ORACLE

Move to Oracle Database 23ai

Everything you need to know about Oracle Multitenant - Part 2

Photo by Lili Kovac on Unsplash

Oracle

DBAs

run the world

ROY SWONGER
Vice President
Database Upgrade, Utilities & Patching

royfswonger

MIKE DIETRICH

Senior Director Product Management
Database Upgrade, Migrations & Patching

mikedietrich

@mikedietrichde

https://mikedietrichde.com





DANIEL OVERBY HANSEN
Senior Principal Product Manager
Database Upgrade, Migrations & Patching

dohdatabase

@dohdatabase

https://dohdatabase.com





RODRIGO JORGE
Senior Principal Product Manager
Database Upgrade, Migrations & Patching

rodrigoaraujorge

@rodrigojorgedba

https://dbarj.com.br/en





ALEX ZABALLA
Distinguished Product Manager
Database Upgrade, Migrations & Patching

alexzaballa

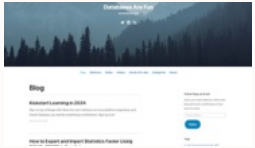
@alexzaballa

https://alexzaballa.com

Find Slides and Much More on Our Blogs

MikeDietrichDE.com

Mike.Dietrich@oracle.com



dohdatabase.com

Daniel.Overby.Hansen@oracle.com

DBArj.com.br

Rodrigo.R.Jorge@oracle.com

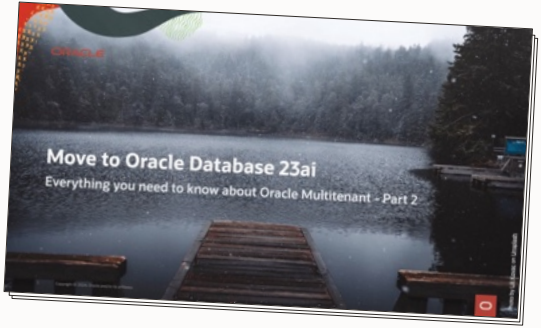
AlexZaballa.com

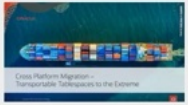
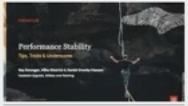
Alex.Zaballa@oracle.com

Download the Slides

https://dohdatabase.com/slides

https://MikeDietrichDE.com/slides



Web Seminar	Slides
Episode 16 (replaces Episode 1 from Feb 2021) Oracle Database Release and Patching Strategy for 19c and 23c *115 minutes – May 10, 2023*	
Episode 17 From SR to Patch – Insights into the Oracle Database Development process *55 minutes – June 22, 2023*	
NEW Episode 18 Cross Platform Migration – Transportable Tablespaces to the Extreme *145 min – February 22, 2024*	
Episode 2 AutoUpgrade to Oracle Database 19c *115 minutes – Feb 20, 2021*	
Episode 3 Performance Stability, Tips and Tricks and Underscores *120 minutes – Mar 4, 2021*	
Episode 4 Migration to Oracle Multitenant	

Recorded Web Seminars

https://MikeDietrichDE.com/videos

More than 35 hours of technical content, on-demand, anytime, anywhere

Operations

Connecting to a PDB

```
--A common user may switch into a different container, including root

alter session set container=pdb1;
```

```
create pluggable database blue ...
```

```
lsnrctl status

...

Service "blue" has 1 instance(s).
  Instance "CDB23", status READY, has 1 handler(s) for this service...
```

```
sqlplus <user>@<hostname>/blue


<alias_name>=(DESCRIPTION=
   (ADDRESS=
      (PROTOCOL=TCP)(HOST=<hostname>)(PORT=1521)
   )
   (CONNECT_DATA=
      (SERVER=DEDICATED)
      (SERVICE_NAME=blue)
   )
)
```

Keep PDB names unique on the entire host - preferably in your entire environment

- Avoid service name collision

MAA guidelines recommend creating your own service

- Avoid the default service

```
--For single instance databases
alter session set container=blue;
exec dbms_service.create_service('SALES', 'SALES');
exec dbms_service.start_service('SALES', NULL);
```

```
--For single instance databases
alter session set container=blue;
exec dbms_service.create_service('SALES', 'SALES');
exec dbms_service.start_service('SALES', NULL);

--For single instance databases (Oracle Restart) or RAC databases
srvctl add service -d $ORACLE_UNQNAME -service SALES –pdb blue
srvctl start service -d $ORACLE_UNQNAME -service SALES
```

```
sqlplus <user>@<hostname>/sales


<alias_name>=(DESCRIPTION=
   (ADDRESS=
      (PROTOCOL=TCP)(HOST=<hostname>)(PORT=1521)
   )
   (CONNECT_DATA=
      (SERVER=DEDICATED)
      (SERVICE_NAME=sales)
   )
)
```

```
select pdb, name from gv$services order by name;

PDB        NAME
---------- --------------------------------
BLUE       blue
BLUE       SALES
```

What about `ORACLE_PDB_SID`?

```
$ export ORACLE_SID=CDB23
$ export ORACLE_PDB_SID=BLUE
$ sqlplus / as sysdba

SQL*Plus: Release 23.0.0.0.0 - Production on Fri Jun 14 07:54:05 2024
Version 23.4.0.24.05

Copyright (c) 1982, 2024, Oracle.  All rights reserved.


Connected to:
Oracle Database 23c EE Extreme Perf Release 23.0.0.0.0 - Production
Version 23.4.0.24.05

SQL> show con_name

CON_NAME
------------------------------
PDB1
```

This is only documented for use with Oracle E-Business Suite

- Not documented for use elsewhere

Using `ORACLE_PDB_SID`

- Doesn't work on Windows

- Produces no error when
  - PDB is not started
  - PDB does not exist
  - You mistype the PDB name
  - Database is not started in normal mode

  You end up in root instead - silently

Using ORACLE_PDB_SID

- MOS note: Performing bequeath direct connections to PDB as SYS and SYSTEM (Doc ID 2728684.1)
- Blog post: Pitfalls: Connect to a PDB directly with ORACLE_PDB_SID

What about `TWO_TASK`?

`TWO_TASK` is just a *shortcut* to a TNS alias

- Does not add any value when connecting to a PDB

```
--TWO_TASK can hold a TNS alias.
--Use TNS connection, instead of a bequeath connection
export TWO_TASK=my_alias
sqlplus system

--It's basically the same as using @ to connect over TNS
sqlplus system@my_alias
```

Migrating your *non-CDB scripts*

```
$ cat /etc/crontab

00 * * * * oracle important.sh
```

```
$ cat /etc/crontab

00 * * * * oracle important.sh

$ cat important.sh

export ORACLE_SID=NONCDB1
sqlplus / <<EOF
   exec shop.orders.process;
   exec shop.sales.calculate;
   exec shop.shipping.track;
EOF
```

You must modify
your scripts and procedures

```
$ cat /etc/crontab

00 * * * * oracle important.sh

$ cat important.sh

export ORACLE_SID=CDB1
sqlplus / <<EOF
   alter session set container=blue;
   exec shop.orders.process;
   exec shop.sales.calculate;
   exec shop.shipping.track;
EOF
```

```
$ cat /etc/crontab

00 * * * * oracle important.sh

$ cat important.sh

sqlplus /@blue <<EOF
   exec shop.orders.process;
   exec shop.sales.calculate;
   exec shop.shipping.track;
EOF
```

Use Secure External Password Store

Scripts

- *Many-as-one* principle
- `catcon.pl`
- `DBMS_SCHEDULER`
- Enterprise Manager

Execute scripts in PDBs using `catcon.pl`

```
$ cat important.sql

exec shop.orders.process;
exec shop.sales.calculate;
exec shop.shipping.track;
```

```
$ cat important.sql

exec shop.orders.process;
exec shop.sales.calculate;
exec shop.shipping.track;

$ cd $ORACLE_HOME/rdbms/admin
$ perl catcon.pl -b important -c blue important.sql
```

```
$ perl catcon.pl \
          -u appuser/apppwd
          -b important
          -c blue,red,green,yellow,purple,orange
          -n 3
          -e -l /tmp/important_log
          important.sql
```

```
--Use command line help to see all options

perl catcon.pl -help
```

Word of caution about `_oracle_script`

```
$ cd $ORACLE_HOME/rdbms/admin
$ grep -i "_oracle_script" * | wc -l

188
```

_oracle_script

- Default value is **FALSE**
- Undocumented
- Used internally by Oracle
- "_ORACLE_SCRIPT"=TRUE PARAMETER Should not be Invoked by Users (Doc ID 2378735.1)

```
conn appuser/apppwd

create table appuser.t1 (c1 number);
alter session set "_oracle_script"=true;
create table appuser.t2 (c1 number);
```

```
conn appuser/apppwd

create table appuser.t1 (c1 number);
alter session set "_oracle_script"=true;
create table appuser.t2 (c1 number);

select object_name, oracle_maintained from user_objects;

OBJECT_NAME ORACLE_MAINTAINED
----------- -----------------
T1          N
T2          Y
```

Do not use `_oracle_script` to customize *PDB$SEED*

- Implement changes in *afterburner* script

Do not change `_oracle_script` except under guidance of Oracle Support

Auto-starting PDBs

```
--A pluggable database does not start together with the CDB
--You must instruct the CDB to start it
--by saving state when the PDB is open

create pluggable database blue ... ;
alter pluggable database blue open;
alter pluggable database blue save state;
```

```
--The view contains the list of PDBs with a saved state
--Any PDB not in this view will not auto-start

select con_name, state, restricted from dba_pdb_saved_states;

CON_NAME   STATE RESTRICTED
---------- ----- ----------
BLUE       OPEN  NO
```

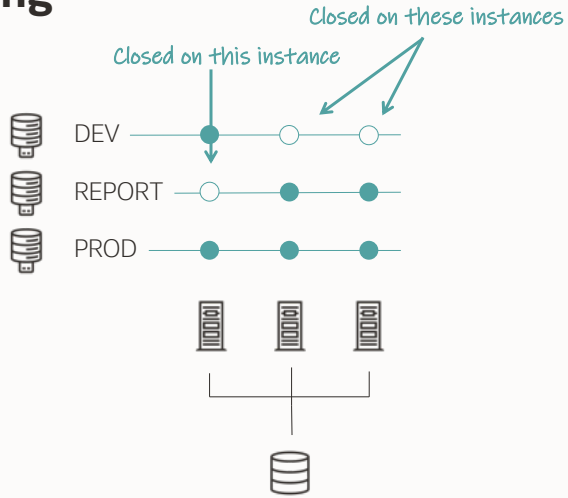
Use Oracle Clusterware to start PDBs in Oracle Restart and Oracle RAC

- Discard saved state and rely on Oracle Clusterware

```
--If a service depends on a PDB, Clusterware starts the PDB for you
--regardless of the saved state

srvctl add service ... -pdbs BLUE
```

PDB Subsetting increase flexibility and allow you to use resources wisely

PDB Subsetting



```
--By default, a service opens on all instances
--The service brings up the PDB on CDB restart
srvctl add service ... -pdb PROD

--Restrict a service to start on select instances only
srvctl add service ... -pdb REPORT -preferred inst2,inst3
srvctl add service ... -pdb DEV -preferred inst1
```

Work across PDBs with `CONTAINERS` clause

```
select con_id, tablespace_name, status
from   containers(dba_tablespaces);

CON_ID TABLESPACE_NAME STATUS
------ --------------- ---------
     1 SYSTEM          ONLINE
     1 SYSAUX          ONLINE
     1 UNDOTBS1        ONLINE
     1 TEMP            ONLINE
     1 USERS           ONLINE
     3 SYSTEM          ONLINE
     3 SYSAUX          ONLINE
     3 UNDOTBS1        ONLINE
     3 TEMP            ONLINE
     3 USERS           ONLINE
```

```
select con_id, tablespace_name, status
from   containers(dba_tablespaces)
where  con_id = 3;

CON_ID TABLESPACE_NAME STATUS
------ --------------- ---------
     3 SYSTEM          ONLINE
     3 SYSAUX          ONLINE
     3 UNDOTBS1        ONLINE
     3 TEMP            ONLINE
     3 USERS           ONLINE
```

```
insert into containers(sh.sales)
       (con_id, country_name, amount)
values (7, 'Canada', 3000);

update containers(sh.sales)
set    country_name = 'USA'
where  con_id in (7,8);
```

Tighten security with PDB Lockdown Profiles

PDB Lockdown Profiles

You can restrict

1. Features
2. Options
3. Statements

PDB Lockdown Profiles

1 Features

AWR
Network access
File access
OS access
... and more

```
alter lockdown profile sec_profile disable feature=('NETWORK_ACCESS');
```

PDB Lockdown Profiles

2 Options

Partitioning
Database queuing

```
alter lockdown profile sec_profile disable option=('PARTITIONING');
```

PDB Lockdown Profiles

3 Statements

```
alter database
alter pluggable database
alter session
create database link
```

... and more

```
alter lockdown profile sec_profile
   disable statement = ('ALTER PLUGGABLE DATABASE')
   clause all except = ('DEFAULT TABLESPACE');
```

PDB Lockdown Profiles

3 Statements

```
alter database
alter pluggable database
alter session
create database link
```

... and more

```
alter lockdown profile sec_profile
   disable statement = ('ALTER SYSTEM')
   clause = ('SET')
   option = ALL EXCEPT ('PLSQL_WARNINGS', 'PLSQL_DEBUG');
```

```
--In root you can define the default lockdown profile
alter session set container=cdb$root;
alter system set pdb_lockdown=sec_profile;

--In a PDB you can override the default
--and set a specific profile
alter session set container=pdb1;
alter system set pdb_lockdown=very_sec_profile;
```

Security

Tighten security even more:

- Allow a PDB to only write to a certain part of the file system

```
create pluggable database ... path_prefix = '/pdbs/blue/'
```

- Ensure a PDB interacts with OS using a specific user

```
alter system set pdb_os_credential=<credential_name>
```

Avoid *noisy neighbors*

- Allow sharing resources
  but everyone must get a fair share

Method 1

Instance caging

- Most simple
- Define `CPU_COUNT` for each PDB
- Hard limit

Method 1

8 CPUs

CPU_COUNT=3

CPU_COUNT=2

CPU_COUNT=2

All non-CDBs share 7 CPU

Method 1

8 CPUs

CPU_COUNT=7

CDB never uses more than 7 CPUs, despite sum of PDBs
At peak, use more resources, but never deplete the CDB

CPU_COUNT=4

CPU_COUNT=4

CPU_COUNT=4

PDBs might fight over CPUs, but each process gets a fair share

Method 2

Memory allocation

- Simple
- Define `SGA_TARGET` for each PDB
- Hard limit

Method 2

8 GB memory

SGA_TARGET=7G

PDB may never use more than 4G of shared memory

SGA_TARGET=4G

SGA_TARGET=4G

SGA_TARGET=4G

If all PDBs are active, cache management comes into play

Requires use of Automatic Shared Memory Management

- Both in CDB and PDB

Optionally, allocate minimum shared pool and buffer cache for a PDB

- Use `SHARED_POOL_SIZE` and `DB_CACHE_SIZE`

You can combine method 1 and 2

- Instance caging and memory allocation

Method 3

Simple Resource Manager

- Elaborate, yet simple to implement
- Enable CDB resource manager
- Allocate minimum shares instead of hard limits
- For advanced use cases

Method 3

8 CPUs

CPU_COUNT=7

At peak, may use up to 5 CPUs

CPU_MIN_COUNT=2

CPU_MIN_COUNT=1

CPU_MIN_COUNT=1

4 CPUs are reserved,
3 are free for all

Method 3

8 GB memory

SGA_TARGET=7G

SGA_MIN_SIZE=2G

At peak, may use
up to 4G shared memory

SGA_MIN_SIZE=1G

SGA_MIN_SIZE=1G

Requires Resource Manager at root level

```
alter session set container=cdb$root;

-- Create an empty resource manager plan with no directives
exec dbms_resource_manager.clear_pending_area;
exec dbms_resource_manager.create_pending_area;
exec dbms_resource_manager.create_cdb_plan('CDB_PLAN');
exec dbms_resource_manager.validate_pending_area;
exec dbms_resource_manager.submit_pending_area;

-- Make plan active in root to enable CDB resource manager
alter system set resource_manager_plan=CDB_PLAN;
```

Method 4

Advanced Resource Manager

- Requires additional configuration, but much greater control
- Use directives instead of shares

You can still control resources inside a PDB with Resource Manager

What about I/O?

- Exadata I/O Resource Management
- Or, `MAX_MBPS` and `MAX_IOPS`

You can run multiple CDBs on the same host and out of the same Oracle home

Inter-instance Resource Management

Shares resources like with non-CDBs:

- `CPU_COUNT`
- `SGA_MAX_SIZE`

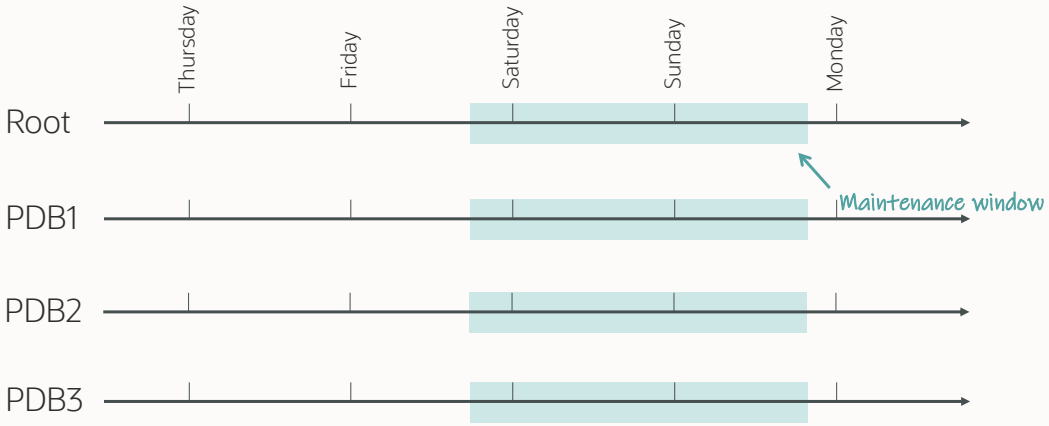
Inter-instance CPU resource manager:

- Controls resource sharing using Linux c-groups
- Check `RESOURCE_MANAGER_CPU_SCOPE`
- Exadata Database Machine and Autonomous Database

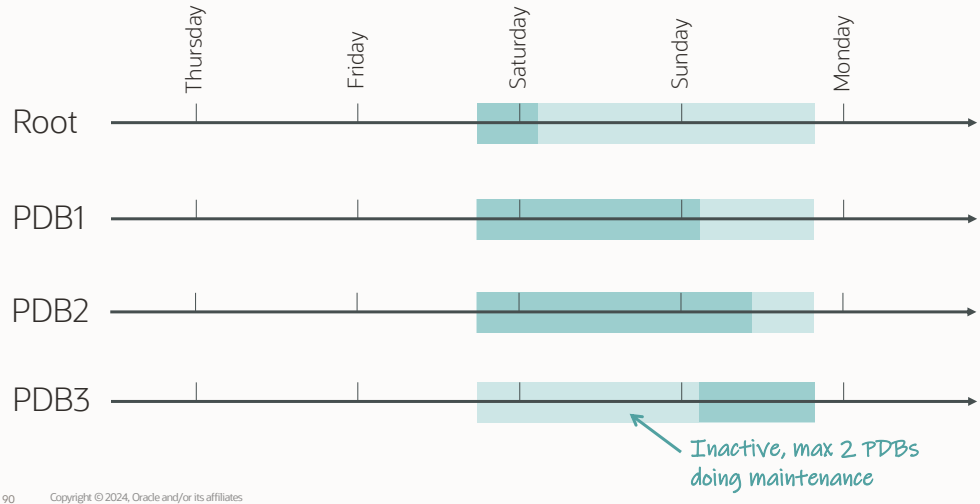
New In 23ai

A word about automated maintenance tasks

Automated Maintenance Tasks

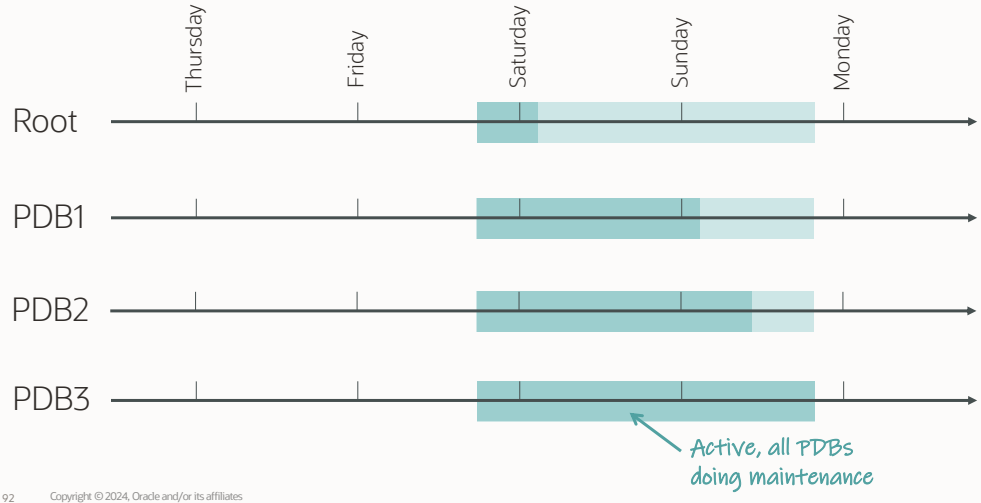


Automated Maintenance Tasks



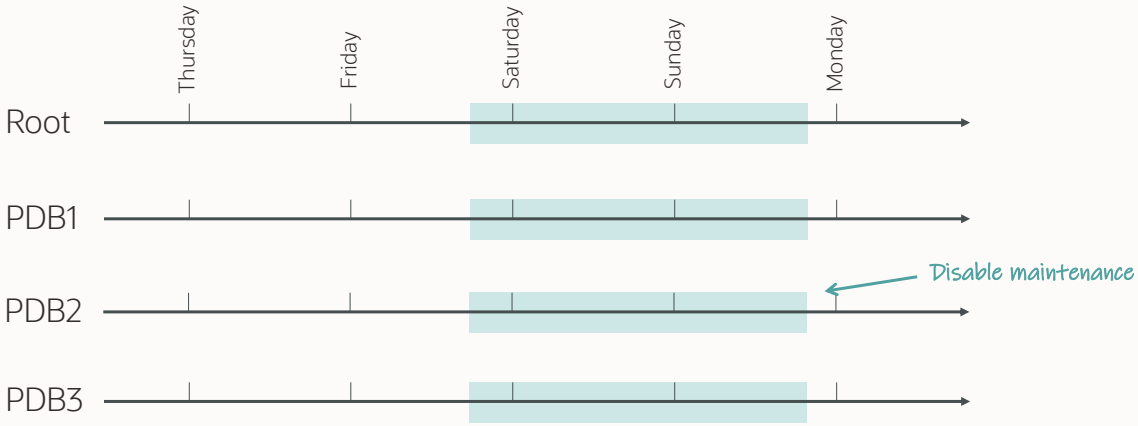
```
--Change the amount of PDBs that can run maintenance tasks at the same time
--Default value 2

alter system set autotask_max_active_pdbs=3;
```

Automated Maintenance Tasks



```
--Selectively disable maintenance tasks in a PDB
--For instance, test databases or databases that are rebuilt frequently

alter session set container=PDB2;
alter system set enable_automatic_maintenance_pdb=false;
```

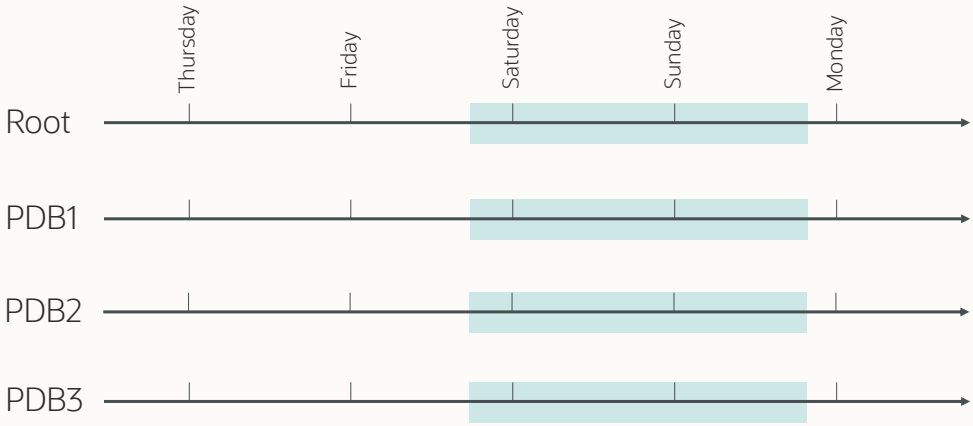
Automated Maintenance Tasks



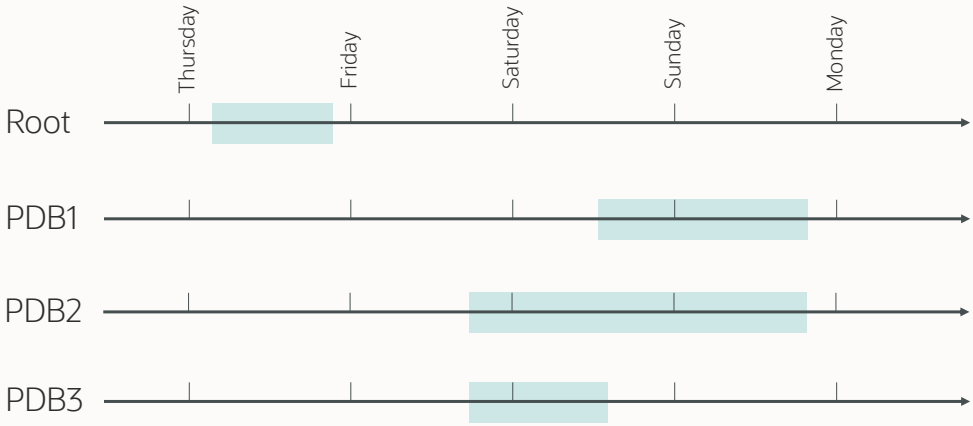
Shift maintenance windows

- Optionally, shorten maintenance windows

Automated Maintenance Tasks



Automated Maintenance Tasks



Selectively disable individual maintenance tasks using `DBMS_AUTO_TASK_ADMIN`

- Does a test database need Automatic Segment Advisor?
- Or Evolve Advisor?

Resource Manager prevents maintenance tasks from *stealing* resources from users

- Consumer group **ORA$AUTOTASK**

Using Automatic Workload Repository

The database gathers AWR data in the CDB and all PDB

- Default setting

AWR

PDB-level
PDB statistics
Some global statistics

AWR

PDB-level
PDB statistics
Some global statistics

CDB-level
Aggregated PDB statistics
All global statistics

AWR

PDB-level

```
alter session set container=pdb1;
exec dbms_workload_repository.create_snapshot;
@?/rdbms/admin/awrrpt
```

CDB-level

```
alter session set container=cdb$root;
exec dbms_workload_repository.create_snapshot;
@?/rdbms/admin/awrrpt
```

AWR

PDB-level

```
alter session set container=pdb1;
exec dbms_workload_repository.modify_snapshot_settings(...
```

CDB-level

```
alter session set container=cdb$root;
exec dbms_workload_repository.modify_snapshot_settings(...
```

AWR

Follows the PDB during clone, relocate and unplug

PDB-level Stored in *SYSAUX* tablespace in PDB

CDB-level Stored in *SYSAUX* tablespace in CDB

```
--Disable collection of AWR data for a specific PDB
--Default value: true

alter session set container=pdb1;
alter system set awr_pdb_autoflush_enabled=false;
```

```
--Use the PDB lockdown profiles to disable the AWR functionality for a PDB

alter lockdown profile profile_name disable feature=('AWR_ACCESS');
```

Get AWR of a PDB

There are different ways to query the AWR data from within a PDB

1. AWR_ROOT_SNAPSHOT
2. AWR_PDB_SNAPSHOT
3. DBA_HIST_SNAPSHOT

Get AWR of a PDB

There are different ways to query the AWR data from within a PDB

1. `AWR_ROOT_SNAPSHOT`
2. `AWR_PDB_SNAPSHOT`
3. `DBA_HIST_SNAPSHOT`

- Only show snapshots taken and stored on the CDB level.
- It will not show AWR data related to other PDBs.

Get AWR of a PDB

There are different ways to query the AWR data from within a PDB

1. AWR_ROOT_SNAPSHOT
2. AWR_PDB_SNAPSHOT
3. DBA_HIST_SNAPSHOT

```
SQL> SHOW CON_NAME
CDB$ROOT
SQL> EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT;
```

Get AWR of a PDB

There are different ways to query the AWR data from within a PDB

1. AWR_ROOT_SNAPSHOT
2. AWR_PDB_SNAPSHOT
3. DBA_HIST_SNAPSHOT

- Only show snapshots taken and stored on the PDB level.
- It will not show AWR snapshots taken on the CDB.
- By default, snapshots at the PDB level are enabled, starting in 23ai.

Get AWR of a PDB

There are different ways to query the AWR data from within a PDB

1. AWR_ROOT_SNAPSHOT
2. AWR_PDB_SNAPSHOT
3. DBA_HIST_SNAPSHOT

```
SQL> SHOW CON_NAME
PDB01
SQL> EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT;
```

Get AWR of a PDB

There are different ways to query the AWR data from within a PDB

1. AWR_ROOT_SNAPSHOT
2. AWR_PDB_SNAPSHOT
3. DBA_HIST_SNAPSHOT

- Show snapshots taken and stored **both** on the CDB and PDB level.

Get AWR of a PDB

There are different ways to query the AWR data from within a PDB

1. AWR_ROOT_SNAPSHOT
2. AWR_PDB_SNAPSHOT
3. DBA_HIST_SNAPSHOT

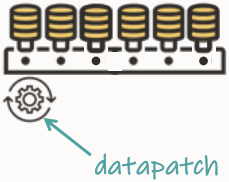
```
SQL> select snap_id, con_id
     from dba_hist_snapshot;

 SNAP_ID     CON_ID
---------- ----------
        98          0
        99          0
       100          0
       101          0
         1          3
         2          3
         3          3

7 rows selected.
```

Patching

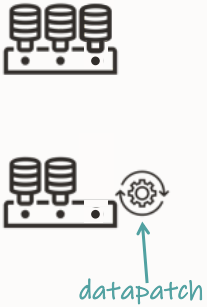
Patching Oracle home in a multitenant is the same as for non-CDB

Multitenant Patching Approaches

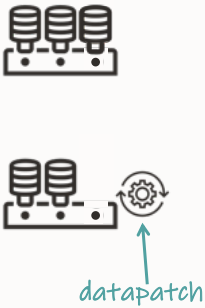
All at once



Unplug-plug



Refreshable clone



The database must be open

Only open PDBs will be patched

- **UPGRADE** mode or restricted session is not needed

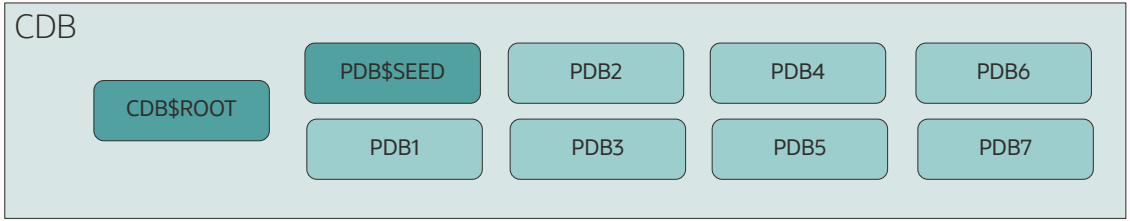
```
$ORACLE_BASE/cfgtoollogs/sqlpatch/.../sqlpatch_invocation.log
```

```
[2024-05-27 20:26:44] Installation queue:
[2024-05-27 20:26:44]   For the following PDBs: CDB$ROOT PDB$SEED
[2024-05-27 20:26:44]     No interim patches need to be rolled back
[2024-05-27 20:26:44]     Patch 35643107 (Database Release Update : 19.21.0 (35643107)):
[2024-05-27 20:26:44]       Apply from 19.1.0 Release to 19.21.0 Release_Update 230930151951
[2024-05-27 20:26:44]     The following interim patches will be applied:
[2024-05-27 20:26:44]       35648110 (OJVM RELEASE UPDATE: 19.21.0.0.231017 (35648110))
[2024-05-27 20:26:44]       35787077 (DATAPUMP BUNDLE PATCH 19.21.0.0.0)
```

Too many PDBs patched in parallel may cause contention and require lots of resources

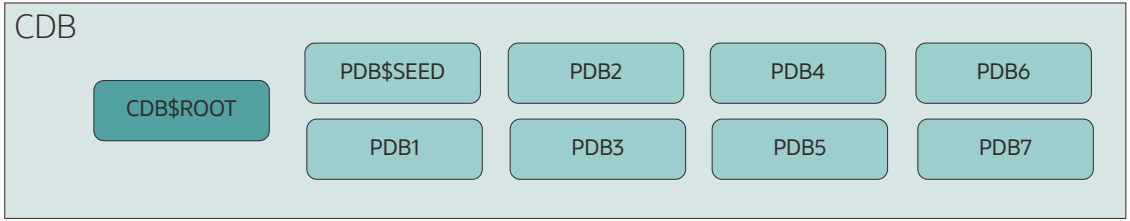
- Consider increasing the **PROCESSES** parameter

Datapatch



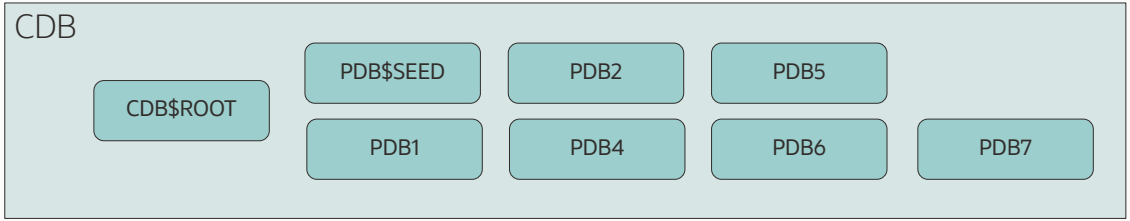
- Datapatch patches CDB$ROOT and PDB$SEED automatically

Datapatch



- Datapatch always patches CDB$ROOT first

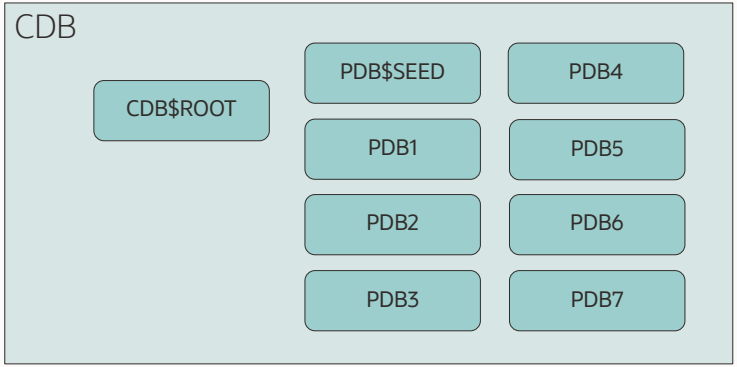
Datapatch



- Datapatch only patches open PDBs

PDB3

Datapatch

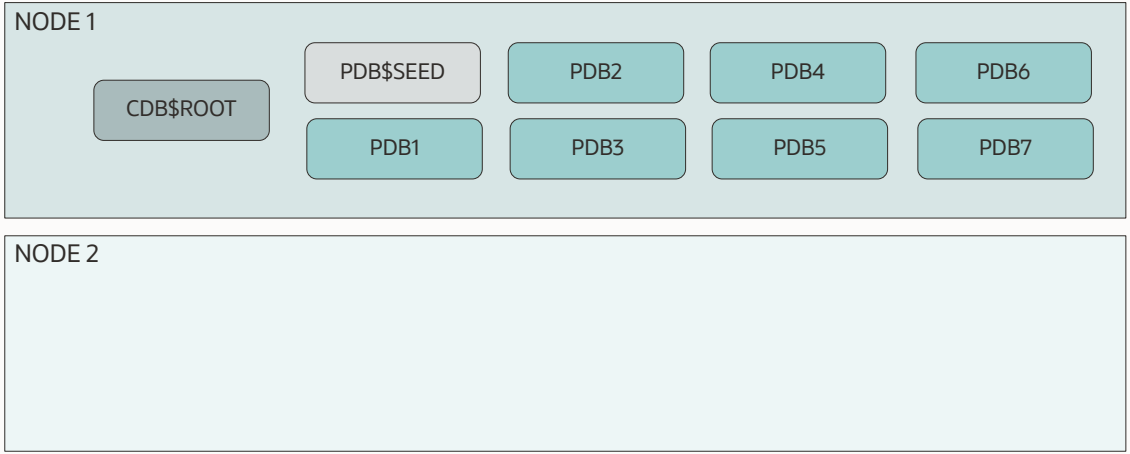


- Datapatch determines parallel degree based on CPU count

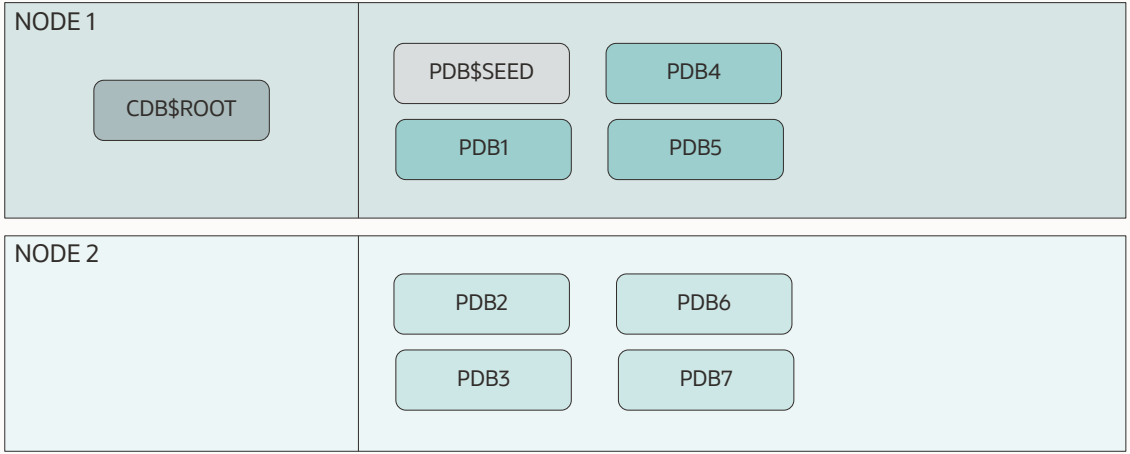
Significantly speed up patching using AutoUpgrade

- Applies to multitenant databases on RAC only

Distributed Patching



Distributed Patching



Distributed Patching

To enable distributed patching

```
$ cat RACCDB.cfg

upg1.source_home=/u01/app/oracle/product/23/dbhome_23_04
upg1.target_home=/u01/app/oracle/product/23/dbhome_23_05
upg1.sid=RACCDB
upg1.tune_setting=proactive_fixups=true,distributed_upgrade=true


$ java -jar autoupgrade.jar -config RACCDB.cfg -mode deploy
```

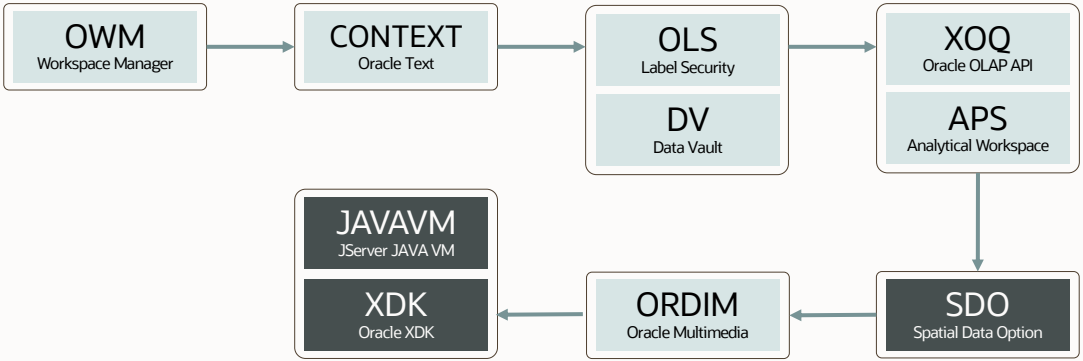
Less components, faster patching

Typical candidates:

- JAVAVM
- ORDIM
- SDO

Component Clean Up Order

If required, remove components before upgrade/plugin



Different Components in PDBs

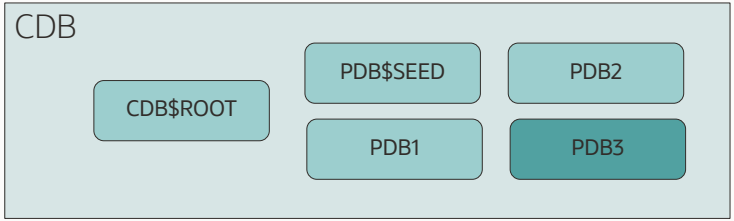
Having different components doesn't increase patching time

- Initially, having different components lead to different patching plans
  - PDB1 and PDB2 were patched sequentially with different plans
  - Timings summed up

- Since Oracle 19.16.0 this is **not** the case anymore
  - PDB1 and PDB2 can be patched in parallel
  - During patch run, *no-op* scripts will be replaced with `nothing.sql`

➔ Faster overall patching in CDB environments

Datapatch Error



- Patching fails in PDB3

```
SQL> shutdown immediate
SQL> startup

SQL> select name, open_mode, restricted from v$pdbs;

NAME            OPEN_MODE  RESTRICTED
--------------- ---------- -----------
PDB$SEED        READ ONLY  NO
PDB1            READ WRITE NO
PDB2            READ WRITE NO
PDB3            READ WRITE YES
```

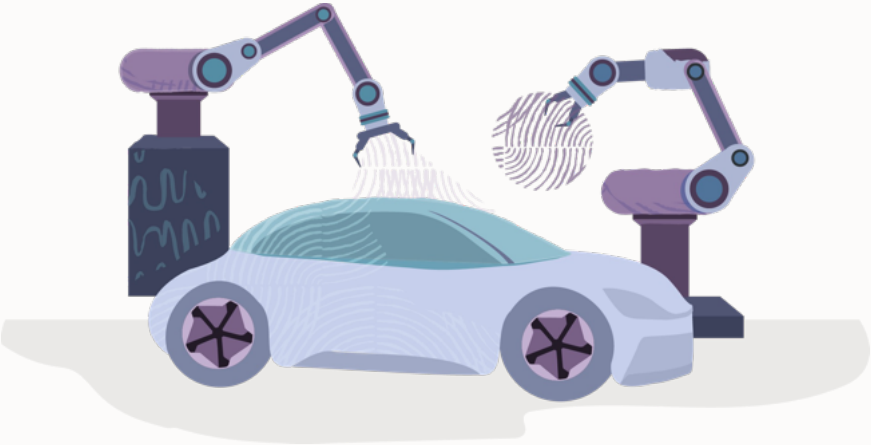
```
--Use with caution. Patching issue must be resolved!

alter system set "_pdb_datapatch_violation_restricted"=FALSE
```

```
SQL> shutdown immediate
SQL> startup

SQL> select name, open_mode, restricted from v$pdbs;

NAME            OPEN_MODE  RESTRICTED
--------------- ---------- -----------
PDB$SEED        READ ONLY  NO
PDB1            READ WRITE NO
PDB2            READ WRITE NO
PDB3            READ WRITE NO
```

You must resolve the patching issue

- Use underscore parameter with caution

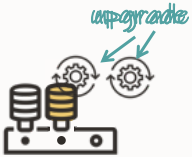
Upgrading



Multitenant Upgrade Approaches

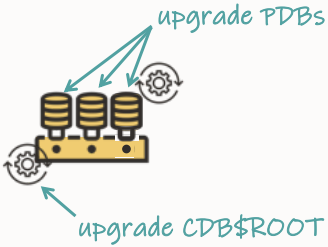
PDB Upgrade

- Happens either for unplug/plug or for non-CDB plugins followed by upgrade
- Can be done with multiple PDBs together



CDB Upgrade

- Upgrade the entire CDB with all PDBs
- CDB$ROOT will be always upgraded first



Multitenant Upgrade Approaches

PDB Upgrade

- Pros
  - Flexibility
  - Fast
  - Control
- Cons
  - You need at least another CDB
  - Resource constraints
  - PDBs need to be cloned or moved
  - Flashback Database can't be used

CDB Upgrade

- Pros
  - Less work, more automation
  - Upgrade many-as-one
  - Happens in-place
  - No extra resources required
  - Flashback Database protection
- Cons
  - Less control
  - Common SLAs needed

Multitenant Upgrade Approaches

PDB Upgrade

Non-CDB to PDB upgrade

```
upg1.source_home=/u01/app/oracle/prod/19
upg1.target_home=/u01/app/oracle/prod/23
upg1.sid=NONCDB19
upg1.target_cdb=CDB23
```

Unplug-plug upgrade

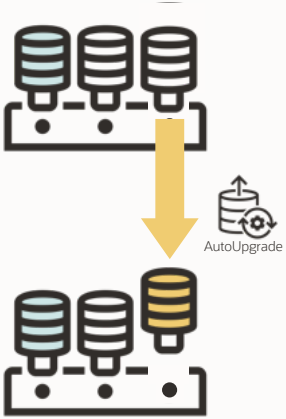
```
upg1.source_home=/u01/app/oracle/prod/19
upg1.target_home=/u01/app/oracle/prod/23
upg1.sid=CDB19
upg1.target_cdb=CDB23
upg1.pdbs=PDB2,PDB3
```

CDB Upgrade

Entire-CDB upgrade

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/23
upg1.sid=CDB
```

PDB Unplug – Plug - Upgrade



PDB gets upgraded with parallel processes

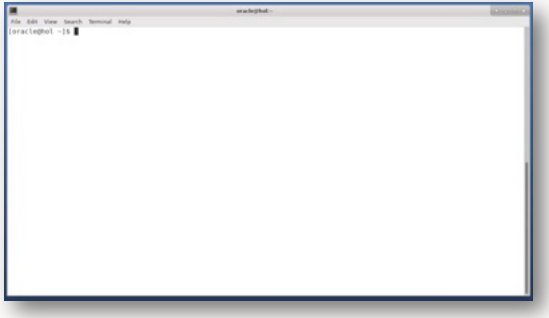
- Minimum 1
- Maximum 8
- Default 2
- You can override the default with:

```
prefix.catctl_options=-N 4
```

PDB Unplug – Plug - Upgrade

Demo

```
upg1.sid=CDB12102
upg1.target_cdb=CDB19
upg1.pdbs=pdb1
upg1.source_home=/u01/app/oracle/product/12102
upg1.target_home=/u01/app/oracle/product/19
```



Watch on YouTube

PDB Unplug – Plug - Upgrade

Upgrade several PDBs

```
upg1.pdbs=pdb1,pdb2,pdb3
```

Rename a PDB

```
upg1.pdbs=pdb1
upg1.target_pdb_name.pdb1=sales
```

Copy data files on plug-in

```
upg1.pdbs=pdb1
upg1.target_pdb_copy_option.pdb1=file_name_convert=('pdb1','sales')
```

Container Database Upgrade

CDB$ROOT gets upgraded always at first with multiple processes

- Minimum 1
- Maximum 8
- Default 4
- You can override the default with:

```
prefix.catctl_options=-n 8
```

Container Database Upgrade

Workers assigned per PDB

- Minimum 1
- Maximum 8
- Default 2
- You can override the default with:

```
prefix.catctl_options=-N 2
```

Container Database Upgrade

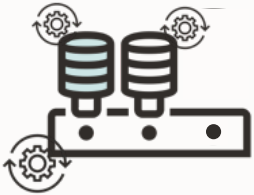
Parallelism calculation

$$\frac{\text{Total number of processors } (n)}{\text{Workers per PDB } (N)} = \text{PDBs upgraded simultaneously}$$

- Default: $\frac{\text{CPU\_COUNT}}{2} = \text{PDBs upgraded simultaneously}$
- You can override the defaults with:

```
prefix.catctl_options=-n 16 -N 4
```

Single Tenant Container Database



Single tenant upgrades take longer

- CDB$ROOT gets upgraded at first
- PDB$SEED and PDB get upgraded in parallel

Container Database Upgrade

Scale by upgrading
more PDBs simultaneously

During upgrade, CPU is a vital resource

Upgrade benchmark

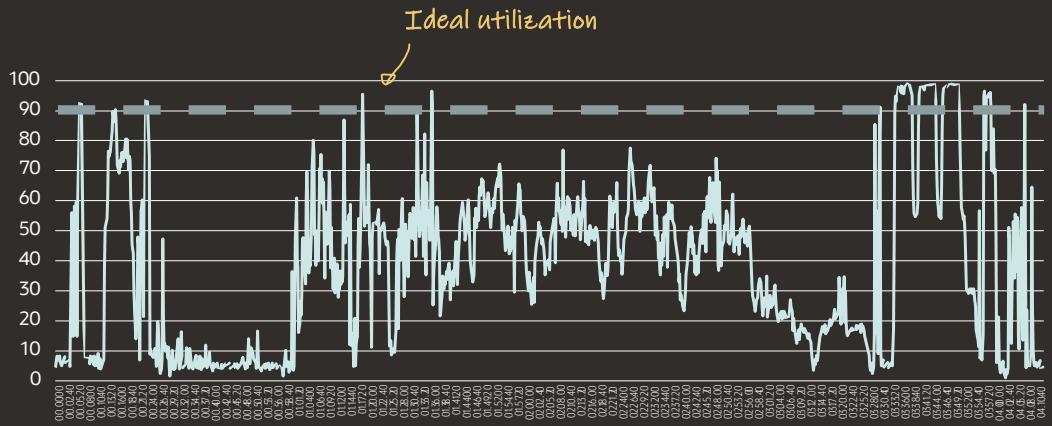
Oracle Database 12.1.0.2 to Oracle Database 19c

16 OPCUs
768 GB memory

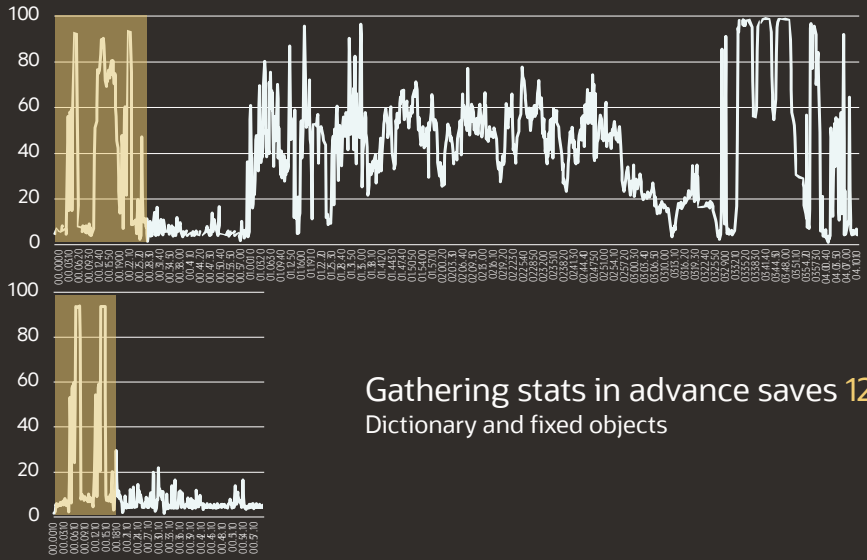
CDB with 52 PDBs
Many database components (17 in total)

CPU_COUNT	32
SGA_TARGET	80G
PGA_AGGREGATE_TARGET	20G

Upgrade Benchmark



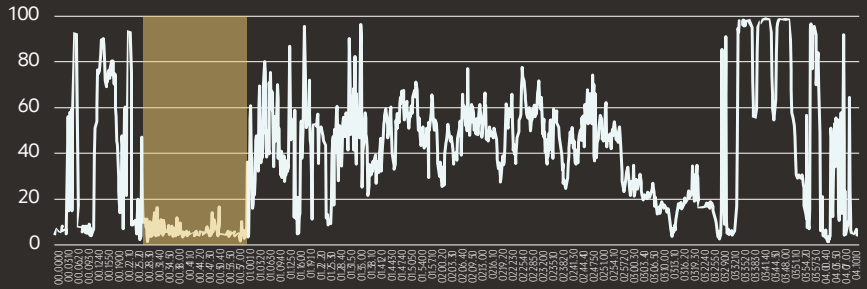
Upgrade Benchmark



Gathering stats in advance saves 12 minutes

Dictionary and fixed objects

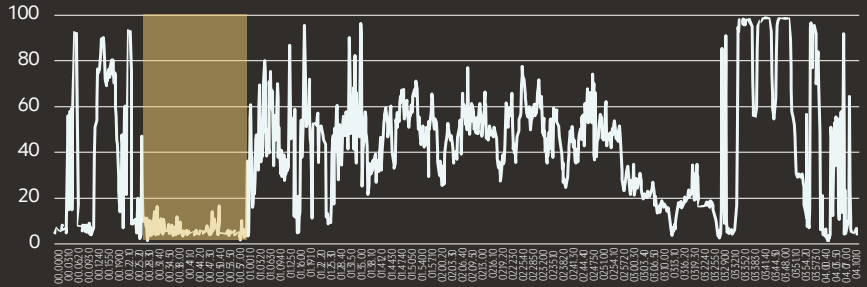
Upgrade Benchmark

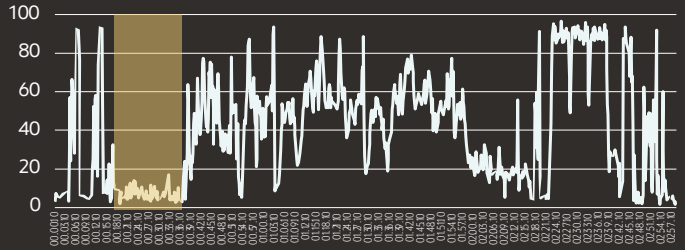


Upgrade CDB$ROOT

- AutoUpgrade automatically assigns 8 parallel processes to CDB$ROOT upgrade
- Speed up the upgrade? Consider removal of unused components

Upgrade Benchmark

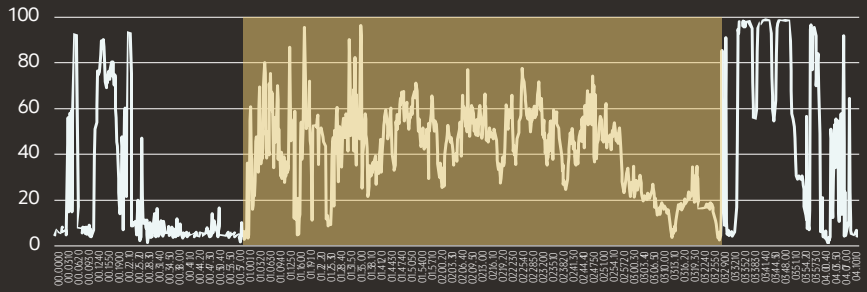




Upgrade CDB$ROOT

- Removing all components
- Result:
  13 minutes faster

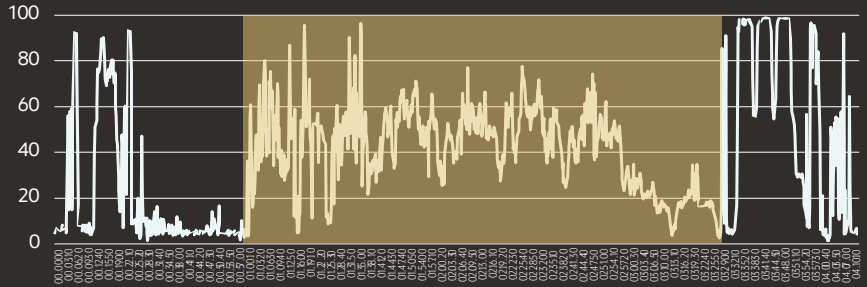
Upgrade Benchmark

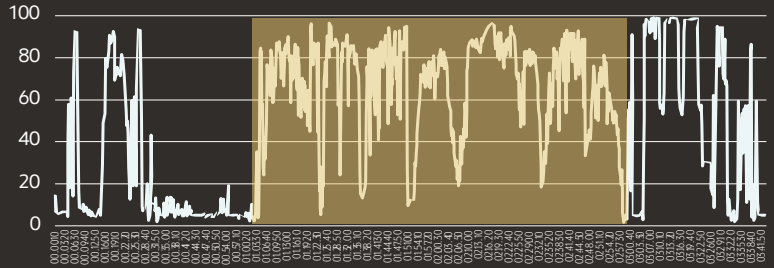


Upgrade PDB$SEED and 52 PDBs

- AutoUpgrade assigns 16 PDBs to be upgraded in parallel
  - `CPU_COUNT` = 32 /2 workers per PDB
- Speed up the upgrade? Consider increasing number of parallel processes

Upgrade Benchmark

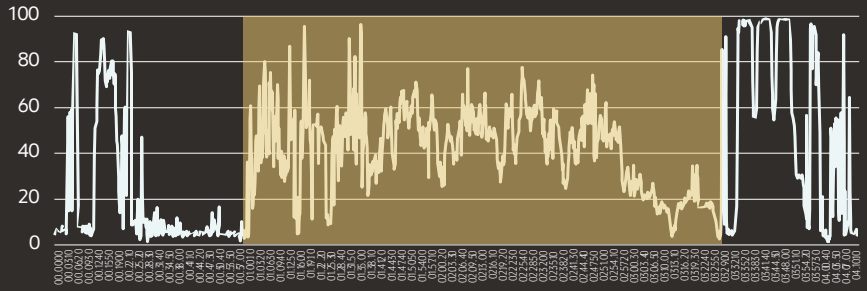




Upgrade PDBs

- 54 parallel processes
  `upg1.catctl_options=-n 54`
- 26 minutes faster

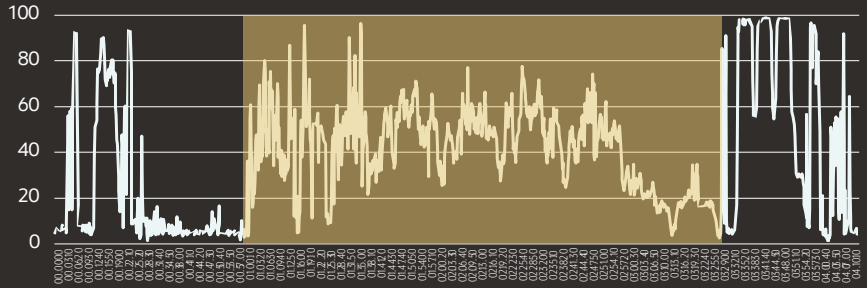
Upgrade Benchmark

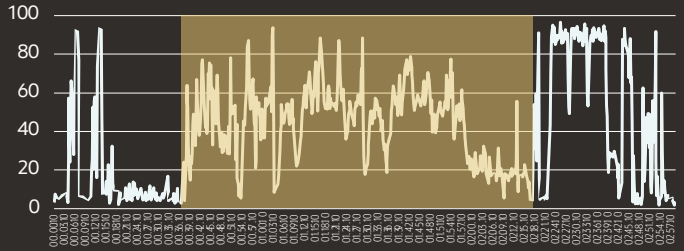


Upgrading CDB$ROOT, PDB$SEED and 52 PDBs

- Speed up the upgrade? Consider removal of unused components

Upgrade Benchmark

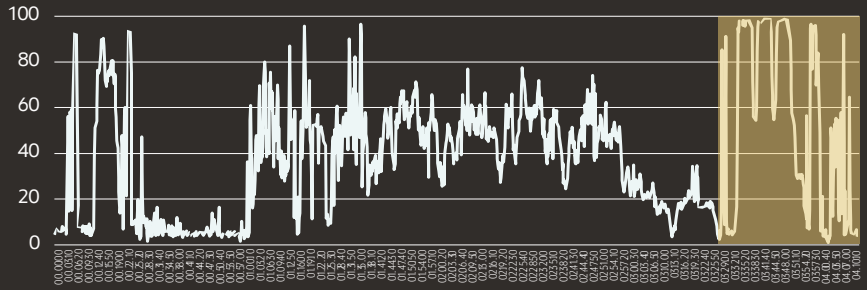




Upgrading all containers

- Removing all <u>components</u>
- Increased parallel processes
- Result:
  48 minutes faster

Upgrade Benchmark



Recompilation and post-upgrade fixups

- Recompilation is already optimized very efficiently
- Potentially, skip or postpone the time zone upgrade

Upgrade Benchmark Results

Gather dictionary and fixed objects stats before upgrade

5% improvement

Remove unused components from root and all PDBs

19% improvement

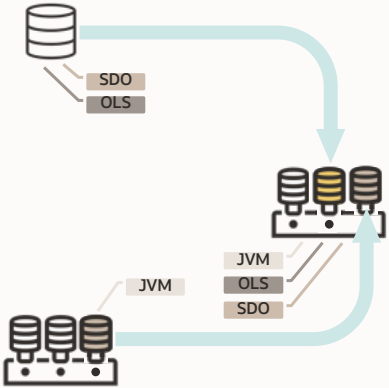
Increase number of PDBs upgraded in parallel

10% improvement

Implement all recommendations

32% improvement

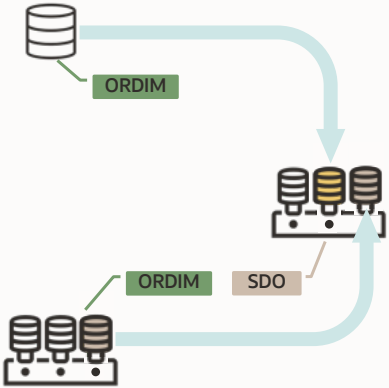
Component Considerations



CDB$ROOT must have the same or a superset of components installed

- Otherwise, a plug in violation will be signaled
- The PDB will not open unrestricted

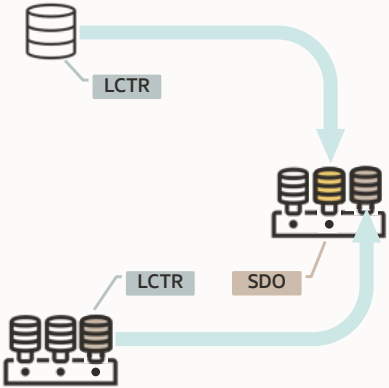
Special Case: Multimedia



When Multimedia (ORDIM) is installed in source, then SDO (Spatial Data Option) must be installed in CDB$ROOT

- `select username from dba_users where username='MDSYS';`
- Otherwise, a plug in violation will be signaled
- The PDB will not open unrestricted

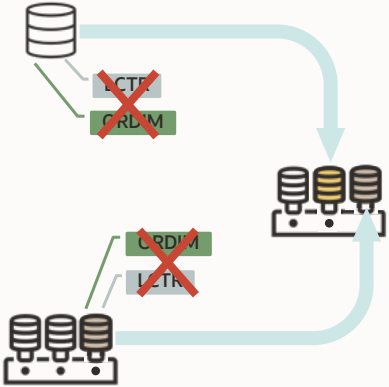
Special Case: Locator



When the Locator is in use in the source, then SDO (Spatial Data Option) must be installed in CDB$ROOT

- Otherwise, a plug in violation will be signaled
- The PDB will not open unrestricted

Alternative Way: Cleanup



If you want to avoid installing SDO into CDB$ROOT, remove ORDIM and Locator beforehand

- `select * from dba_registry`
  `where comp_id in ('ORDIM','LCTR');`
- Removal options?
  - Blog post: ORDIM cleanup
  - Blog post: LCTR treatment

```
--There are three recompilation scripts available:
--utlrp.sql    => classic one
--utlprp.sql   => parallel recompile – needs '--pN' option
--utlprpom.sql => only Oracle maintained – needs '--pN' option

cd $ORACLE_HOME/rdbms/admin
perl catcon.pl \
   -b recomp -l /tmp \
   -n 10 \
   utlprpom.sql '--p16'
```

Replay Upgrade

Replay Upgrade is a performance feature used to upgrade a single PDB

- Available since Oracle Database 21c

```
--The database automatically starts an upgrade
--when you plug in a lower-release PDB

SQL> alter pluggable database pdb1 open;

Pluggable database altered.

Elapsed: 00:06:01.95
```

```
SQL> select property_name, property_value
     from   database_properties
     where  property_name like '%OPEN%';

PROPERTY_NAME          PROPERTY_VALUE
---------------------- --------------------
CONVERT_NONCDB_ON_OPEN true
UPGRADE_PDB_ON_OPEN    true
```

```
SQL> select property_name, property_value
     from   database_properties
     where  property_name like '%OPEN%';

PROPERTY_NAME          PROPERTY_VALUE
---------------------- --------------------
CONVERT_NONCDB_ON_OPEN true
UPGRADE_PDB_ON_OPEN    true
```

Traditional Upgrade

Phase 1
Phase 2
Phase 3
Phase 4
Phase 5
Phase 6
Phase 7
Phase 8
...
Phase *nnn*

Traditional Upgrade

Phase 1
Phase 2
Phase 3
Phase 4
Phase 5

```
├── @a2300932.sql
├── @a2300933.sql
├── @a23009xx.sql
└── @c2300000.sql
```

Phase 6
Phase 7
Phase 8
...
Phase *nnn*

Traditional Upgrade

@a2300932.sql

```
VARIABLE initfile VARCHAR2(32)
COLUMN :initfile NEW_VALUE init_file NOPRINT;

Rem ====================================================================
Rem SQLJTYPE
Rem ====================================================================

BEGIN
  IF sys.dbms_registry.is_loaded('JAVAVM',sys.dbms_registry.release_version) = 1 THEN
     :initfile := 'initsjty.sql';
  ELSE
     :initfile := 'nothing.sql';
  END IF;
END;
/
SELECT :initfile FROM DUAL;
@@&init_file
```

Traditional Upgrade

```
@@&init_file
```

```
.
.
[more PL/SQL code]
.
.
CREATE TABLE SYS.T1 ...
CREATE INDEX SYS.T1I1 ...
.
.
[more PL/SQL code]
.
.
```

Comparison

Traditional

Phase 1
Phase 2
Phase 3
Phase 4
Phase 5
Phase 6
Phase 7
Phase 8
...
Phase *nnn*

Replay

```
DROP INDEX SYSTEM.IDX$FLOW ...
CREATE OR REPLACE ...
ALTER TYPE ...
CREATE FUNCTION ...
CREATE TABLE SYS.T1 ...
CREATE INDEX SYS.T1I1 ...
DROP INDEX MDSYS.IDX$IK ...
DROP TABLE MDSYS.TBL$TT ...
CREATE OR REPLACE ...
ALTER TYPE ...
GRANT SELECT ON ...
CREATE VIEW ...
```

```
select sqlstmt from pdb_sync$;

sqlstmt
---------------------------------------
ALTER SESSION SET "_oracle_script_counter"=7
alter pluggable database application app$cdb$pdbonly$ncdbtopdb begin install '1.0.upgmode'
alter session set "_enable_view_pdb"=false
alter session set NLS_LENGTH_SEMANTICS=BYTE
INSERT INTO sys.utl_recomp_skip_list select obj# from obj$ where BITAND(flags, 4194304)=0 ...
create or replace view sys.cdb$common_root_objects sharing=object as
select u.name owner, o.name object_name, o.type# object_type, o.namespace nsp,
       o.subname object_subname, o.signature object_sig,
       decode(bitand(o.flags, (65536+131072+4294967296)),
       4294967296+65536, 'EDL', 131072, 'DL', 'MDL') sharing
  from sys.obj$ o, sys.user$ u
 where o.owner#=u.user# and bitand(o.flags, (65536+131072+4294967296)) <> 0
   and bitand(o.flags,0)=0
```

Traditional vs Replay

Traditional

- Triggered by AutoUpgrade
- Runs `catalog.sql` / `catproc.sql`
- Many `CREATE OR REPLACE` statements for objects that didn't change
- Customizable
- Used by AutoUpgrade

Replay

- Triggered by `OPEN` command
- Runs the captured statements
- Only statements that actually do some change
- Automated

No-Op Operations

During replay, *no-op* statements will be skipped

- 12.2.0.1 ➔ 21c upgrade
  - 74531 total statements - 50% are no-ops
- 18c ➔ 19c upgrade
  - 68374 total statements - 73% are no-ops

Replay Upgrade

After upgrade

- Call Datapatch
  ```
  $ORACLE_HOME/OPatch/datapatch -pdbs PDB1 -verbose
  ```
- Call AutoUpgrade
  ```
  java –jar autoupgrade.jar –config PDB1.conf -fixups
  ```

Replay Upgrade – on failure?

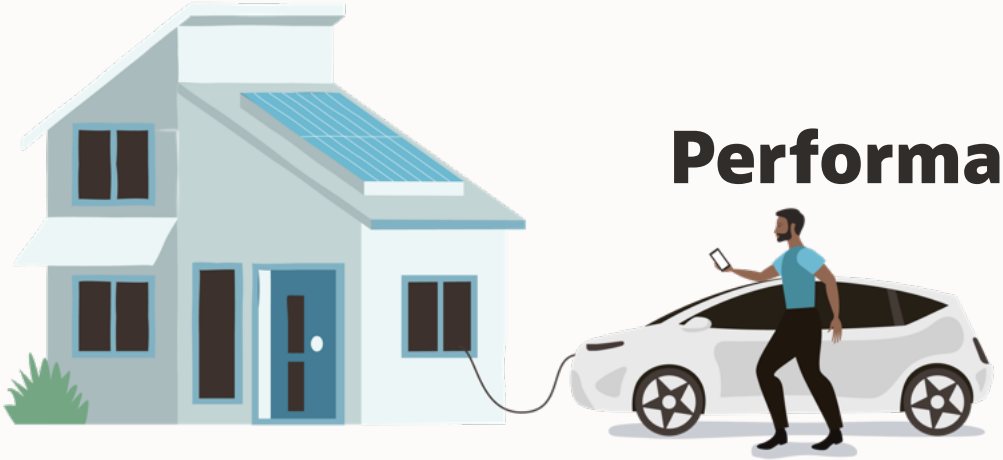
If Replay Upgrade fails

- Check for errors:
  - `SELECT * FROM DBA_APP_ERRORS`
  - Check alert log
  - Trace files
- Revert to traditional upgrade

```
--To disable replay upgrade
ALTER DATABASE UPGRADE SYNC OFF;
--Or
ALTER DATABASE PROPERTY SET UPGRADE_PDB_ON_OPEN='false';

--To disable convert on open
ALTER DATABASE PROPERTY SET CONVERT_NONCDB_ON_OPEN='false';
```

Replay Upgrade

Documentation



Performance

Proactive Fixups

Faster Upgrades with many PDBs

Proactive Fixups result in faster upgrades

- For CDBs with many PDBs

Proactive Fixups?

Performance feature

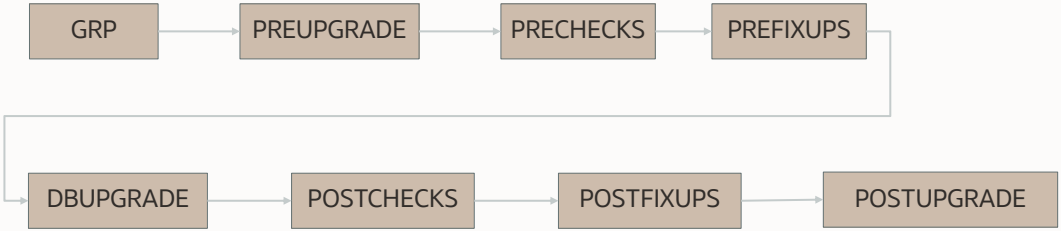
Start PDB post-upgrade tasks as soon as a PDB has been upgraded

- Independently of other PDBs

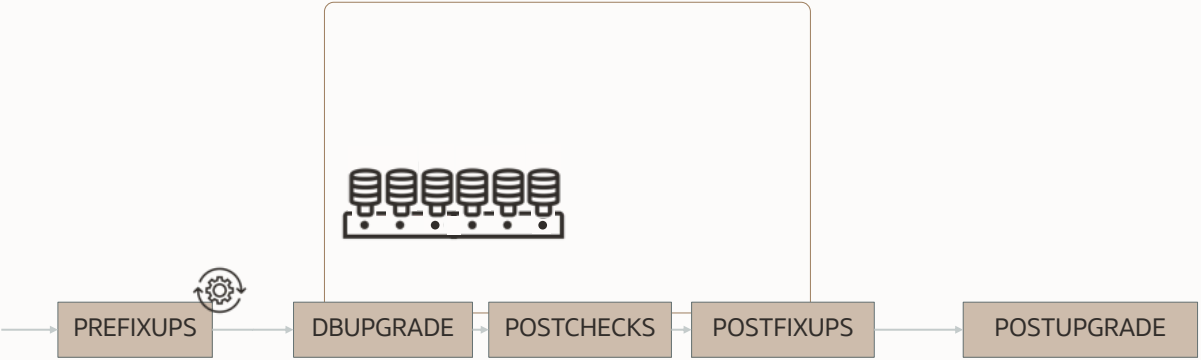
Isolates errors in PDBs

Valid for CDB upgrades only

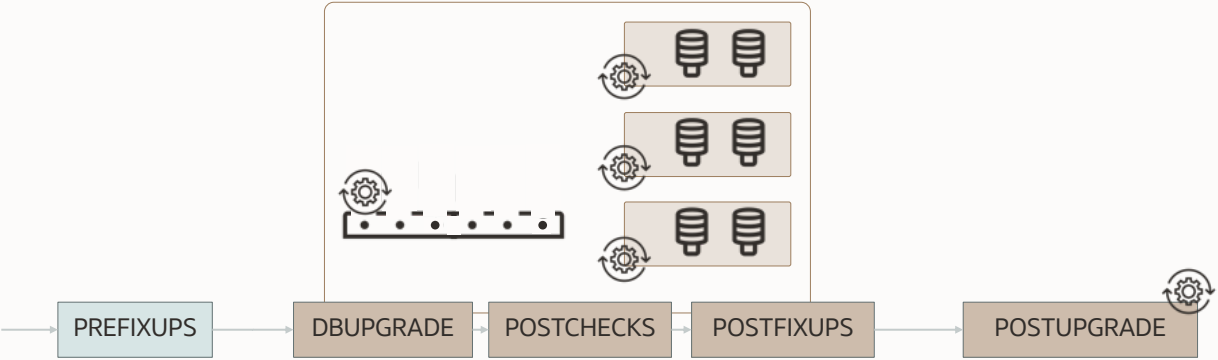
Classic Upgrade Flow



Proactive Fixups Flow



Proactive Fixups Flow



Proactive Fixups Command Output

```
Stage-Progress Per Container
        +--------+----------+--------+
        |Database|     Stage|Progress|
        +--------+----------+--------+
        |PDB$SEED| DBUPGRADE|    91 %|
        |   PDB01|POSTFIXUPS|     0 %|
        |   PDB02| DBUPGRADE|    20 %|
        |   PDB03|POSTFIXUPS|    25 %|
        |   PDB04|POSTFIXUPS|    75 %|
        |   PDB05|POSTFIXUPS|    10 %|
        |   PDB06| DBUPGRADE|     6 %|
        |   PDB07| DBUPGRADE|    91 %|
        |   PDB08| DBUPGRADE|    91 %|
        |   PDB09| DBUPGRADE|    91 %|
        +--------+----------+--------+
```

Performance Gain

4 PDBs + ROOT | 4 Cores

Default			Proactive Fixups		
INFO	PREUPGRADE	<1 min	INFO	PREUPGRADE	<1 min
INFO	PRECHECKS	1 min	INFO	PRECHECKS	1 min
INFO	PREFIXUPS	8 min	INFO	PREFIXUPS	7 min
INFO	DRAIN	<1 min	INFO	DRAIN	<1 min
INFO	DBUPGRADE	143 min	INFO	DBUPGRADE	130 min
INFO	POSTCHECKS	2 min	INFO	POSTCHECKS	<1 min
INFO	POSTFIXUPS	34 min	INFO	POSTFIXUPS	<1 min
INFO	POSTUPGRADE	1 min	INFO	POSTUPGRADE	1 min
	TOTAL	179 min		TOTAL	130 min

Performance Gain

16 PDBs + ROOT | 8 Cores | Defaults

Default			Proactive Fixups		
INFO	PREUPGRADE	<1 min	INFO	PREUPGRADE	<1 min
INFO	PRECHECKS	<1 min	INFO	PRECHECKS	<1 min
INFO	PREFIXUPS	<1 min	INFO	PREFIXUPS	14 min
INFO	DRAIN	2 min	INFO	DRAIN	2 min
INFO	DBUPGRADE	210 min	INFO	DBUPGRADE	195 min
INFO	POSTCHECKS	3 min	INFO	POSTCHECKS	<1 min
INFO	POSTFIXUPS	46 min	INFO	POSTFIXUPS	<1 min
INFO	POSTUPGRADE	<1 min	INFO	POSTUPGRADE	1 min
	TOTAL	259 min		TOTAL	195 min

The more PDBs, the greater the benefit

Proactive Fixups isolate each PDB

- Errors in any given PDB don't effect others

PDB Isolation

DEFAULT

Error in a PDB upgrade:
- Entire job halts
- Job can't complete

PROACTIVE FIXUPS

Error in a PDB upgrade:
- Other upgrades continue
- Job completes

Restore points protect on CDB-level only.
You can only flashback the entire CDB.

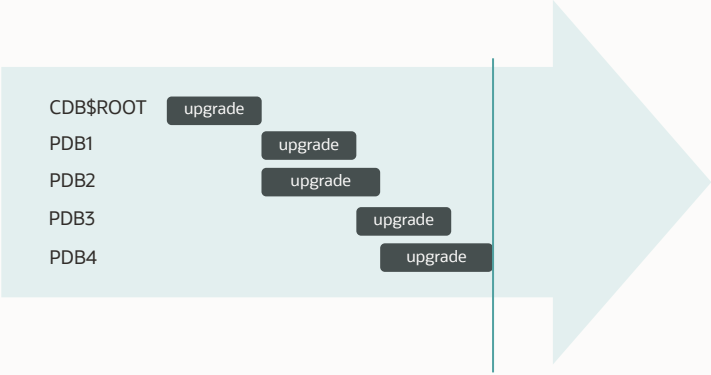
Some PDBs are more important

Control the order of the upgrade

Proactive Fixups Availability

DEFAULT

prefix.make_pdbs_available=false

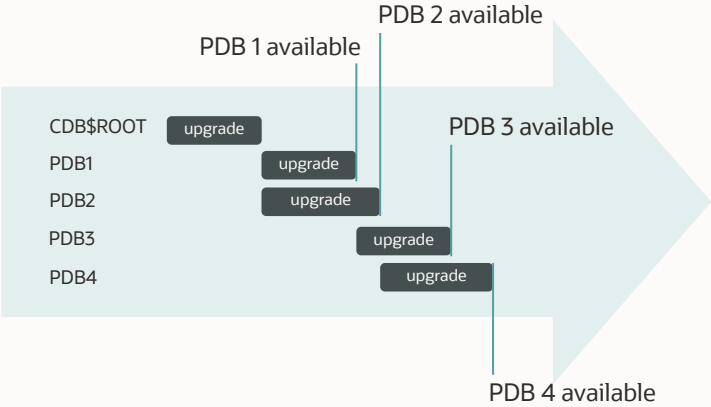


All PDBs become available

Proactive Fixups Availability

IMMEDIATELY AVAILABLE

prefix.make_pdbs_available=true

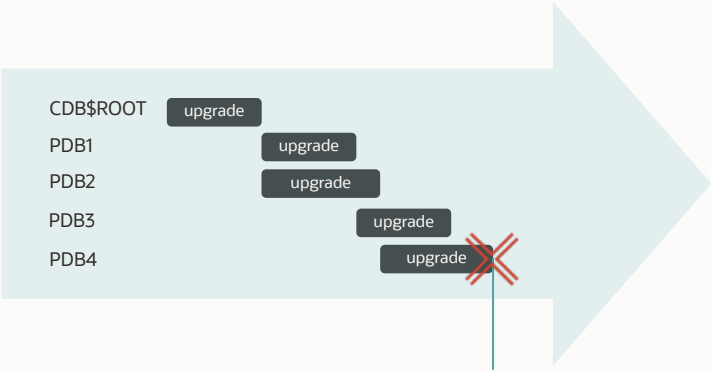


```
alter pluggable database SALESPROD priority 1;

alter pluggable database SALESDEV priority 2;

alter pluggable database SALESUAT priority 2;

alter pluggable database SALESTEST priority 3;
```

PDB Availability

IMMEDIATELY AVAILABLE

prefix.make_pdbs_available=true



PDB 4 crashes ...
Flashback entire CDB?

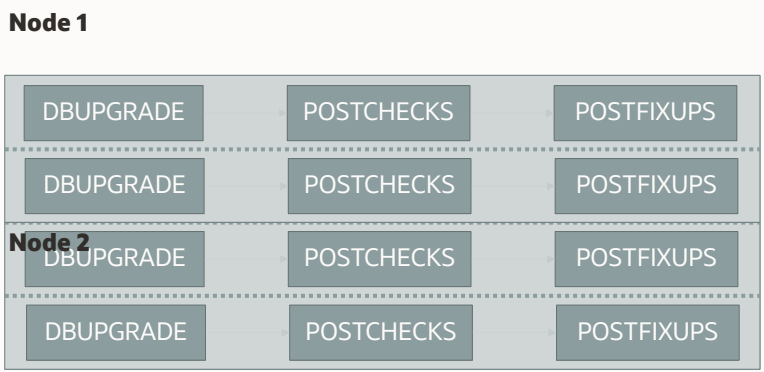
Distributed Upgrade

Leverage multiple cluster nodes

Distributed upgrade uses all nodes in a cluster resulting in faster upgrades of CDBs

- Applies to RAC only
- Requires Proactive Fixups

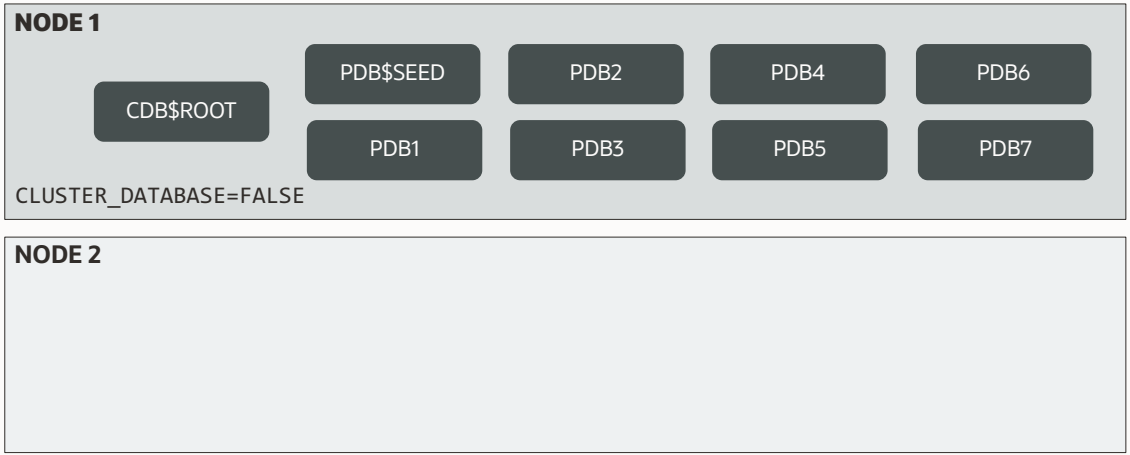
Distributed Upgrade Concept



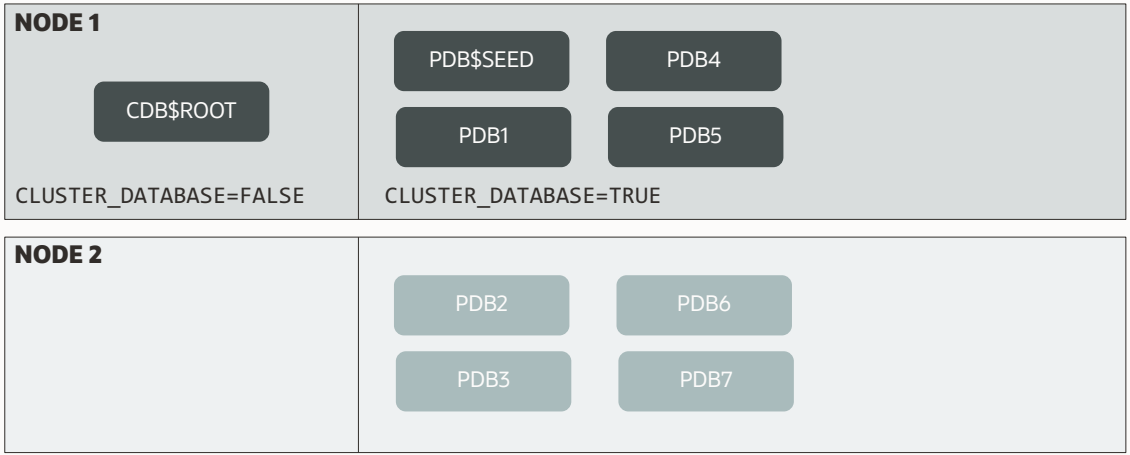
How does Distributed Upgrade work?

- Performance feature
- Valid for CDB upgrades on RAC only
- First, CDB$ROOT upgrades on local node
  `CLUSTER_DATABASE=FALSE`
- Then, leverage resources on all nodes to upgrade PDBs
  `CLUSTER_DATABASE=TRUE`

Regular Upgrade



Distributed Upgrade



Distributed Upgrade Console Output

```
Stage-Progress Per Container
        +--------+----------+--------+----+
        |Database|     Stage|Progress|Node|
        +--------+----------+--------+----+
        |PDB$SEED| DBUPGRADE|    91 %| au1|
        |   PDB01|POSTFIXUPS|     0 %| au1|
        |   PDB03|POSTFIXUPS|     0 %| au1|
        |   PDB04|POSTFIXUPS|     0 %| au1|
        |   PDB05|POSTFIXUPS|     0 %| au1|
        |   PDB02| DBUPGRADE|    91 %| au2|
        |   PDB06| DBUPGRADE|    91 %| au2|
        |   PDB07| DBUPGRADE|    91 %| au2|
        |   PDB08| DBUPGRADE|    91 %| au2|
        |   PDB09| DBUPGRADE|    91 %| au2|
        +--------+----------+--------+----+
```

Distributed Upgrade

Enable distributed upgrade:

```
$ cat RACDB.cfg

upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/23
upg1.sid=RACDB
upg1.tune_setting=distributed_upgrade=true

$ java -jar autoupgrade.jar -config RACDB.cfg -mode deploy
```

41% faster

In benchmark, time saved by using
distributed upgrade

2 node RAC database
4 CPUs each
CDB with 8 PDBs

By default, AutoUpgrade uses two nodes

```
--Control how many nodes will be used

upg1.tune_setting=distributed_upgrade=true,active_nodes_limit=n
```

Time Zone Upgrade

Near-Zero Downtime?

All available time zone files get shipped since Oracle Database 19.18.0

- Does not apply to Oracle Database 21c
- Files are in `$ORACLE_HOME/oracore/zoneinfo`

Time Zone Check

Check current version:

```
alter session set container='CDB$ROOT';
alter system set "_exclude_seed_cdb_view"=false scope=both;

select value$, con_id from containers(SYS.PROPS$) where
NAME='DST_PRIMARY_TT_VERSION' order by con_id;

   VALUE$     CON_ID
_________ _________
32                  1
32                  2
32                  4
```

AutoUpgrade

Config file parameter: *prefix*.timezone_upg=YES

- Default for upgrades: YES
- Default for patching: NO
- In case *DST-source > DST-target*, AutoUpgrade copies necessary files

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/23
upg1.sid=CDB
upg1.timezone_upg=NO
```

Manual Time Zone Upgrade

Make sure all PDBs are open unrestricted

Make sure all PDBs restart automatically

- This is very important due to the restart happening

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -b tzcheck
-d $ORACLE_HOME/rdbms/admin -n 1 -l /tmp utltz_upg_check.sql
```

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -b tzapply
-d $ORACLE_HOME/rdbms/admin -n 1 -l /tmp utltz_upg_apply.sql
```

How to patch all PDBs with the a new time zone file?

Posted on December 18, 2018 by Mike.Dietrich Patch Recommendation Single-/Multitenant Time Zone - DST

Yesterday I wrote about how to adjust the time zone setting in the PDB$SEED as by default the time zone scripts won't touch the PDB$SEED when you execute them. And in addition, MOS Note:1509653.1 tells you, that the PDB$SEED can't be adjusted. But this leads to a weird mix of time zone settings across a Multitenant deployment. Which I'd guess is not desired. Following a tweet reply by Marco Mischke I realized: I explained how to patch the PDB$SEED – but I didn't explain **how to patch all PDBs with the a new time zone file**?

Photo by Lauren Mizzoni on Unsplash

Blog: How to patch all your PDBs with a new time zone patch?

Near-zero downtime time zone upgrade

- Introduced in Oracle Database 21c
- Provided check/apply scripts work only from 23.4 onward

Near-Zero Downtime Time Zone Upgrade

Parameter:

`TIMEZONE_VERSION_UPGRADE_ONLINE=TRUE;`

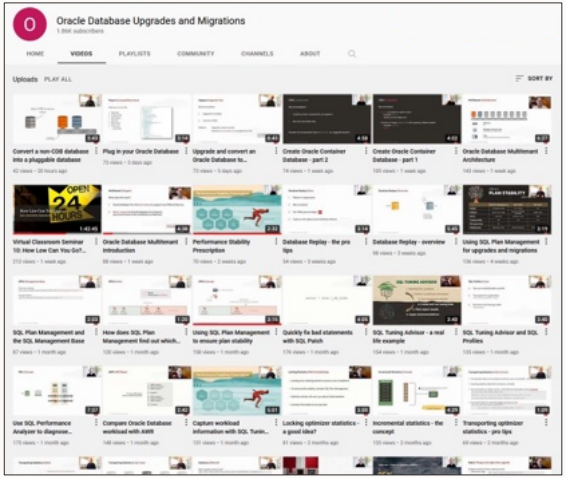
- No `STARTUP UPGRADE` anymore
- Complete database restart is still required
  - You decide the point of restart
  - Before the restart happens, database needs to do conversions

- Be aware:
  - Tables will be rebuilt with ONLINE MOVE
  - No further capacity checks happen

Blog post for more details

Wrapping Up

YouTube | @UpgradeNow



Link

- 300+ videos
- New videos every week
- No marketing
- No buzzwords
- All tech

SCAN ME

Thank You