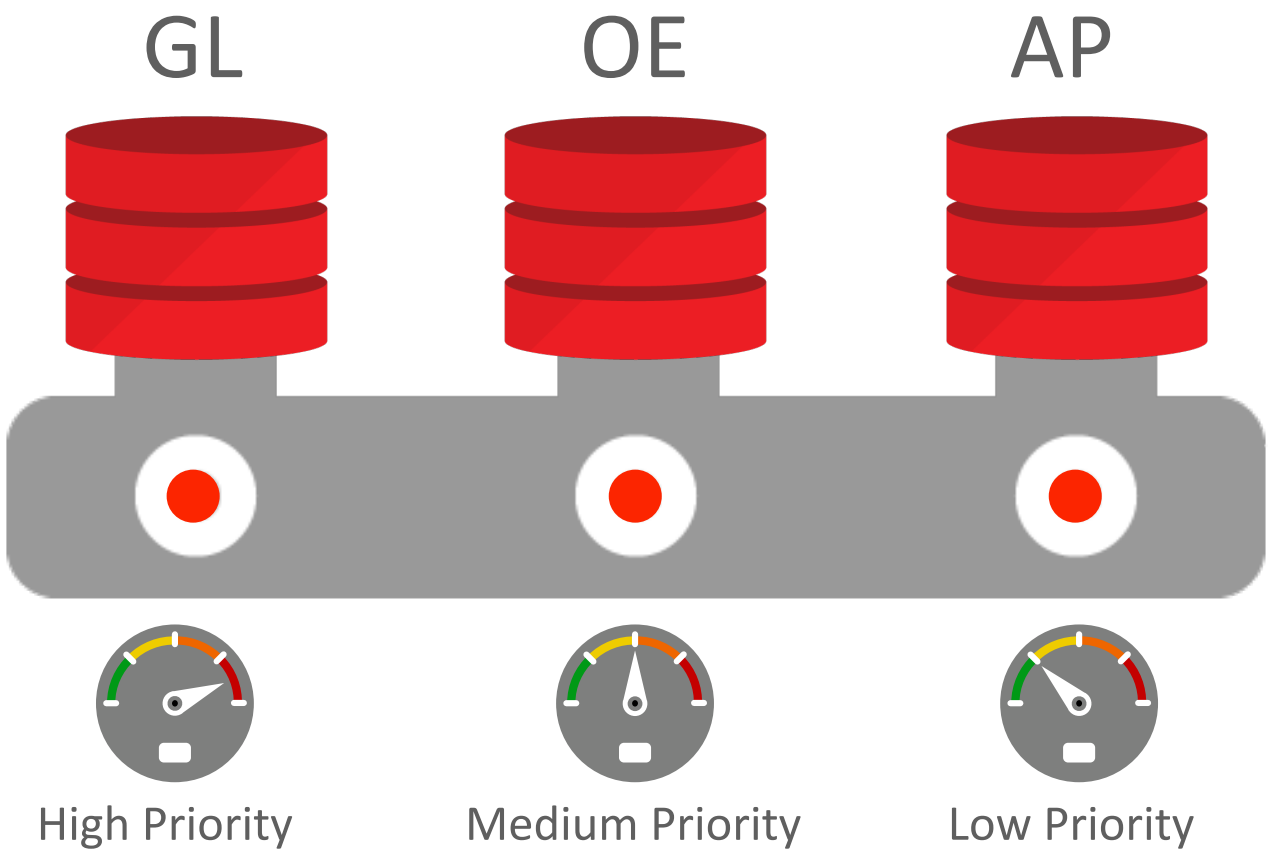
- Size SYSAUX appropriately
  - Monitor free space in CDB and in all PDBs
  - Ensure 10GB free before upgrade to 12.1.0.2
  - Look for ORA-1652 temp extent allocation errors in the mmon slaves trace files
- Monitor the Audit Directory
  - Audit can be defined in the context of the CDB or PDB
  - Monitor and purge the audit destination
- $ORACLE_HOME/rdbms/admin/catcon.pl
  - Execute SQL*Plus scripts or SQL statements in
    - a non-CDB
    - all or subset of PDBs in a CDB

- Review AutoTask scheduling
  - By default, all 3 auto tasks start at the same time in all PDBs.
  - Stagger the schedule to manage performance
  - Enable 'gather stats' and disable segment advisor and auto sql tuning advisor if they are not needed.
  - Define an DBRM policy to limit CPU
- Archive Logging
  - Pre-Consolidation: Evaluate and Size Aggregate Redo Rate
  - Size Archive Log Dest and Monitor Purging Policies
  - Calibrate disk quota for FAST RECOVER AREA per consolidation density, file sizes and target recovery window

Resource Management

Managing Shared Resources

Resource management in a multitenant environment



Resource Allocation Strategy

Default Resource Allocation

- Same allocation for all PDBs
- Specify a share and each one gets the same minimum at 100% utilization
- Each PDB can use up to 100% CPU

Specify a minimum allocation

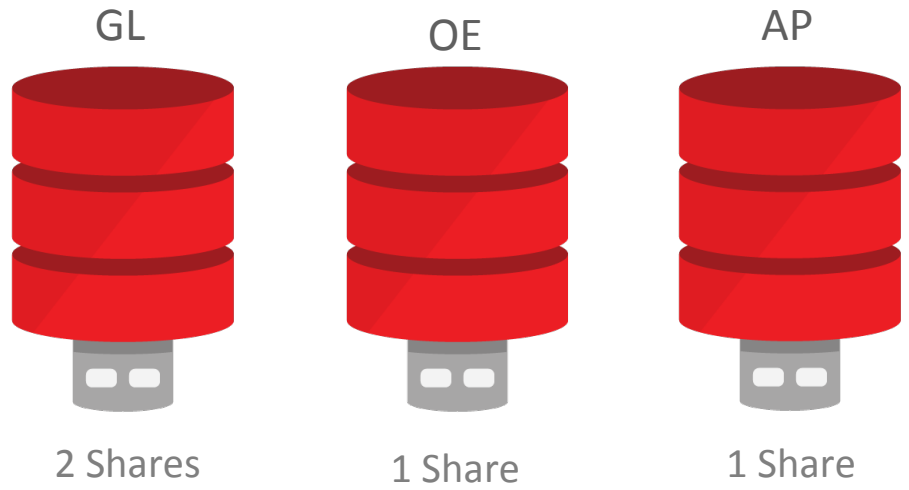
- Ensures all PDBs get a specific share of the resources
- Allows any PDB to consume any unused resources
- Kicks in at 100% resource utilization.

Specify a minimum and maximum

- Ensures all PDBs get a specific share of the resources
- Prevents a PDB from taking more than the maximum value assigned.
- May result in unused capacity

Manage CPU

A CDB Resource Plan uses *shares* to specify how CPU is distributed between PDBs

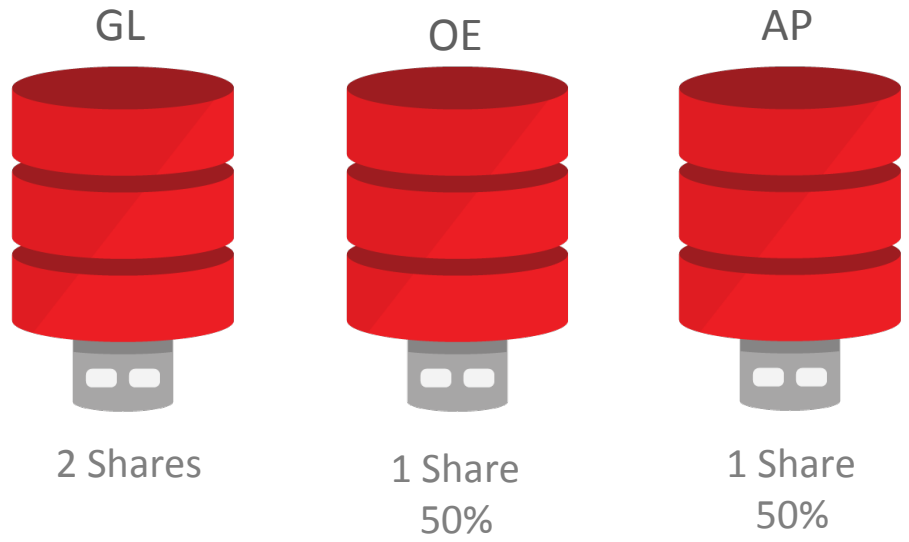


Pluggable Database	Shares	Guaranteed CPU	Maximum CPU
GL	2	2/4 = 50%	100%
OE	1	1/4 = 25%	100%
AP	1	1/4 = 25%	100%

Manage CPU

A CDB Resource Plan uses *utilization limits* to limit the CPU usage for a PDB.

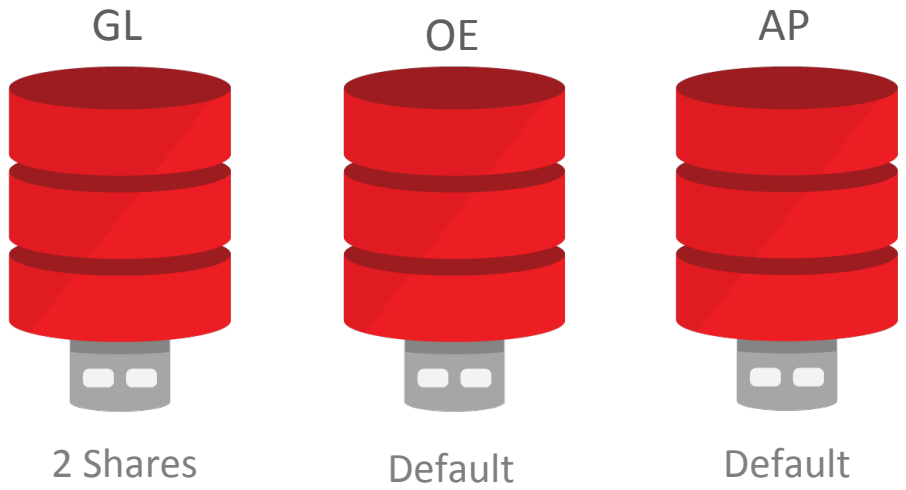
With utilization limits, your CPU may be under-utilized.



Pluggable Database	Shares	Guaranteed CPU	Utilization Limit	Maximum CPU
GL	2	2/4 = 50%		100%
OE	1	1/4 = 25%	50%	50%
AP	1	1/4 = 25%	50%	50%

Manage CPU

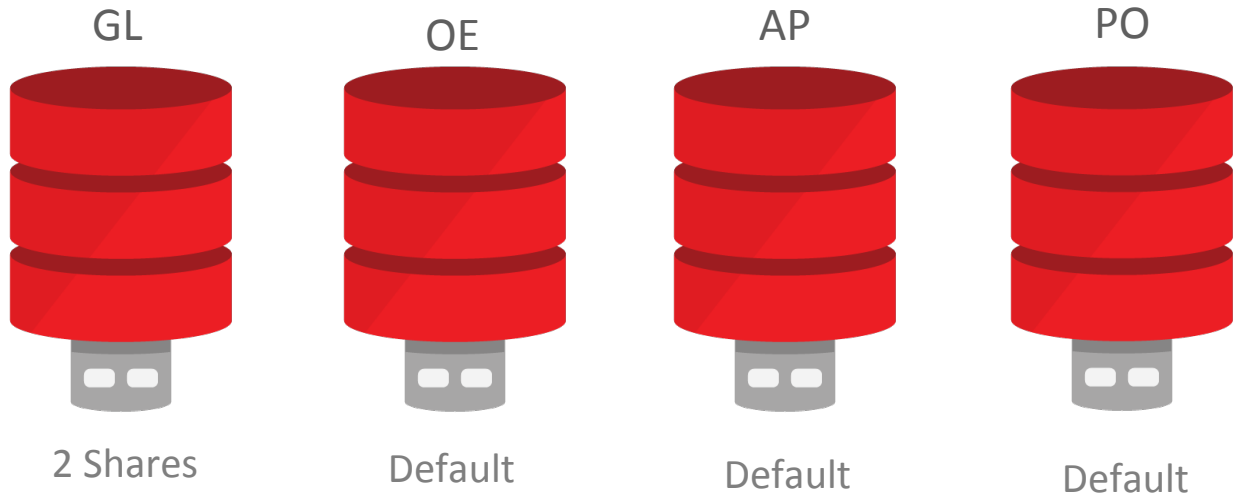
Configure a *default directive*: the default shares and utilization limit for PDBs.



Pluggable Datab	Shares	Guaranteed CPU	Utilization Limit	Maximum CPU
(Default Directive)	1		50%	
GL	2	2/4 = 50%		100%
OE	Default (1)	1/4 = 25%	Default (50%)	50%
AP	Default (1)	1/4 = 25%	Default (50%)	50%

Manage CPU

With a default directive, you don't need to modify the resource plan for each PDB plug and unplug



Pluggab…se	Shares	Guaranteed CPU	Utilization Limit	Maximum CPU
(Default D…)	1		50%	
GL	2	2/5 = 40%		100%
OE	Default (1)	1/5 = 20%	Default (50%)	50%
AP	Default (1)	1/5 = 20%	Default (50%)	50%
PO	Default (1)	1/5 = 20%	Default (50%)	50%

Parallel Statement Queuing in PDBs

Specifies the probability that this PDB will get to launch the next parallel operation

Limit the total number of parallel servers the PDB can use at a time.

Limit the DOP of the PDB's parallel operations.

PDB	Shares	Parallel Server Percentage	Parallel Degree Limit
GL	2	50	4
OE	1	50	8
AP	1	50	16

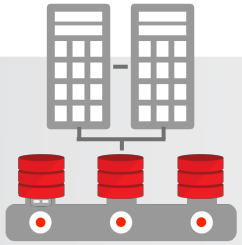
(P)DBaaS

Unprecedented Agility with Portable Pluggability

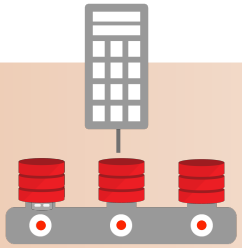
PDB migrates through SLAs as it becomes more mission critical

GOLD RAC, Data Guard

SILVER RAC



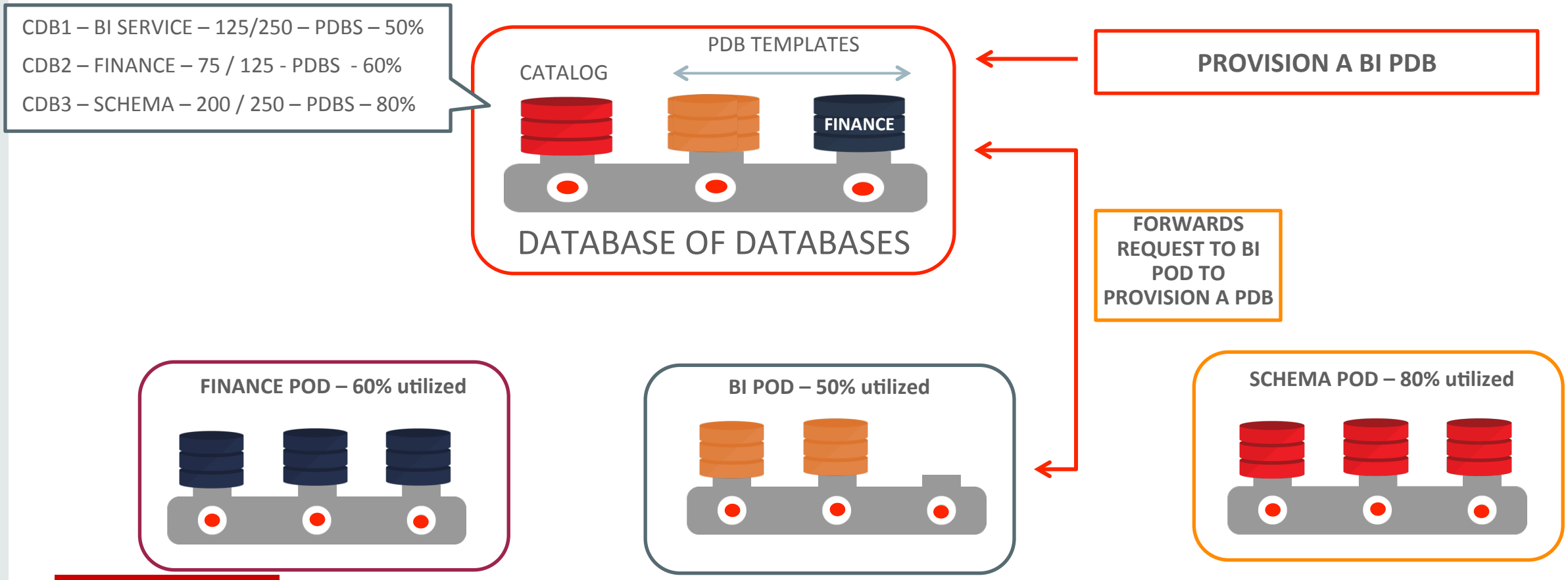
BRONZE Backups



Deployment

PDB Provisioning in PDBaaS

A central CDB with a catalog and a PDB templates



Security

Management

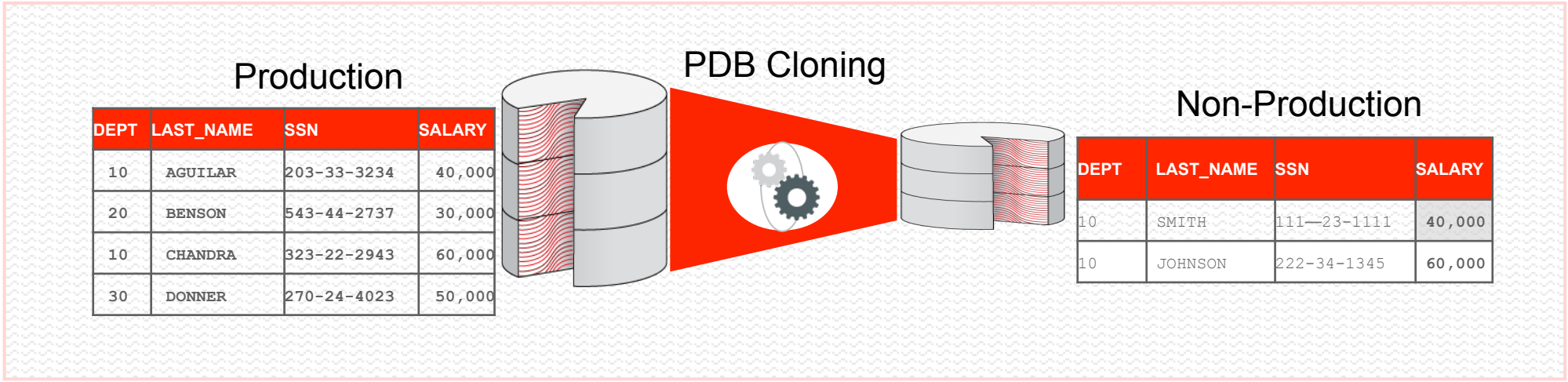
Common Users

- Understand, define and Generalize your common user & local user policies
  - Define the role separation between local and common users
  - Avoid GRANT session to common user globally
    - New PDBs plugged in inherit common user
    - Enforce local grant session to the common user
- Standardize on common user prefix
  - Avoids plug in violations if the common user is unknown to the CDB

- Do not create objects in the common users schema
- Do not change privileges on Oracle supplied common users
  - Grant the privilege within the targeted PDBs
- Validate impact of common users prior to unplug/plug operations
  - 'conflicts' are not captured in PDB compatibility checks

Management

Security Practices - After Clone Trigger



- After Clone Trigger allows to execute PL/SQL on the target
- Useful to mask sensitive information on the target
- Trigger code removed after cloning is successful

Q&A

Key Benefits

Benefit	Capability Enabled
Minimize CapEx	• More applications per server
Minimize OpEx	• Manage many as one (reduced patching!) • Standardized procedures & service levels • Enable self-service provisioning
Maximize Agility	• Snapshot cloning for development and testing • Portability through “pluggability” • Scalability with RAC
Easy	• To Adopt: Applications run unchanged • To Use: Interface is SQL

Integrated Cloud

Applications & Platform Services


ORACLE®

Upgrade and Plugin as PDB

- Upgrade or use existing 12.1 non-cdb
- Start database read-only
- Create XML description file

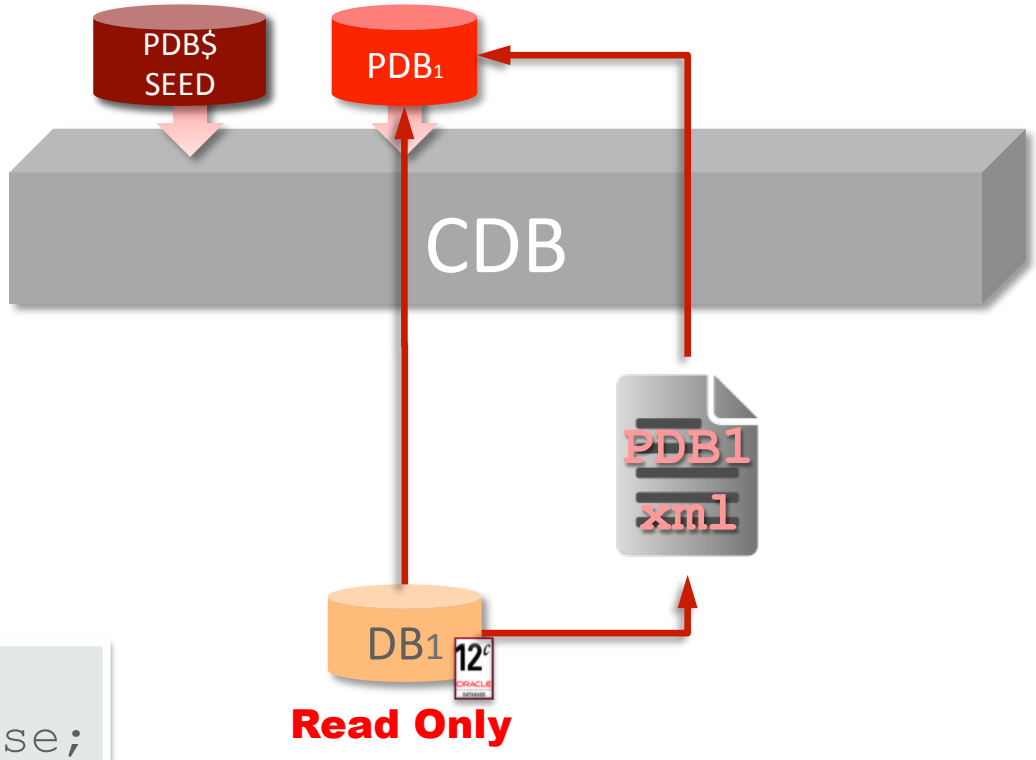
```
exec DBMS_PDB.DESCRIBE('PDB1.xml');
```

- Shutdown database
- Run PDB compatibility check
- Plugin database

```
create pluggable database PDB1
using ('PDB1.xml') nocopy tempfile reuse;
```

- Convert to PDB

```
start ?/rdbms/admin/noncdb_to_pdb.sql
```



Create Pluggable Database

- **Creating from scratch**

```
SQL> create pluggable database PDB1 admin user admin identified by password;
```

- **Plugging a New PDB**

```
SQL> create pluggable database PDB1 using '/tmp/pdb1.xml';
```

- **Using file_name_convert**

```
SQL> create pluggable database PDB1 using '/tmp/pdb1.xml'
     file_name_convert = ('/u01/oradata/cdb1',' /u01/oradata/cdb2');
```

Create Pluggable Database

- **Unplugging and plugging a Pluggable database**

```
At the source

SQL> alter pluggable database PDB1 close;

SQL> alter pluggable database PDB1 unplug into '/tmp/pdb1.xml';

SQL> drop pluggable database PDB1 keep datafiles;

At the target

SQL> create pluggable database PDB1 using '/tmp/pdb1.xml';
```

Cloning Pluggable Database

- **Full clone from another PDB.**

```
SQL> create pluggable database PDB1 from PDB2;
```

- **Snapshot clone of a PDB**

```
SQL> create pluggable database PDB1 from PDB2 snapshot copy;
```

- **Subset clone using USER_TABLESPACES clause**

```
SQL> create pluggable database PDB1 from PDB2
     user_tablespaces = ('tbs1','tbs2');
```

Cloning Pluggable Database

- **Metadata only clone**

```
SQL> create pluggable database PDB1 from PDB2 nodata;
```

- **Remote clone of a PDB**

```
SQL> create pluggable database PDB1 from PDB2@CDB2_link;
```

- **Remote snapshot clone of PDB where CDBs share the same storage**

```
SQL> create pluggable database PDB1 from PDB2@CDB2_LINK
     snapshot copy;
```

Cloning Pluggable Database

- **Storage Agnostic clone**

```
SQL> Alter system set clonedb=TRUE scope=spfile;
-- Restart database and source stays in read-only
SQL> create pluggable database PDB1 from PDB2 snapshot copy;
```

- **Closing a pluggable database**

```
SQL> alter pluggable database PDB1 close <<immediate>>;
```

- **Dropping a pluggable database**

```
SQL> drop pluggable database PDB1 including datafiles / Keep datafiles;
```