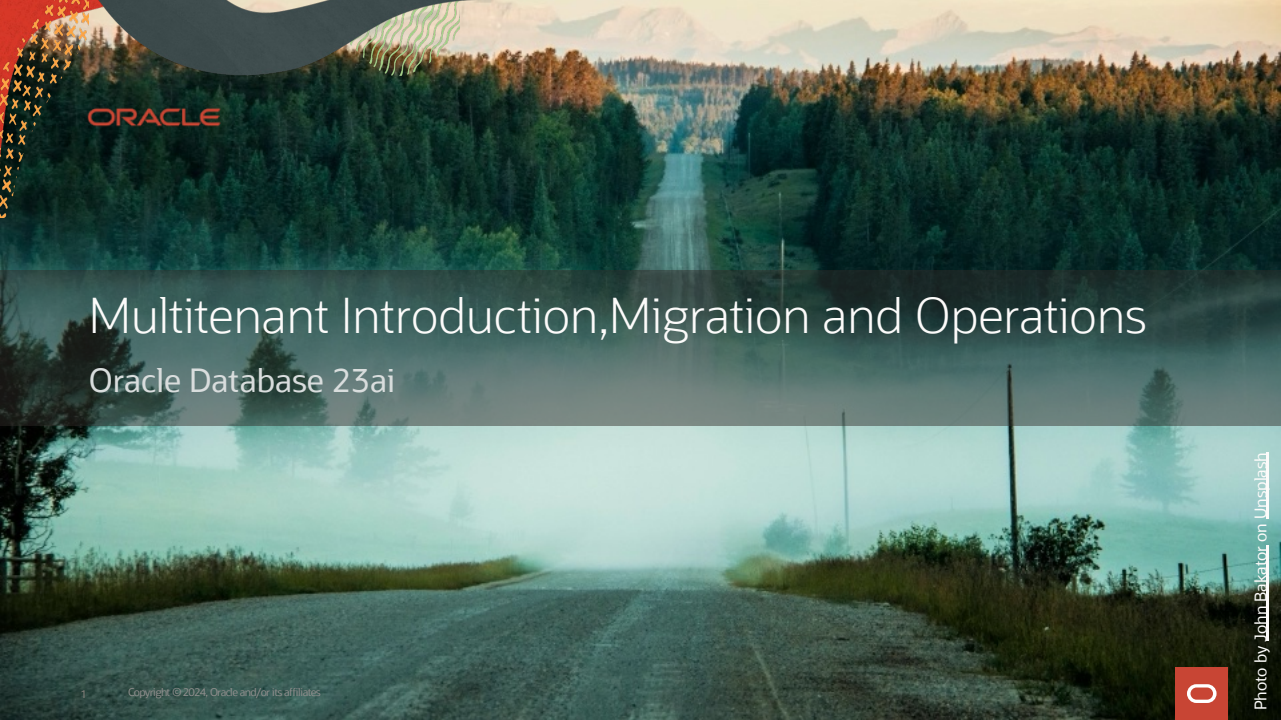


Multitenant Introduction,Migration and Operations

Oracle Database 23ai

Photo by John Bakator 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

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	

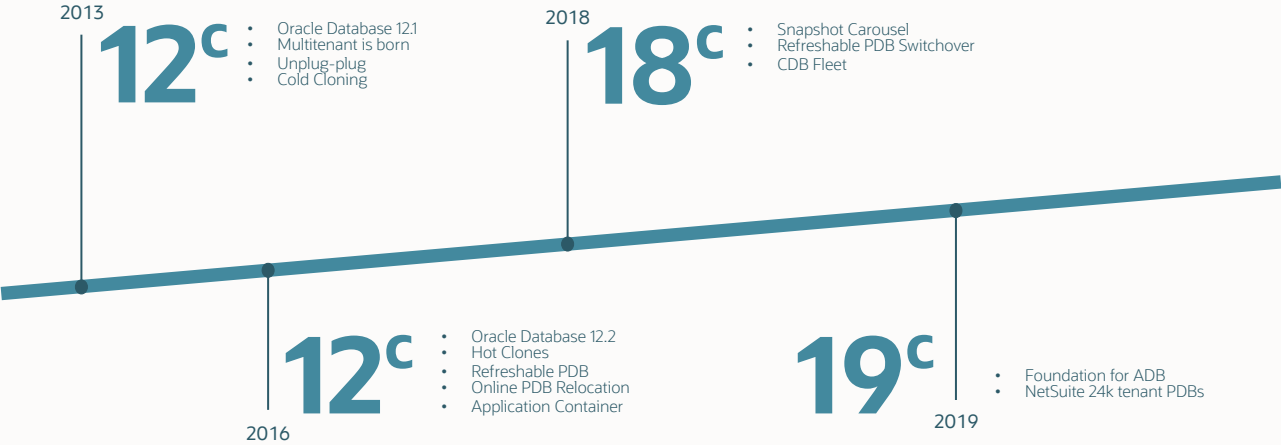
Recorded Web Seminars

https://MikeDietrichDE.com/videos

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

Architecture

Multitenant Evolution



Starting with Oracle Database 21c,
installation of non-CDB Oracle Database architecture
is no longer supported

Upgrade Guide, 23ai

Once you upgrade beyond Oracle Database 19c, you must convert to the multitenant architecture

Oracle Database 19c is the last release to support the non-CDB architecture

Next Long Term Support release

Oracle Database 23ai

Upgrade possible only from:

- Oracle Database 19c
- Oracle Database 21c

Multitenant enables an Oracle database to function as a container database

Generally, you don't need to change your application to use a pluggable database

Single vs. Multitenant

Single Tenant

One PDB
No extra license

Multitenant

Multiple PDBs
Extra license if more than 3 PDBs

```
--Use up to 3 user-created PDBs without a license for Multitenant
--Does not include CDB$ROOT and PDB$SEED
--Applies to Oracle Database 19c and newer, including SE2

alter system set max_pdbs=3;
```


Multitenant

1

2

3

You always create a new CDB

- `CREATE DATABASE ... ENABLE PLUGGABLE DATABASE`
- Using DBCA

Or clone an existing CDB

Multitenant

1

2

3

When you create a new CDB, it contains:

- The root container
- The seed container

Multitenant

1

2

3

You can create PDBs:

- From the seed container
- By cloning other PDBs
- By converting a non-CDB

Be cautious making changes to *PDB$SEED*

- Your own customizations do not belong *PDB$SEED*

Containers

CDB$ROOT is always 1

PDB$SEED is always 2

User-created PDBs have ID 3 or above

```
SQL> select con_id, name
     from   v$containers;


    CON_ID        NAME
_________ ___________
         1    CDB$ROOT
         2    PDB$SEED
         3        PDB1
         4        PDB2
```

```
alter session set container=CDB$ROOT;
show con_id

CON_ID
------------------------------
1

alter session set container=PDB1;
select sys_context('USERENV', 'CON_ID') as con_id from dual;

CON_ID
------------------------------
3
```


You can switch between containers,
but a transaction belong to one PDB only

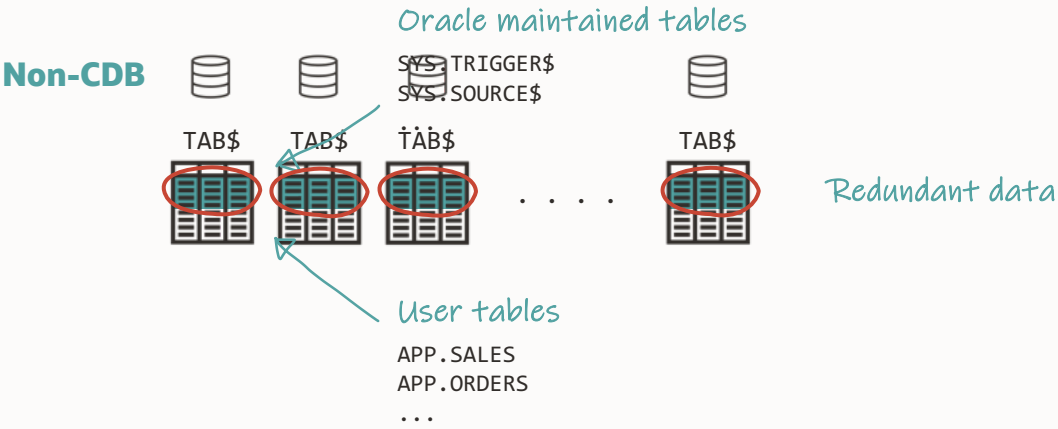
```
alter session set container=PDB1;
insert into table1 values (...);


alter session set container=PDB2;
insert into table2 values (...);
```

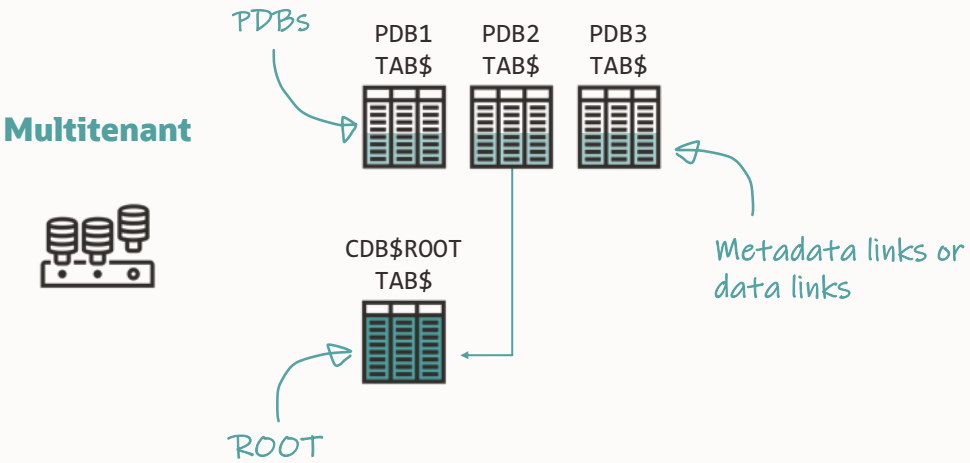
```
ORA-65023: active transaction exists in container PDB1
```


Multitenant implements data dictionary separation

Data Dictionary Architecture



Data Dictionary Architecture



Data Dictionary Architecture

Deduplication

By storing data just once, you can save space in the data dictionary.

Faster

Smaller dictionaries take less time to patch or upgrade.

Easier

With much metadata stored in root, there is less work for a patch or upgrade.

CDB views describe the entire CDB including all PDBs

- Column **CON_ID** indicates the originating container

Data Dictionary Architecture

```
SELECT * FROM USER_OBJECTS
              ALL_OBJECTS
              DBA_OBJECTS
```

```
SELECT * FROM CDB_OBJECTS
```

Applies to any data dictionary view

```
CDB_ALL_TABLES
CDB_ANALYTIC_VIEW_ATTR_CLASS
.
.
.
CDB_XTERNAL_TAB_SUBPARTITIONS
```

Originating container

```
alter session set container=CDB$ROOT;

select con_id, tablespace_name
from   cdb_tablespaces;
```

```
CON_ID TABLESPACE_NAME
------ ---------------
     1 SYSTEM
     1 SYSAUX
     1 UNDOTBS1
     1 TEMP
     1 USERS
     2 SYSTEM
     2 SYSAUX
     2 UNDOTBS1
     2 TEMP
     4 SYSTEM
     4 SYSAUX
     4 UNDOTBS1
     4 TEMP
```

PDB$SEED information, hidden by default

```
alter session set container=PDB1;

select tablespace_name from dba_tablespaces;
```

```
TABLESPACE_NAME
---------------
SYSTEM
SYSAUX
UNDOTBS1
TEMP
```

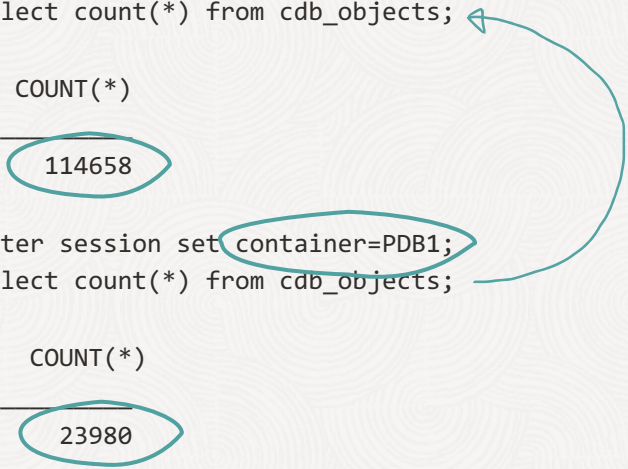
A PDB never sees information from other PDBs

```
alter session set container=CDB$ROOT;
select count(*) from cdb_objects;

  COUNT(*)
----------
    114658

alter session set container=PDB1;
select count(*) from cdb_objects;

  COUNT(*)
----------
     23980
```



You define common objects in root, and they are available in all PDBs

- Most options in application containers

```
alter session set container=CDB$ROOT;

create user C##OPS identified by oracle;
grant create session to C##OPS container=all;

create pluggable database pdb1 ... ;
alter pluggable database pdb1 open;

conn C##OPS/oracle@pdb1
Connected.
```


Common Objects

You can define common:

- Profiles
- Roles
- User
- Audit configuration
- Other objects available in application containers

Useful for:

- Maintenance and monitoring users
- Self-service functionality
- Separation of duties
- Enforcing security

You can change the common prefix with `COMMON_USER_PREFIX`

- We do not recommend changing it
- Use extreme care if you choose to do so

Create same set of common objects in all CDBs to avoid issues during clone/relocate

The database creates common directories

```
alter session set container=PDB1;

select directory_path, origin_con_id
from   cdb_directories
where  directory_name='DATA_PUMP_DIR';

DIRECTORY_PATH                                                                  ORIGIN_CON_ID
/u01/app/oracle/admin/CDB2/dpdump/13D6BC6605416ECEE065000000000001                      1

create or replace directory DATA_PUMP_DIR AS '/tmp';

ERROR at line 1:
ORA-65040: operation not allowed from within a pluggable database

drop directory DATA_PUMP_DIR;

ERROR at line 1:
ORA-65040: operation not allowed from within a pluggable database
```

PDB GUID

Common directory, created in root

Use your own directories
if you want to decide on the directory path

You make most configuration in the root container

```
alter session set container=PDB1;
```

```
alter database backup controlfile to trace;
```

```
ORA-65040: operation not allowed from within a pluggable
database
```


Non-CDB Compatible

- Some `ALTER DATABASE` and `ALTER SYSTEM` commands fail in a PDB

- Enable non-CDB compatibility by setting `NONCDB_COMPATIBLE=TRUE`
  - When you can't change the application
  - When you accept the reduced security

```
SQL> alter system set noncdb_compatible=true;
SQL> shutdown immediate
SQL> startup
```


```
SQL> alter system set noncdb_compatible=true;
SQL> shutdown immediate
SQL> startup

SQL> alter session set container=PDB1;
SQL> alter database backup controlfile to trace;

Database altered.
```


Fine-tune PDBs with instance parameters

- Parameters apply to PDBs as well
- Some parameters are PDB modifiable

```
SQL> select name from v$system_parameter where ispdb_modifiable='TRUE';

NAME
_____________________________________
adg_account_info_tracking
allow_rowid_column_type
approx_for_aggregation
approx_for_count_distinct
approx_for_percentile
.
.
.
xml_handling_of_invalid_chars

246 rows selected.
```


Use ORAdiff to find PDB modifiable parameters

- Free tool
- https://oradiff.oracle.com

A cloned or moved PDB keeps the changed parameters

- Certain exceptions exist

```
--Find specific parameters that has been defined in a specific PDB

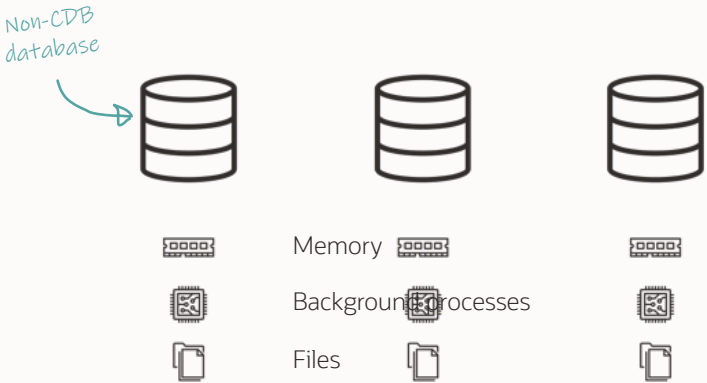
select name, value from v$system_parameter where con_id=<id>;
```


Parameters

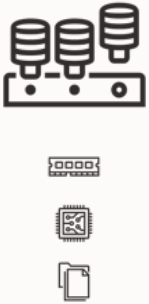
Customer RecommendedInitialization parameters in a Multitenant database - Facts and additional information (Doc ID 2101596.1)

Share resources between PDBs

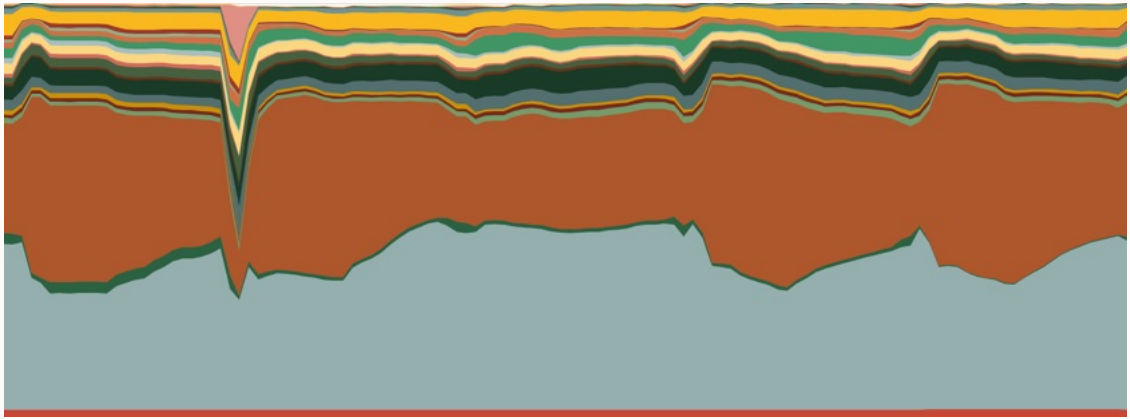
Resource Consolidation



Resource Consolidation

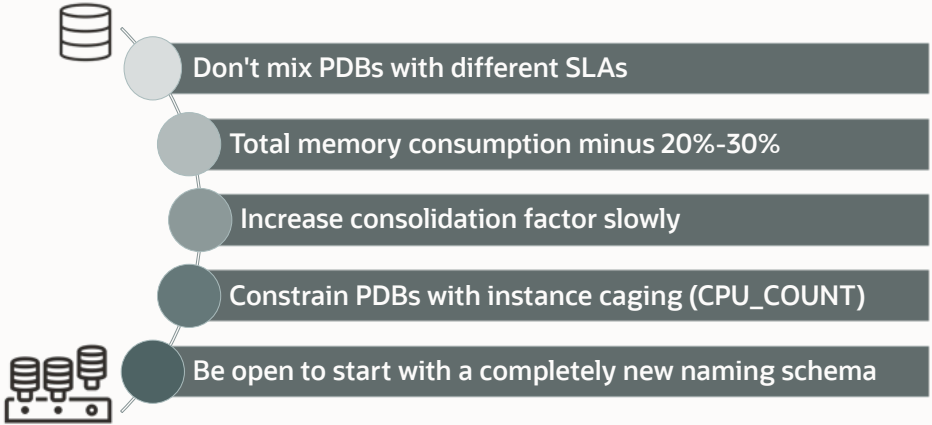


Resource Consolidation



Consolidation Strategies?

There is no "*best*" strategy



Using a Swingbench benchmark,
a single-core machine could host nine non-CDBs
before reaching 75 % CPU utilization

By going multitenant the number of databases reached 123 PDBs

A US Health Care provided managed to

- Reduce the number of database instances by 7x
- Reduce the number of physical servers by 50 %

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

Consolidation

Schema consolidation

Virtual Private Database

PDB consolidation

- Less complexity
- Better isolation
- Operational benefits
- Easier cloning

A global provider of financial services states

The multitenant architecture gives us complete client separation out of the box, without having to maintain a Virtual Private Database setup.

We went away from Virtual Private Database and consolidated our different clients in individual PDBs.

This reduced the complexity of our database implementation and made operations much easier.

The *many-as-one* principle eases maintenance operations

Many-as-one

Patch databases
one by one

Many-as-one

Patch all containers
in one operation

Many-as-one

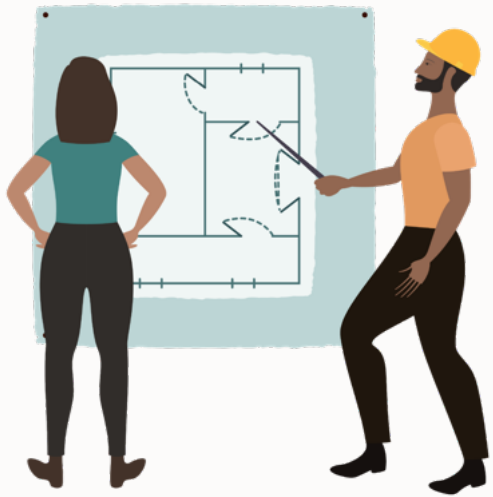
Applies to:

- Upgrading
- Patching
- Configuring and performing backups backups
- Configuring Data Guard
- Configuring RAC
- Monitoring
- … and many other operations

Benefits

The multitenant architecture enables

1. Self-contained PDBs
2. Common and easier operations
3. Resource sharing and consolidation



Create

Create Container Database

1. Character set
2. Components
3. `COMPATIBLE`

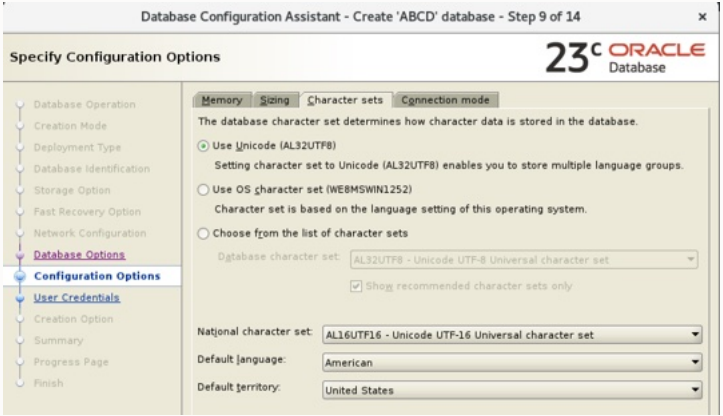
Create Container Database

1 Character set

- Always choose AL32UTF8
- Allows PDBs with any character set

2 Components

3 COMPATIBLE



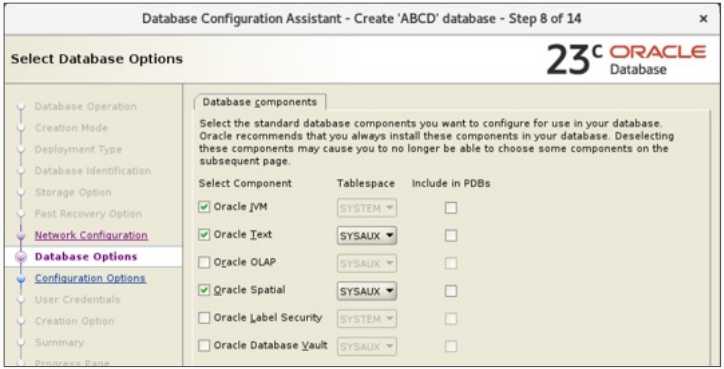
Create Container Database

1 Character set

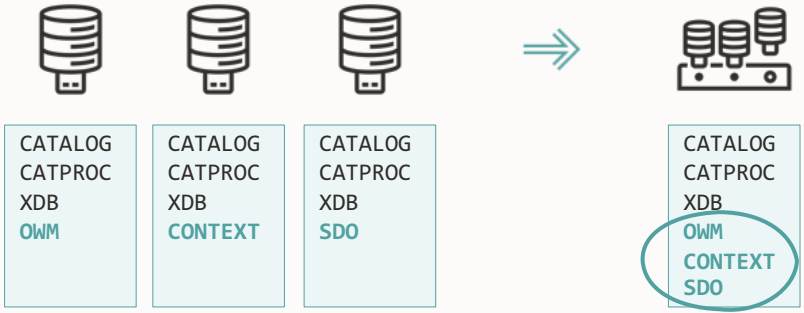
2 Components

- Install as many as you need
- No more than that

3 COMPATIBLE



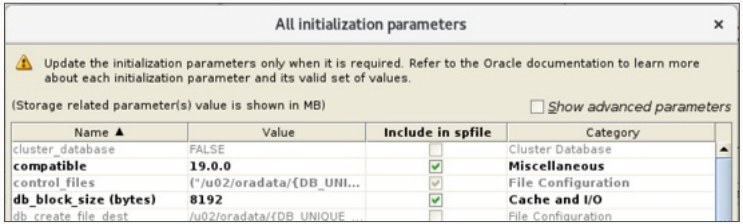
Components



Create Container Database

1 Character set

2 Components



3 COMPATIBLE

- Keep at the default setting, *23.4.0*
- Unless you want the option of downgrade

```
-- Always set compatible to the default of a release
-- Use three digits only

alter system set compatible='23.4.0' scope=spfile;
```

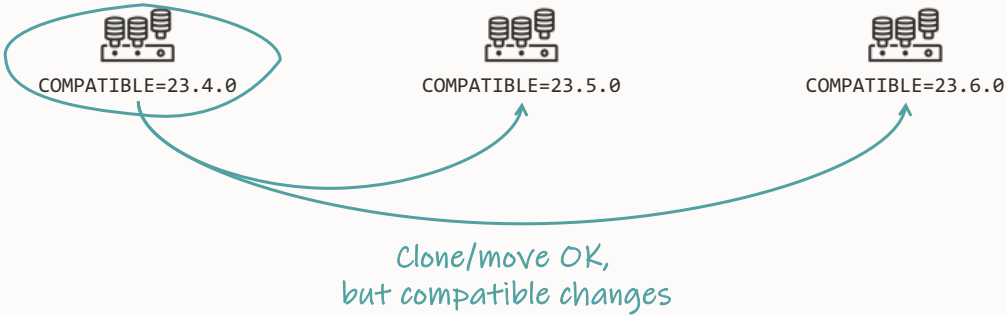
```
-- Should I change compatible when patching?
-- No, this is a bad idea

alter system set compatible='23.5.0' scope=spfile;
```

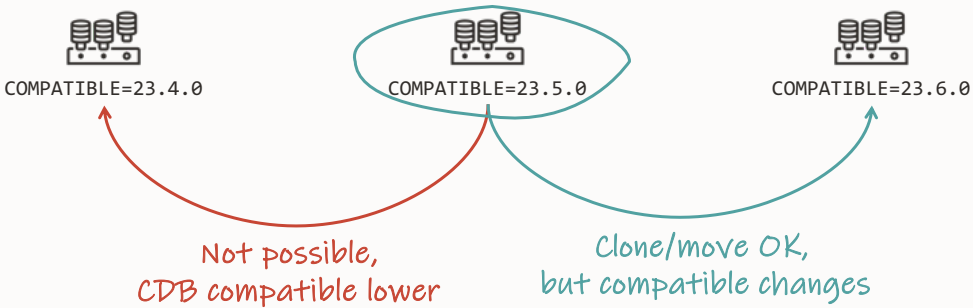

Compatible

- On plug-in, a PDB adopts `COMPATIBLE` of the CDB silently and without confirmation
- Changing `COMPATIBLE` is irreversible
- Might prevent future move or clone of the PDB

Compatible



Compatible



Keep **`COMPATIBLE`** at the same setting in all CDBs on the same release

```
--Allows CDB views to include information on PDB$SEED objects.
--By default, such information is hidden.
--https://mikedietrichde.com/2017/07/21/why-exclude_seed_cdb_view-is-now-an-underscore-in-oracle-12-2/

alter system set "_exclude_seed_cdb_view"=false;
```


```
alter session set container=CDB$ROOT;

alter system set "_exclude_seed_cdb_view"=false;

select con_id, tablespace_name
from   cdb_tablespaces;
```

```
CON_ID TABLESPACE_NAME
------ ---------------
     1 SYSTEM
     1 SYSAUX
     1 UNDOTBS1
     1 TEMP
     1 USERS
     2 SYSTEM
     2 SYSAUX
     2 UNDOTBS1
     2 TEMP
     4 SYSTEM
     4 SYSAUX
     4 UNDOTBS1
     4 TEMP
```


Create Container Database

https://mikedietrichde.com/2018/08/08/creating-cdbs-non-cdbs-with-less-options/
https://mikedietrichde.com/2017/07/11/always-create-custom-database/
https://mikedietrichde.com/2017/07/26/remove-clean-components-oracle-11-2-12-2/

Migration

Migration to multitenant is a one-time operation that requires downtime

- No downtime when using Oracle GoldenGate

Migration

1 Plug in

2 Convert

Migration

1 Plug in

First, check if database is compatible with CDB

1. Generate manifest file in non-CDB
2. Check compatibility in CDB

```
--In source, generate manifest file
--You can also generate a manifest file of a remote database using a db link
--You can generate a manifest file on a running database

exec dbms_pdb.describe('/tmp/DB19.xml');
```


Manifest File

What is a manifest file

- Data files
- Components
- Parameters
- Services
- Patch level
- Time zone

  ... and more

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
  <xmlversion>1</xmlversion>
  <pdbname>DB12</pdbname>
  <cid>0</cid>
  <byteorder>1</byteorder>
  <vsn>203424000</vsn>
  <vsns>
    <vsnnum>12.2.0.1.0</vsnnum>
    <cdbcompt>12.2.0.0.0</cdbcompt>
    <pdbcompt>12.2.0.0.0</pdbcompt>
    <vsnlibnum>0.0.0.0.24</vsnlibnum>
    <vsnsql>24</vsnsql>
    <vsnbsv>8.0.0.0.0</vsnbsv>
  </vsns>
  <dbid>1852833295</dbid>
  <ncdb2pdb>1</ncdb2pdb>
  <cdbid>1852833295</cdbid>
  <guid>86D5DC2587337002E0532AB2A8C0A57C</guid>
  <uscnbas>4437941</uscnbas>
  <uscnwrp>0</uscnwrp>
  <undoscn>8</undoscn>
  <rdba>4194824</rdba>
  <tablespace>
    <name>SYSTEM</name>
    <type>0</type>
    <tsn>0</tsn>
    <status>1</status>
    <issft>0</issft>
    <isnft>0</isnft>
    <encts>0</encts>
    <flags>0</flags>
    <bmunitsize>8</bmunitsize>
    <file>
      <path>/u02/oradata/DB12/system01.dbf</path>
      <afn>1</afn>
      <rfn>1</rfn>
```

```
--In CDB, check compatibility
--If PDB name changes, add parameter to check_plug_compatibility

set serveroutput on
BEGIN
   IF dbms_pdb.check_plug_compatibility('/tmp/DB19.xml') THEN
      dbms_output.put_line('PDB compatible? ==> Yes');
   ELSE
      dbms_output.put_line('PDB compatible? ==> No');
   END IF;
END;
/
```


```
--Always check the details

select type, message
from   pdb_plug_in_violations
where  name='DB19' and status<>'RESOLVED';

TYPE       MESSAGE
_____________________________________________________________________________________________________________
ERROR      '19.9.0 Release_Update' is installed in the CDB but no release updates are installed in the PDB
ERROR      DBRU bundle patch 201020: Not installed in the CDB but installed in the PDB
ERROR      PDB's version does not match CDB's version: PDB's version 12.2.0.1.0. CDB's version 19.0.0.0.0.
WARNING    CDB parameter compatible mismatch: Previous '12.2.0' Current '19.0.0'
WARNING    PDB plugged in is a non-CDB, requires noncdb_to_pdb.sql be run.
```


Always check `PDB_PLUG_IN_VIOLATIONS` after any plug-in operation

- An error prevents the PDB from opening unrestricted

Migration

1 Plug in

Then, perform plug-in

1. Shut down non-CDB
2. Plug into CDB

Migration

1. Restart database in read-only mode

```
SQL> shutdown immediate
SQL> startup mount
SQL> alter database open read only;
```

2. Generate manifest file and shut down

```
SQL> exec dbms_pdb.describe('/tmp/DB19.xml');
SQL> shutdown immediate;
```

3. In CDB, create PDB from manifest file

```
SQL> create pluggable database DB19 using '/tmp/DB19.xml';
```

Migration

1 Plug in

2 Convert

Migration

2 Convert

1. Complete conversion with `noncdb_to_pdb.sql`
2. Requires downtime, but you run it only once
3. Irreversible

Migration

1. Open PDB

```
SQL> alter pluggable database DB19 open;
SQL> alter session set container=DB19;
```

2. Convert

```
SQL> @?/rdbms/admin/noncdb_to_pdb.sql
```

3. Restart PDB

```
SQL> alter pluggable database DB19 close;
SQL> alter pluggable database DB19 open;
```

Migration

4. Check plug-in violations

```
SQL> select type, message
     from   pdb_plug_in_violations
     where  name='DB19' and status<>'RESOLVED';
```

5. Ensure PDB is open `READ WRITE` and unrestricted

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

6. Configure PDB to auto-start

```
SQL> alter pluggable database DB19 save state;
```

You can rename a database on plug-in

- PDB name is unique in a CDB
- Check `target_pdb_name`

The CDB registers all services from the non-CDB on plug-in

- Service name is unique in a CDB
- Other PDBs might use the same service name

Don't save state when Grid Infrastructure manages the database

- If you set a service to start in Grid Infrastructure, the corresponding PDB starts as well

Demo

Multitenant migration
19c non-CDB

Watch on YouTube

Downtime

How much downtime do we need?

`noncdb_to_pdb.sql`

- Runtime varies, typically 10-20 minutes
- Depends on dictionary complexity
- Not database size

Pre-tasks and post-tasks

- Follow best practices
- Data Guard

Expect total downtime 1-2 hours

- Simple databases even faster
- It's a migration, don't rush it

MIGRATION
options

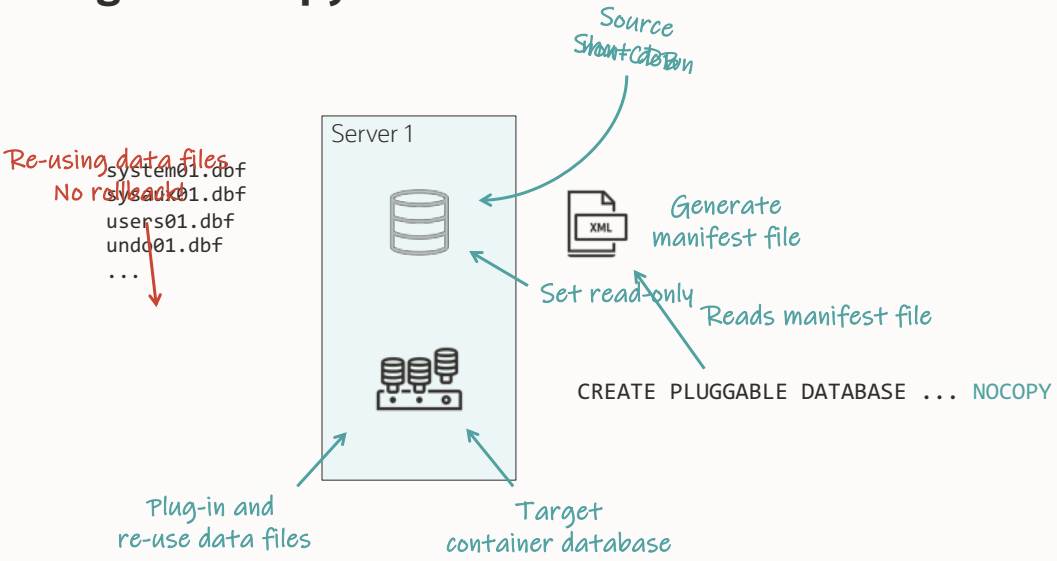
Other Options

Refreshable

Plug-in Copy

Plug-in NoCopy

Plug-in NoCopy



```
#AutoUpgrade config file
#Convert DB12 to PDB in CDB2 - re-use data files
upg1.source_home=/u01/app/oracle/product/19.22.0
upg1.target_home=/u01/app/oracle/product/19.23.0
upg1.sid=DB12
upg1.target_cdb=CDB2

#Fully automated migration
java -jar autoupgrade.jar -config db12.cfg -mode deploy
```

```
#AutoUpgrade config file
#Convert DB12 to PDB in CDB2 - re-use data files
#Upgrade from Oracle Database 19c to 23ai
upg1.source_home=/u01/app/oracle/product/19.22.0
upg1.target_home=/u01/app/oracle/product/23.4.0
upg1.sid=DB12
upg1.target_cdb=CDB2
```

Demo

Multitenant migration
Including upgrade to Oracle Database 23ai
Using AutoUpgrade

Watch on YouTube

Consider using **MOVE** instead of **NOCOPY** to move files into proper directory

- Use **FILE_NAME_CONVERT** clause

MIGRATION
options

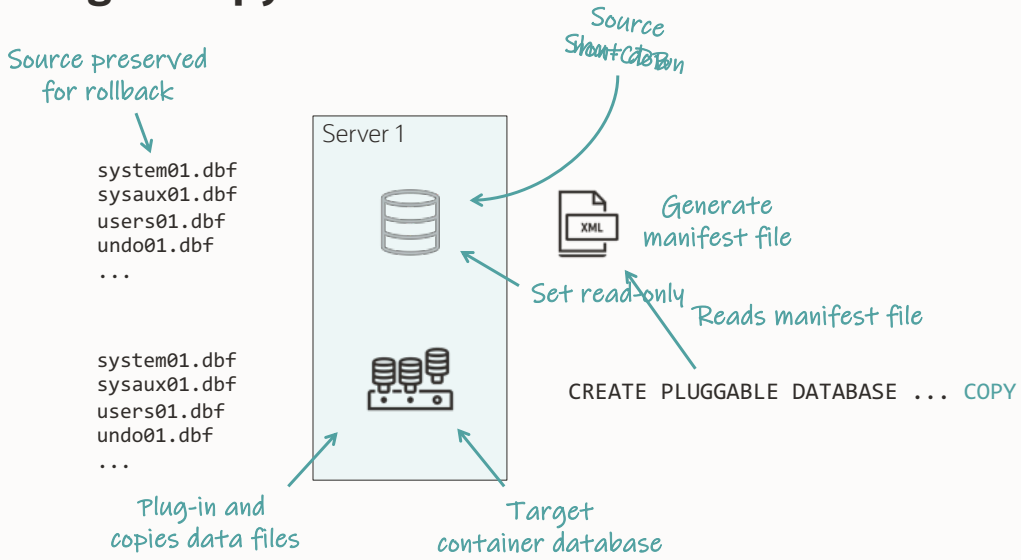
Other Options

Refreshable

Plug-in Copy

Plug-in NoCopy

Plug-in Copy



```
#AutoUpgrade config file
#Convert DB12 to PDB in CDB2 - copy data files
upg1.source_home=/u01/app/oracle/product/19.22.0
upg1.target_home=/u01/app/oracle/product/19.23.0
upg1.sid=DB12
upg1.target_cdb=CDB2
upg1.target_pdb_copy_option.db12=file_name_convert('/u01', '/u02')

#Fully automated migration
java -jar autoupgrade.jar -config db12.cfg -mode deploy
```


```
#AutoUpgrade config file
#Convert DB12 to PDB in CDB2 - copy data files
#Generate OMF names - also for ASM
upg1.source_home=/u01/app/oracle/product/19.22.0
upg1.target_home=/u01/app/oracle/product/19.23.0
upg1.sid=DB12
upg1.target_cdb=CDB2
upg1.target_pdb_copy_option.db12=file_name_convert=NONE
```

Be sure to shut down the source database.

Use *prefix*`.close_source=yes`

MIGRATION
options

Other Options

Refreshable

Plug-in Copy

Plug-in NoCopy

Refreshable Clone

CREATE

Create PDB from non-CDB over a database link

REFRESH

Apply redo from non-CDB to keep PDB up-to-date

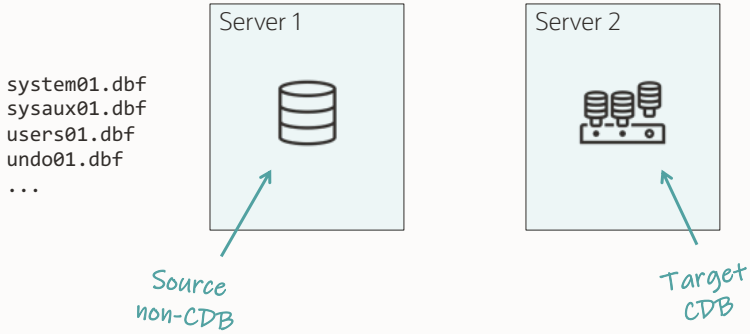
OUTAGE

Disconnect users and refresh PDB for the last time

CONVERT

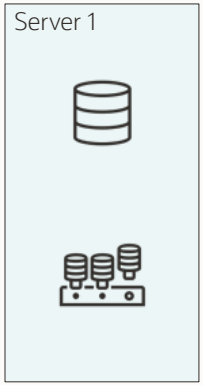
To become a proper PDB, it must be converted

Refreshable Clone PDB



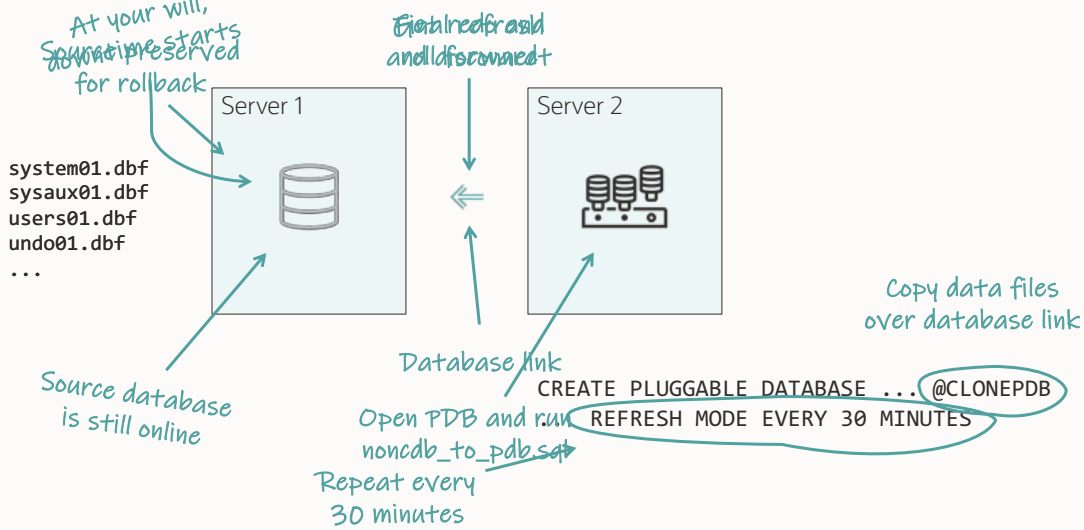
Refreshable Clone PDB

```
system01.dbf
sysaux01.dbf
users01.dbf
undo01.dbf
...
```



Could be same server as well

Refreshable Clone PDB



Refreshable Clone

Source non-CDB	Target CDB

```
CREATE USER dblinkuser
       IDENTIFIED BY ... ;

GRANT CREATE SESSION,
      CREATE PLUGGABLE DATABASE,
      SELECT_CATALOG_ROLE TO dblinkuser;

GRANT READ ON sys.enc$ TO dblinkuser;
```

```
CREATE DATABASE LINK CLONEPDB
       CONNECT TO dblinkuser
       IDENTIFIED BY ...
       USING 'noncdb-alias';
```

Refreshable Clone

Source non-CDB	Target CDB

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/23
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.source_dblink.NONCDB1=CLONEPDB
upg1.target_pdb_name.NONCDB1=PDB1

--Specify relative start time
--upg1.start_time=+1h30m
```

Refreshable Clone

Source non-CDB	Target CDB

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/23
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.source_dblink.NONCDB1=CLONEPDB 300
upg1.target_pdb_name.NONCDB1=PDB1

--Specify relative start time
--upg1.start_time=+1h30m
```

Refreshable Clone

Source non-CDB	Target CDB

```
upg1.source_home=/u01/app/oracle/product/19
upg1.target_home=/u01/app/oracle/product/23
upg1.sid=NONCDB1
upg1.target_cdb=CDB1
upg1.source_dblink.NONCDB1=CLONEPDB 300
upg1.target_pdb_name.NONCDB1=PDB1
upg1.start_time=19/05/2024 02:00:00
--Specify relative start time
--upg1.start_time=+1h30m
```

Refreshable Clone

1

Run on source

```
autoupgrade.jar ... –mode analyze
autoupgrade.jar ... –mode fixups
```

2

Run on target

```
autoupgrade.jar ... –mode deploy
```

Refreshable Clone


```
autoupgrade.jar ... –mode deploy
```

```
upg1.start_time=19/05/2024 02:00:00
```

The source non-CDB stays intact to allow rollback

Demo

Multitenant migration
Upgrade to Oracle Database 19c
Using Refreshable Clone PDBs

Watch on YouTube

Refreshable clone works only with deferred recovery on standby database

- You must restore the PDB on standby database after disconnect from non-CDB

Refreshable Clone PDB

- After creating the refreshable clone PDB, don't restart the source database

```
ORA-00283: recovery session canceled due to errors
ORA-65339: unsupported operation on the source PDB
```

- In the source database, refreshable clone PDB supports:
  - Creating new tablespaces
  - Extending existing data files
  - Adding new data files

Ensure archive logs are available on disk during migration

Works cross-platform, but not cross-endian

- Use cross-platform transportable tablespaces or Data Pump

Source database release must be eligible for direct upgrade to target release

- 19c to 23ai is possible
- 12.2 to 23ai is not possible

You can set source database in read-only mode before the final refresh

- Ideal for migrations

Cloning

CLONING

AutoUpgrade uses **CREATE PLUGGABLE DATABASE** statement with **PARALLEL** clause which clones the database using multiple parallel processes

PARALLEL

Based on system resources and current utilization the database automatically determines a proper parallel degree

TRANSFER

A new file transfer protocol that can bypass several layers in the database to achieve very high transfer rates

NETWORK

Watch out for network saturation. Control parallelism in the AutoUpgrade config file

The database applies automatic parallelism based on available system resources

- Control using *prefix*.parallel_pdb_creation_clause

```
SQL> select message, sofar, totalwork,time_remaining as remain, elapsed_seconds as ela
     from v$session_longops
     where opname='kpdbfCopyTaskCbk' and sofar != totalwork;
```

MESSAGE	SOFAR	TOTALWORK	REMAIN	ELA
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 643199 out of 1310720 Blocks done	643199	1310720	134	129
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 443007 out of 1310720 Blocks done	443007	1310720	213	109
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 436351 out of 1310720 Blocks done	436351	1310720	216	108
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 370431 out of 1310720 Blocks done	370431	1310720	256	101

```
SQL> select message, sofar, totalwork,time_remaining as remain, elapsed_seconds as ela
     from v$session_longops
     where opname='kpdbfCopyTaskCbk' and sofar != totalwork;
```

MESSAGE	SOFAR	TOTALWORK	REMAIN	ELA
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 643199 out of 1310720 Blocks done	643199	1310720	134	129
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 443007 out of 1310720 Blocks done	443007	1310720	213	109
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 436351 out of 1310720 Blocks done	436351	1310720	216	108
kpdbfCopyTaskCbk: /u01/app/oracle/oradata/CDB2/EDA 3: 370431 out of 1310720 Blocks done	370431	1310720	256	101

```
SQL> select sql_text
     from v$sql s, v$session_longops l
     where s.sql_id=l.sql_id and l.opname='kpdbfCopyTaskCbk';
```

```
SQL_TEXT
/* SQL Analyze(256,0) */ SELECT /*+PARALLEL(4) NO_STATEMENT_QUEUING  */ * FROM X$KXFTASK /*kpdbfParallelCopyOrMove,PDB_FILE_COPY*/
```


Zürcher
Kantonalbank

Customer Case | Zürcher Kantonalbank

Customer

Project

Constraints

Preparation

Migration

Success?

Remarks

A reliable partner for over 150 years

- The bank for the people of Zurich since 1870
- With over 5'100 employees one of the largest employers in the canton of Zurich
- Globally networked full-service bank with strong regional and local roots

Customer Case | Zürcher Kantonalbank

Customer

Project

Constraints

Preparation

Migration

Success?

Remarks

Current situation

- Oracle databases on old OS and on Oracle Exadata
- 2023:
  - Migrate everything to Exadata until end of 2023
  - Consolidation to Multitenant and to the next long-term support release

Planned solution: AutoUpgrade

Customer Case | Zürcher Kantonalbank

Customer
Project
Constraints
Preparation
Migration
Success?
Remarks

Test setup

- 3 non-CDB databases of different size

Source	Size / GB
TEST40 (108)	165
TEST42 (107)	555
TEST41 (106)	18'496

- Exadata X6-2 compute node
- 7 storage cells (2x X6-2L / 3x X7-2L / 2x X8-2L)
- Oracle Database 19.15.0
- No additional options

Customer Case | Zürcher Kantonalbank

Customer

Project

Constraints

Preparation

Migration

Success?

Remarks

Cloning user

```
create user dblinkuser identified by Oracle_4UOracle_4U;
```

Permissions

```
grant CONNECT, RESOURCE, CREATE PLUGGABLE DATABASE,
      SELECT_CATALOG_ROLE to dblinkuser;
grant ALL ON SYS.ENC$ to dblinkuser;
```

Database link

```
create database link TEST42.DOMAIN connect to dblinkuser
identified by oracle_4uoracle_4u using ‘test42.domain’;
```

Customer Case | Zürcher Kantonalbank

Customer

Project

Constraints

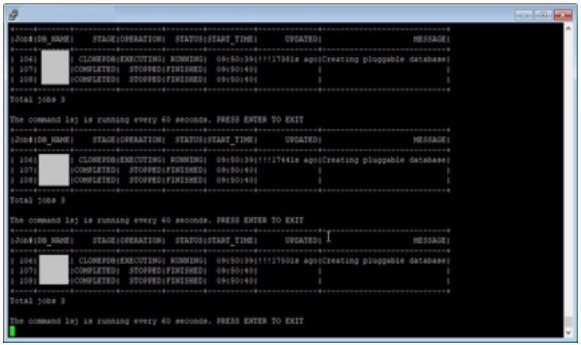
Preparation

Migration

Success?

Remarks

Migration in progress

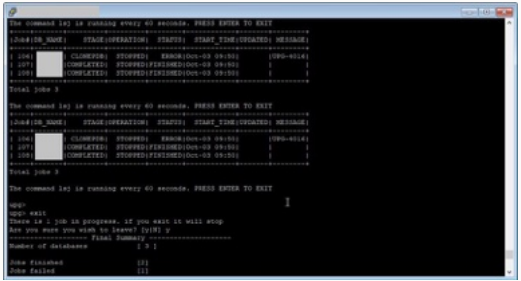


Source	Runtime/Min
TEST40 (108)	26
TEST42 (107)	ongoing
TEST41 (106)	ongoing

Customer Case | Zürcher Kantonalbank

- Customer
- Project
- Constraints
- Preparation
- **Migration**
- Success?
- Remarks

Migration completed



Source	Runtime/Min
TEST40 (108)	26
TEST42 (107)	226 (~3.5h)
TEST41 (106)	1770 (29h)

Customer Case | Zürcher Kantonalbank

Customer

Project

Constraints

Preparation

Migration

Success?

Remarks

First non-CDBs migrated successfully

- Project is ongoing

Customer Case | Zürcher Kantonalbank

Customer

Project

Constraints

Preparation

Migration

Success?

Remarks

For large databases, make sure archives aren't cleaned up

- Solution: restore archivelogs from backup

User profile with IDLE_TIME lead to kill of the session

- Solution: assign a different profile to the clone user

Summary

- Very comfortable to use
  - Everything happens automatically
  - Does not require user interaction
- Simple syntax
- No license costs associated
- Perfect for pre-migration test

- Very Stable

MIGRATION
options

Other Options

Refreshable

Plug-in Copy

Plug-in NoCopy

Other Options

It is also possible to migrate using

1. Data Pump
2. Transportable Tablespaces
3. GoldenGate

- Well-known and proven method
- Extremely flexible
- Migrate from lower version
- Migrate from cross-Endian
- Preserves source database for fallback

Other Options

It is also possible to migrate using

1. Data Pump
2. Transportable Tablespaces
3. GoldenGate

- Faster for larger databases
- Migrate from lower version
- Migrate from cross-Endian
- Preserves source database for fallback

Other Options

It is also possible to migrate using

1. Data Pump
2. Transportable Tablespaces
3. GoldenGate

- Only zero downtime option
- Migrate from lower version
- Migrate from cross-Endian
- Preserves source database for fallback
- Active-active replication for ultimate solution

comparing MIGRATION options

	Plug-in NoCopy	Plug-in Copy	Refreshable Clone PDB	Data Pump	Transportable	GoldenGate
Downtime	Less	Considerable	Less	Considerable	Minimal	None
Rollback	No	Yes	Yes	Yes	Yes	Yes
Cross-platform (same Endian)	Yes	Yes	Yes	Yes	Yes	Yes
Cross-Endian	No	No	No	Yes	Yes	Yes
Cross-version	No	No	Yes	Yes	Yes	Yes
Complexity	Low	Low	Low	Medium	Medium	High

Best Practices for multitenant migration

Gather dictionary stats before migration

- Preferably also after migration

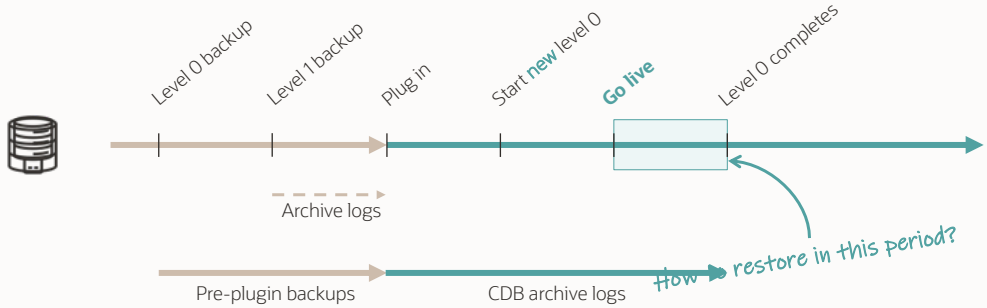
Perform a dictionary check before migration

- Use `DBMS_DICTIONARY_CHECK`

Backup your database after migration

- Level 0

Recovery Strategy



Practice restore with pre-plugin backups

- Check `DBMS_PDB.EXPORTRMANBACKUP`

What about Oracle RAC Database

RAC

- Nothing special, it works seamlessly
- Connect to one instance and plug in
- No need to start with `CLUSTER_DATABASE=FALSE`

What happens during plug-in with Oracle Data Guard

Data Guard

Plug-in on primary propagates to standby database via redo

1 Enabled recovery

2 Deferred recovery

Data Guard

1

Enabled recovery

```
create pluggable database ... standbys=all
```

Standby records PDB creation

Standby locates data files

MRP applies redo to PDB

PDB is immediately protected

Default

2

Deferred recovery

Data Guard

1

Enabled recovery

```
create pluggable database ... standbys=all
```

Standby records PDB creation

Standby locates data files

MRP applies redo to PDB

PDB is immediately protected

2

Deferred recovery

```
create pluggable database ... standbys=none
```

Standby records PDB creation

Standby ignores data files

MRP skips redo

PDB protected after restore

You can specify recovery mode for each standby database

- `CREATE PLUGGABLE DATABASE ... STANDBYS=STDBY1,STDBY3`
- `CREATE PLUGGABLE DATABASE ... STANDBYS=ALL EXCEPT STDBY2`

```
--Check the recovery mode of each PDBs
--Possible values: ENABLED, DISABLED

select name, recovery_status from v$pdbs;
```


In AutoUpgrade, specify recovery mode using *prefix*.manage_standbys_clause

- Value is inserted directly into **STANDBYS** clause in **CREATE PLUGGABLE DATABASE** statement

Enabled Recovery

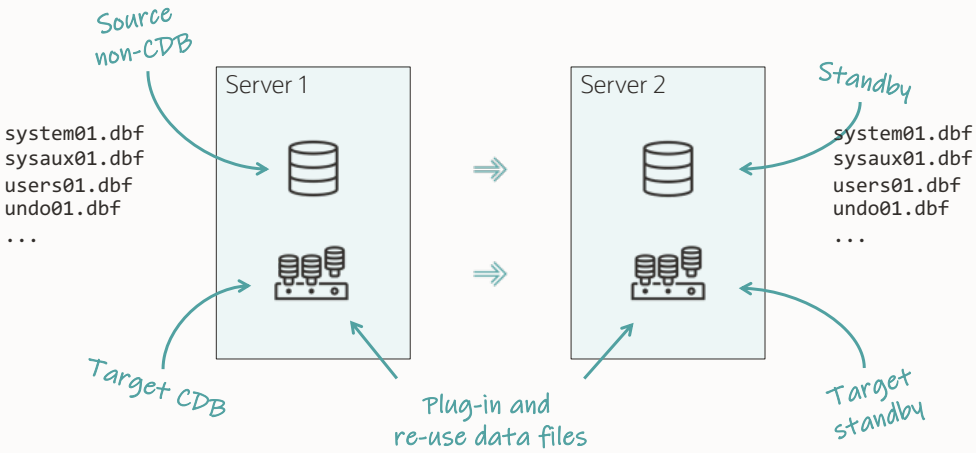
- A more complex approach
- Requires additional work before and during plug-in
- Standby database protects the PDB immediately after plug-in
- When you use `STANDBYS=ALL` or a list
- Default

Enabled Recovery

Enabled recovery applies to:

- All standby databases when using `STANDBYS=ALL`
- Or, those standby databases mention in `STANDBYS=<list>`
- Or, those not mentioned in `STANDBYS=ALL EXCEPT <list>`

Enabled Recovery



All data files on primary and standby must be at the same SCN

Reusing Data Files

Primary

```
SQL> shutdown immediate
SQL> startup mount
SQL> alter system
     flush redo to stdby no confirm apply;
SQL> alter database open read only;
SQL> select checkpoint_change#
     from v$datafile_header where file#=1;
```


```
SQL> exec dbms_pdb.describe('/home/oracle/prmy.xml');
SQL> shutdown immediate
```

Standby

```
DGMGRL> edit database stdby set state='APPLY-OFF';
SQL> shutdown immediate
SQL> startup
SQL> alter database
     recover managed standby database cancel;
```


```
SQL> alter database recover managed standby database
     until change 2319950;
SQL> select checkpoint_change#
     from v$datafile_header where file#=1;
```

```
SQL> shutdown immediate
```

Enabled Recovery

- The plug-in happens on the primary database
- The plug-in uses the manifest file
- The manifest file contains information on data files from the primary database only

How does the standby database know which files to plug in?

Enabled Recovery

How does the standby database know which files to plug in?

1. Regular files
2. OMF in regular file system
3. ASM

Enabled Recovery

1 Regular files

- Standby search for data files at the same location as the primary
- Override with `DB_FILE_NAME_CONVERT`
- Or, override with `STANDBY_PDB_SOURCE_FILE_DIRECTORY`

Enabled Recovery

2 OMF in regular file system

- Standby search for data files at the OMF location (`DB_CREATE_FILE_DEST`)
- Move data files from non-CDB location into OMF location
- Or, create soft links in OMF location pointing to data file location

Enabled Recovery

3 ASM

- Standby search for data files at the OMF location (`DB_CREATE_FILE_DEST`)
- Now, it becomes a little tricky…

Enabled Recovery | ASM

23ai
Non-CDB
Primary

23ai
Non-CDB
Standby

```
SQL> select name from v$datafile;

NAME
--------------------------------------------------------
+DATA/DB_BOSTON/DATAFILE/system.269.1103046537
+DATA/DB_BOSTON/DATAFILE/sysaux.270.1103046537
+DATA/DB_BOSTON/DATAFILE/users.273.1103046827
```

```
SQL> select name from v$datafile;

NAME
--------------------------------------------------------
+DATA/DB_CHICAGO/DATAFILE/system.265.1103050007
+DATA/DB_CHICAGO/DATAFILE/sysaux.266.1103050007
+DATA/DB_CHICAGO/DATAFILE/users.269.1103050009
```

Same file,
but different name

Enabled Recovery | ASM

23ai
Non-CDB
Primary

23ai
Non-CDB
Standby

The manifest file contains

- File path on primary database only
- Not standby database

```
SQL> exec dbms_pdb.describe('/tmp/manifest_DB.xml');
```

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
  <xmlversion>1</xmlversion>
  <pdbname>PDB1</pdbname>
  ...
  <guid>DDB49CFEFD8ED4FCE053E801000A078C</guid>
  ...
  <tablespace>
    <name>USERS</name>
    ...
    <file>
       <path>+DATA/DB_BOSTON/DATAFILE/users.273.1103046827</path>
```

Enabled Recovery | ASM

Target primary

23ai
CDB
Primary

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

- Manifest file lists the location of data files on primary
- No information about standby databases

23ai
CDB
Standby

Target standby

Enabled Recovery | ASM

23ai
CDB
Primary

+DATA/DB_BOSTON/DATAFILE/users.273.1103046827

Redo record says:
Plug in this data file

No good, data file
has a different name

23ai
CDB
Standby

+DATA/DB_CHICAGO/DATAFILE/users.269.1103050009

Enabled Recovery | ASM

23ai
CDB
Primary

+DATA/DB_BOSTON/DATAFILE/users.273.1103046827

OK, let's check the OMF directory

23ai
CDB
Standby

+DATA/DB_CHICAGO/DATAFILE/users.269.1103050009

+DATA/CDB1_CHICAGO/<PDB_GUID>/DATAFILE

It's empty

Enabled Recovery | ASM

23ai
CDB
Primary

+DATA/DB_BOSTON/DATAFILE/users.273.1103046827

OK, let's check the OMF directory

23ai
CDB
Standby

+DATA/DB_CHICAGO/DATAFILE/users.269.1103050009

+DATA/CDB1_CHICAGO/<PDB_GUID>/DATAFILE

It's empty

I'll just move the file in ASM

There's no move command in ASM.
How about copying?

```
ASMCMD> cp users.269.1103050009
        +DATA/DB_CHICAGO/.../users.273.1103046827

ASMCMD-8016: copy source '+DATA/DB_BOSTON/.../users.269.1103050009' and target
'+DATA/DB_CHICAGO/.../users.273.1103046827' failed
ORA-15056: additional error message
ORA-15046: ASM file name 'users.273.1103046827' is not in single-file creation form
ORA-06512: at "SYS.X$DBMS_DISKGROUP", line 617
ORA-06512: at line 3 (DBD ERROR: OCIStmtExecute)
```

Only a database can produce files with ASM/OMF data file names

There's no `move` command in ASM.
But you can create *aliases*

- Similar to file system soft links

```
SQL> alter diskgroup data add alias
     '+DATA/DB_CHICAGO/DATAFILE/users.269.1103050009'
     for
     '+DATA/CDB1_CHICAGO/<PDB_GUID>/DATAFILE/users.269.1103050009':
```



```
SQL> alter diskgroup data add alias
     '+DATA/DB_CHICAGO/DATAFILE/users.269.1103050009'
     for
     '+DATA/CDB1_CHICAGO/<PDB_GUID>/DATAFILE/users.269.1103050009':
```

```
SQL> alter diskgroup data add alias
     '+DATA/DB_CHICAGO/DATAFILE/users.269.1103050009'
     for
     '+DATA/CDB1_CHICAGO/<PDB_GUID>/DATAFILE/users.269.1103050009':
```

Standby PDB OMF location

```
SQL> alter diskgroup data add alias
     '+DATA/DB_CHICAGO/DATAFILE/users.269.1103050009'
     for
     '+DATA/CDB1_CHICAGO/<PDB_GUID>/DATAFILE/users.269.1103050009':
```

Name does not matter.
Standby scans all files in OMF directory

Data Guard | Re-use Data Files

23ai
CDB
Primary

23ai
CDB
Standby

- Standby database scans its own OMF directory for data files
- Standby ignores file names and look at file headers
- Standby will find aliases and find the real file locations

Data Guard | Re-use Data Files

23ai CDB Primary

23ai CDB Standby

Looking for file like on primary

```
Recovery scanning directory +DATA/DB_BOSTON/... for any matching files
Deleted Oracle managed file +DATA/DB_BOSTON/...
Successfully added datafile 37 to media recovery
Datafile #37: +DATA/DB_CHICAGO/DATAFILE/users.269.1103050009
```

Follows alias and finds the real file

Demo

Multitenant migration
19c non-CDB to 23ai
With Data Guard and re-using data files

Watch on YouTube

Move data files into proper OMF location and get rid of aliases

Data File Location


```
SQL> select file#, name from v$datafile;

FILE# NAME
----- --------------------------------------------------
14     +DATA/DB_BOSTON/DATAFILE/system.269.1103046537
15     +DATA/DB_BOSTON/DATAFILE/sysaux.270.1103046537
16     +DATA/DB_BOSTON/DATAFILE/users.273.1103046827
```

Non-CDB OMF location

The database does not care.
But humans might care

- A database can use files from a non-OMF location

Online Data File Move

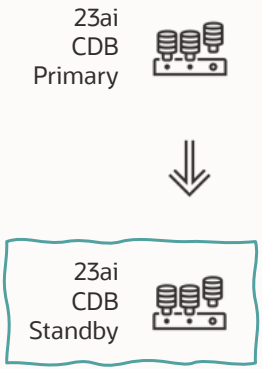

```
SQL> ALTER DATABASE MOVE DATAFILE 14;

SQL> select file#, name from v$datafile;

FILE# NAME
----- ------------------------------------------------------------
14     +DATA/CDB_BOSTON/<PDB GUID>/DATAFILE/system.269.1103046537
15     +DATA/DB_BOSTON/DATAFILE/sysaux.270.1103046537
16     +DATA/DB_BOSTON/DATAFILE/users.273.1103046827
```

- The database copies the file, then drops the alias and original file
- Requires additional disk space
- Online operation

Online Data File Move



- On standby, online datafile move works only when CDB and PDB are open
- Stop redo apply before opening, unless you have a license for Active Data Guard
- Requires time and disk space to hold a copy of the data file
- Removes ASM alias and original file upon completion

```
DGMGRL> edit database <STANDBY_UNQNAME> set state='apply-off';
```


```
DGMGRL> edit database <STANDBY_UNQNAME> set state='apply-off';
```

```
SQL> alter database open read only;
SQL> alter pluggable database PDB1 open read only;
SQL> alter session set container=PDB1;
SQL> alter database move datafile <file#>;
SQL> alter database move datafile <file#>;
SQL> ...
SQL> alter database move datafile <file#>;
```


```
DGMGRL> edit database <STANDBY_UNQNAME> set state='apply-off';

SQL> alter database open read only;
SQL> alter pluggable database PDB1 open read only;
SQL> alter session set container=PDB1;
SQL> alter database move datafile <file#>;
SQL> alter database move datafile <file#>;
SQL> ...
SQL> alter database move datafile <file#>;

srvctl stop database -d $ORACLE_UNQNAME -o immediate
srvctl start database -d $ORACLE_UNQNAME -o mount
```


```
DGMGRL> edit database <STANDBY_UNQNAME> set state='apply-off';

SQL> alter database open read only;
SQL> alter pluggable database PDB1 open read only;
SQL> alter session set container=PDB1;
SQL> alter database move datafile <file#>;
SQL> alter database move datafile <file#>;
SQL> ...
SQL> alter database move datafile <file#>;

srvctl stop database -d $ORACLE_UNQNAME -o immediate
srvctl start database -d $ORACLE_UNQNAME -o mount

DGMGRL> edit database <STANDBY_UNQNAME> set state='apply-on';
```

```
DGMGRL> edit database <STANDBY_UNQNAME> set state='apply-off';

SQL> alter database open read only;
SQL> alter pluggable database PDB1 open read only;
SQL> alter session set container=PDB1;
SQL> alter database move datafile <file#>;
SQL> alter database move datafile <file#>;
SQL> ...
SQL> alter database move datafile <file#>;

srvctl stop database -d $ORACLE_UNQNAME -o immediate
srvctl start database -d $ORACLE_UNQNAME -o mount

DGMGRL> edit database <STANDBY_UNQNAME> set state='apply-on';
```

Just move data files,
if Active Data Guard

While relocating data files, apply lag increases if redo apply is off

- Redo transport is still active
- Switchover/failover time increases

What happens with enabled recovery
if the standby fails to find the data files?

Enabled Recovery | Missing Data Files

What if a standby database fails to find data files?

- If Active Data Guard and PDB Recovery Isolation is turned on
  - New feature in Oracle Database 21c
  - Recovery disabled for PDB
  - Recovery proceeds in the entire CDB, except in specific PDB
  - Standby automatically restores data files from primary and re-enables recovery afterward
  - PDB protected after auto-restore

- If not, recovery halts in the **entire** CDB
  - **This is a critical situation**

What about AutoUpgrade and enabled recovery?

Enabled Recovery | AutoUpgrade

The current version (24.1) does not support plugging in with enabled recovery

- Enabled recovery requires work on both primary and standby hosts
- You must execute commands at specific times
- It's complicated - but we're working on it

What about **AS CLONE** clause

- `CREATE PLUGGABLE DATABASE ... AS CLONE`

Enabled Recovery | As Clone

- When you plug in a non-CDB the GUID doesn't change
  - Get GUID from the manifest file or `V$CONTAINERS`
  - GUID only changes when you use `AS CLONE` keyword

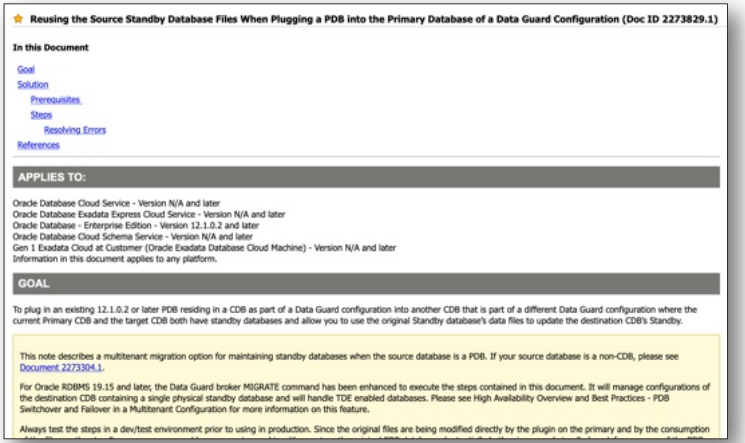
- Impossible to re-use standby data files when
  - Using OMF and `CREATE PLUGGABLE DATABASE ... AS CLONE`
  - Regardless of whether you use regular file system or ASM
  - Only works with regular files and non-OMF

Enabled Recovery | As Clone

- Regular file system and non-OMF
  - Put data files at the same location as primary data files
  - Take `FILE_NAME_CONVERT` into account (`CREATE PLUGGABLE DATABASE`)
  - Adjust according to `DB_FILE_NAME_CONVERT`

Data Guard | Enabled Recovery

Reusing the Source Standby Database Files When Plugging a PDB into the Primary Database of a Data Guard Configuration (Doc ID 2273829.1)



Reusing the Source Standby Database Files When Plugging a PDB into the Primary Database of a Data Guard Configuration (Doc ID 2273829.1)

In this Document

Goal
Solution
Prerequisites
Steps
Resolving Errors
References

APPLIES TO:

Oracle Database Cloud Service - Version N/A and later
Oracle Database Exadata Express Cloud Service - Version N/A and later
Oracle Database - Enterprise Edition - Version 12.1.0.2 and later
Oracle Database Cloud Schema Service - Version N/A and later
Gen 1 Exadata Cloud at Customer (Oracle Exadata Database Cloud Machine) - Version N/A and later
Information in this document applies to any platform.

GOAL

To plug in an existing 12.1.0.2 or later PDB residing in a CDB as part of a Data Guard configuration into another CDB that is part of a different Data Guard configuration where the current Primary CDB and the target CDB both have standby databases and allow you to use the original Standby database's data files to update the destination CDB's Standby.

This note describes a multitenant migration option for maintaining standby databases when the source database is a PDB. If your source database is a non-CDB, please see Document 2273304.1.

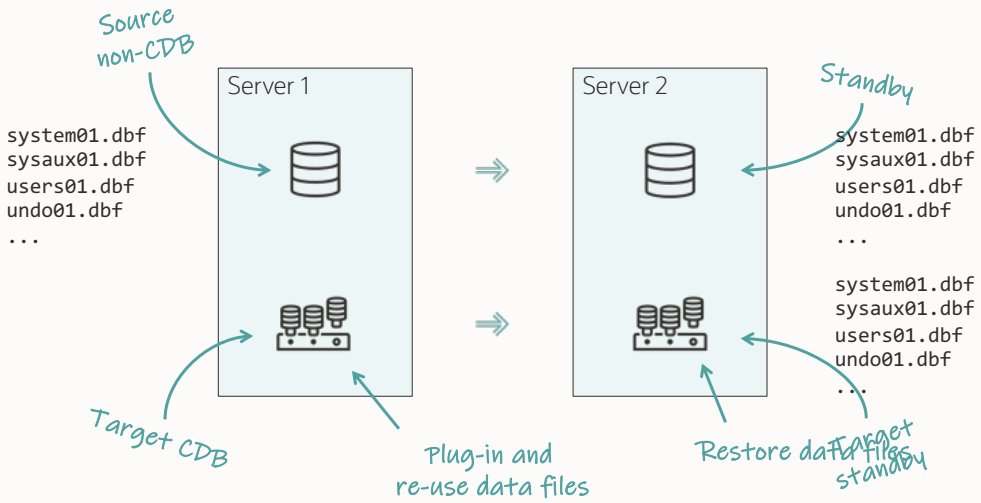
For Oracle RDBMS 19.15 and later, the Data Guard broker MIGRATE command has been enhanced to execute the steps contained in this document. It will manage configurations of the destination CDB containing a single physical standby database and will handle TDE enabled databases. Please see High Availability Overview and Best Practices - PDB Switchover and Failover in a Multitenant Configuration for more information on this feature.

Always test the steps in a dev/test environment prior to using in production. Since the original files are being modified directly by the plugin on the primary and by the consumption

Deferred Recovery

- The simplest approach
- Requires additional work after plug-in
- You must restore the PDB and re-enabledrecovery
- Standby database protects the PDB after restore
- When you use `STANDBYS=NONE`

Deferred Recovery



Deferred Recovery

Deferred recovery applies to:

- All standby databases when using `STANDBYS=NONE`
- Or, those standby databases mention in `STANDBYS=ALL EXCEPT <list>`
- Or, those not mentioned in `STANDBYS=<list>`

Deferred Recovery

Source
Non-CDB

Target
Primary

```
SQL> create pluggable database ...
     standbys=none;
```

Target
Standby

Deferred Recovery

```
SQL> create pluggable database ...
     standbys=none;
```

PDB created
Data files missing

Deferred Recovery

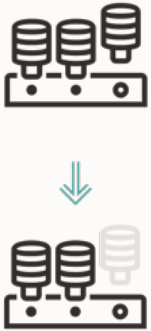
```
SQL> show pdbs

CON_NAME    OPEN MODE
PDB1        READ WRITE
```

```
SQL> show pdbs

CON_NAME    OPEN MODE
PDB1        MOUNTED
```

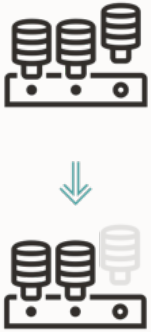
Deferred Recovery



```
SQL> select name, recovery_status
     from v$pdbs;

NAME    RECOVERY_STATUS
PDB1    DISABLED
```

Deferred Recovery



```
RMAN> restore pluggable database
      ... from service ... ;

SQL> alter pluggable database
     enable recovery;
SQL> alter database datafile
     ... online;
```

Deferred Recovery

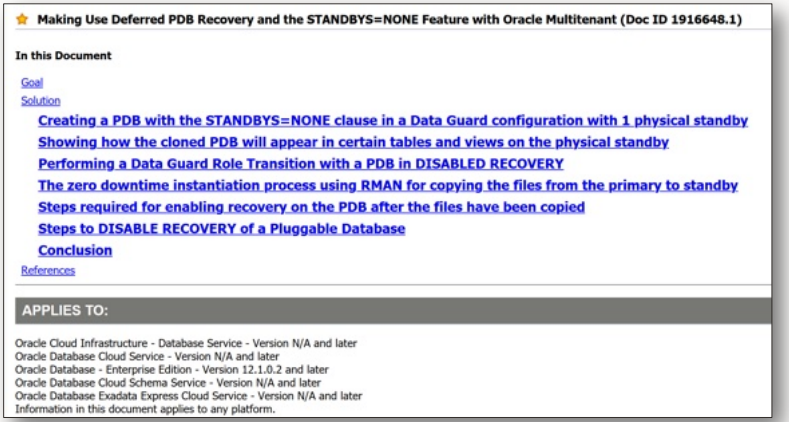
```
RMAN> restore pluggable database
      ... from service ... ;

SQL> alter pluggable database
     enable recovery;
SQL> alter database datafile
     ... online;
```

- Automated process in Oracle Database 21c
- PDB Recovery Isolation
- Requires Active Data Guard

Data Guard | Deferred Recovery

Making Use Deferred PDB Recovery and the STANDBYS=NONE Feature with Oracle Multitenant (Doc ID 1916648.1)



Making Use Deferred PDB Recovery and the STANDBYS=NONE Feature with Oracle Multitenant (Doc ID 1916648.1)

In this Document

Goal
Solution
- **Creating a PDB with the STANDBYS=NONE clause in a Data Guard configuration with 1 physical standby**
- **Showing how the cloned PDB will appear in certain tables and views on the physical standby**
- **Performing a Data Guard Role Transition with a PDB in DISABLED RECOVERY**
- **The zero downtime instantiation process using RMAN for copying the files from the primary to standby**
- **Steps required for enabling recovery on the PDB after the files have been copied**
- **Steps to DISABLE RECOVERY of a Pluggable Database**
- **Conclusion**

References

APPLIES TO:

Oracle Cloud Infrastructure - Database Service - Version N/A and later
Oracle Database Cloud Service - Version N/A and later
Oracle Database - Enterprise Edition - Version 12.1.0.2 and later
Oracle Database Cloud Schema Service - Version N/A and later
Oracle Database Exadata Express Cloud Service - Version N/A and later
Information in this document applies to any platform.

Data Guard | Additional Information

Data Guard Impact on Oracle Multitenant Environments (Doc ID 2049127.1)

The physical standby database and redo apply will normally expect a new PDB's datafiles to have been pre-copied to the standby site and be in such a state that redo received from the primary database can be immediately applied. The standby database ignores any file name conversion specification on the CREATE PLUGGABLE DATABASE statement and relies solely on the standby database's initialization parameter settings for DB_CREATE_FILE_DEST and DB_FILE_NAME_CONVERT for locations and file naming.

For these cases, Oracle recommends deferring recovery of the PDB using the STANDBYS=NONE clause on the CREATE PLUGGABLE DATABASE statement. Recovery of the PDB can be enabled at some point in the future once the PDB's data files have been copied from the primary database to the standby database in a manner similar to that documented in Document 1916648.1.

Don't jeopardize your Data Guard

- Test the procedure and verify before go-live

How a customer handled the migration

- Customer case

Customer Case

Customer | Project | Result | Learnings

- **Swisscom** - Switzerland's leading telco
- One of the leading IT companies in Switzerland
- One of Switzerland’s most sustainable and innovative companies

Customer Case

Customer Case

Customer | Project | Result | Learnings

Oracle Siebel CRM

Non-CDB

- Database size: 18 TB
- Release: 19.17.0
- Average active users: 3000

Single-tenant architecture

Customer Case

Customer | Project | Result | Learnings

Oracle Siebel CRM

Non-CDB

- 6,000 tables
- 29,000 indexes
- Partitioning
- LOBs
- 51 bigfile tablespaces

Single-tenant architecture

Customer Case

Data Guard with five standby databases

Each database is a 4-node RAC database

Running on Exadata Database Machine

Streaming data to microservices using GoldenGate

After consulting the business,
a plan was made for the standby databases

Customer Case

Standby 1: DR

Re-use data files, ASM alias

Standby 2-3: Auxiliary DR

Restore data files

Standby 4-5: Reporting

Restore data files

Customer Case

Customer | Project | **Result** | Learnings

- Total downtime was 3 hours 30 minutes
  - Planned maintenance window was 4 hours
  - Most time spent on application work and GoldenGate configuration
- Database migration
  - `noncdb_to_pdb.sql`: 18 minutes

Customer Case

Customer | Project | Result | Learnings

1. Test, test, and test
2. Create a detailed runbook
3. Remove complexity from the project to the extent possible
4. Work together and double-check all actions

Converting on Exadata Database Service and Exadata Cloud@Customer

Exadata Database Service

1. Use tooling to create a new CDB - or use an existing one
2. Plug in and convert using a method of choice
3. Tooling automatically adapts the new PDB after a while

Converting Oracle E-Business Suite

E-Business Suite

- Oracle E-Business Suite Release 12.2 and 12.1.3 support multitenant architecture
- But only in single-tenant architecture
- Useful MOS notes
  - FAQ: Oracle E-Business Suite and the Oracle Multitenant Architecture (Doc ID 2567105.1)
  - Interoperability Notes: Oracle E-Business Suite Release 12.2 with Oracle Database 19c (Doc ID 2552181.1)
  - Getting Started with Oracle E-Business Suite on Oracle Cloud Infrastructure (Doc ID 2517025.1)
  - Using Oracle 19c Oracle RAC Multitenant (Single PDB) with Oracle E-Business Suite Release 12.2 (Doc ID 2530665.1)

Plugging in copies of the same database

As Clone

Each PDB has a unique GUID
- Check `V$CONTAINERS`

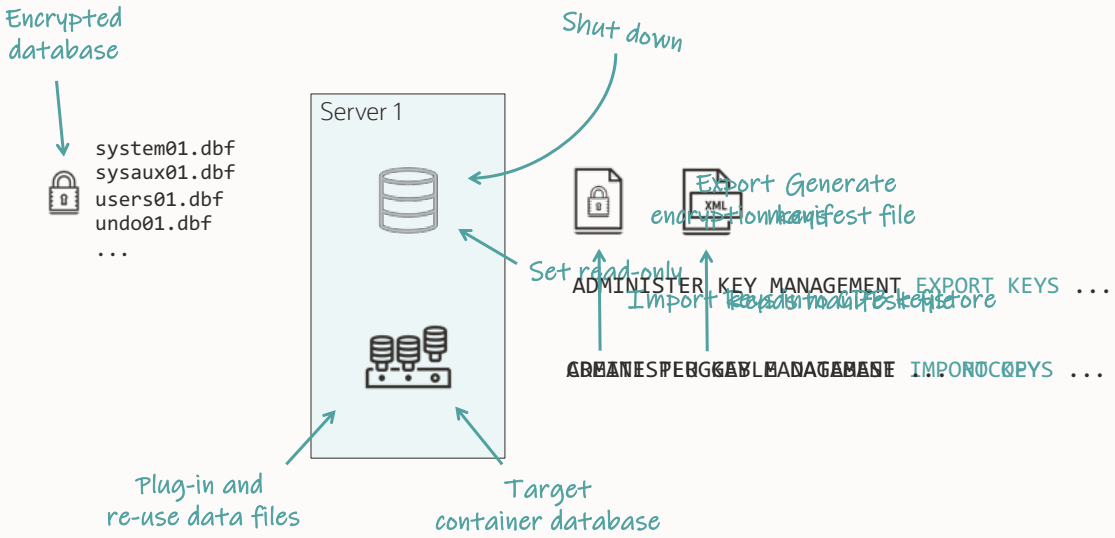
If you plug in the same database multiple times,
there are conflicting GUIDs

Use `CREATE PLUGGABLE DATABASE ... AS CLONE` to generate new GUIDs on plug-in

Migrating encrypted databases

- TDE Tablespace Encryption

Encrypted Database



AutoUpgrade fully supports encrypted databases

Encrypted Database

Certain database operations require passwords or secrets

```
CREATE PLUGGABLE DATABASE ... KEYSTORE IDENTIFIED BY <password>

ALTER PLUGGABLE DATABASE ... UNPLUG INTO ... ENCRYPT USING <secret>

CREATE PLUGGABLE DATABASE ... DECRYPT USING <secret>

ADMINISTER KEY MANAGEMENT ... KEYSTORE IDENTIFIED BY <password>
```

Secure External Password Store

Operator stores database keystore password
in a Secure External Password Store

AutoUpgrade Keystore

Operator loads database keystore password
into AutoUpgrade keystore ahead of upgrade

```
# Configure AutoUpgrade to work on encrypted databases
# Specify path for AutoUpgrade keystore

global.keystore=/etc/oracle/keystores/autoupgrade/DB12
global.autoupg_log_dir=/u01/app/oracle/cfgtoollogs/autoupgrade
upg1.source_home=/u01/app/oracle/product/12.2.0.1
upg1.target_home=/u01/app/oracle/product/19
upg1.sid=DB12
```


Encrypted Database

Analyze the database for upgrade readiness

```
$ java -jar autoupgrade.jar -config PDB1.cfg -mode analyze
```

Summary report will show which keystore passwords are needed

```
REQUIRED ACTIONS
================
      1.  Perform the specified action ...
      ORACLE_SID                      Action Required
      ------------------------------  ------------------------
      CDB1                            Add TDE password
      CDB2                            Add TDE password
```

Demo

Multitenant migration
Including upgrade to Oracle Database 19c
Using AutoUpgrade

Watch on YouTube

Oracle Key Vault

You can migrate an encrypted non-CDB using Oracle Key Vault

- Deploy an additional OKV client on the target database
- Configure both endpoints to point to the same default keystore

In the unlikely event of …

- Rollback and fallback options

PDB conversion is irreversible

- Not even Flashback Database can help

Rollback Options | Before Go-Live

1 Leave a copy

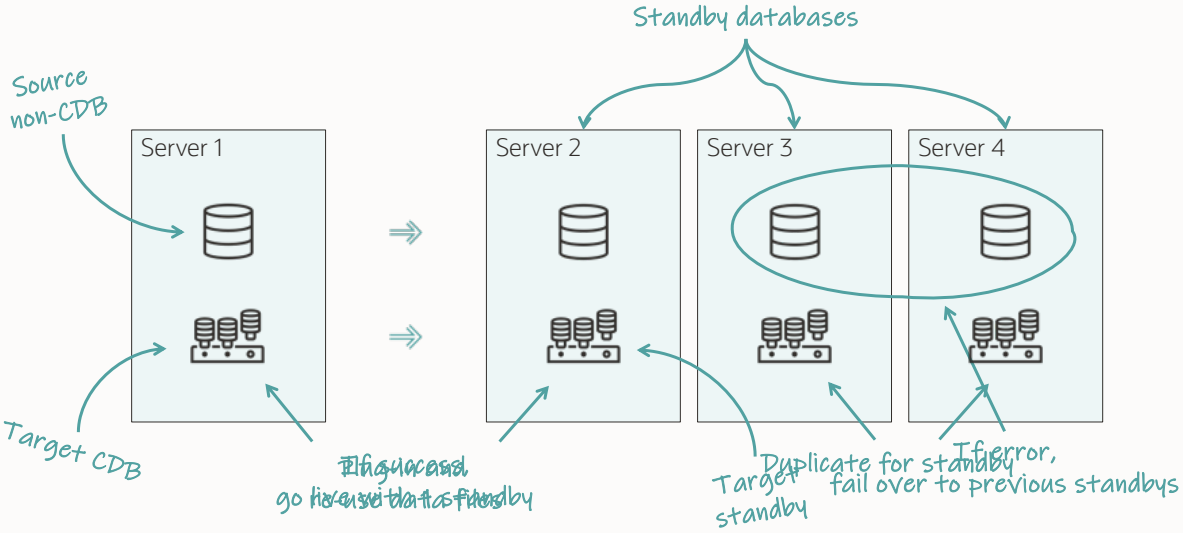
2 RMAN Restore

Rollback Options | Before Go-Live

1 Leave a copy

- `CREATE PLUGGABLE DATABASE ... COPY`
- Refreshable clone PDBs
- Incrementally roll forward data files copies
- Leave a standby database behind

Rollback Options | Before Go-Live



Rollback Options | Before Go-Live

2 RMAN Restore

- Time-consuming
- May not satisfy business requirements

Fallback Options | After Go-Live

1 Back to non-CDB

- Data Pump
- Transportable Tablespaces
- GoldenGate

An alternative option to fall back from upgrade and PDB conversion

Fallback Options | After Go-Live

If you upgraded and converted

- From Oracle Database 19c to 23ai

1 Back to 19c, stay multitenant

- Downgrade
- `COMPATIBLE` must be *19.0.0* in 23ai CDB

2 Back to 19c, back to non-CDB

- Follow 1
- Transportable tablespace back into non-CDB
- Alternatively, Data Pump from 23ai directly to 19c non-CDB

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
```

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

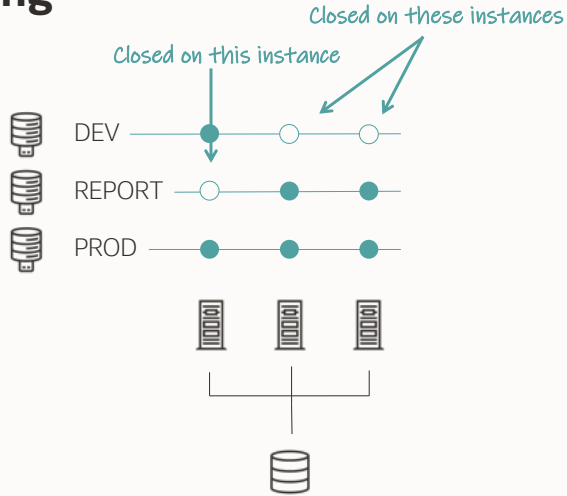
- Discard saved state and rely on Oracle Clusterware

```
--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

Method 2

Memory allocation

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

Method 2

PDB may never use more than 4G of shared memory

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:

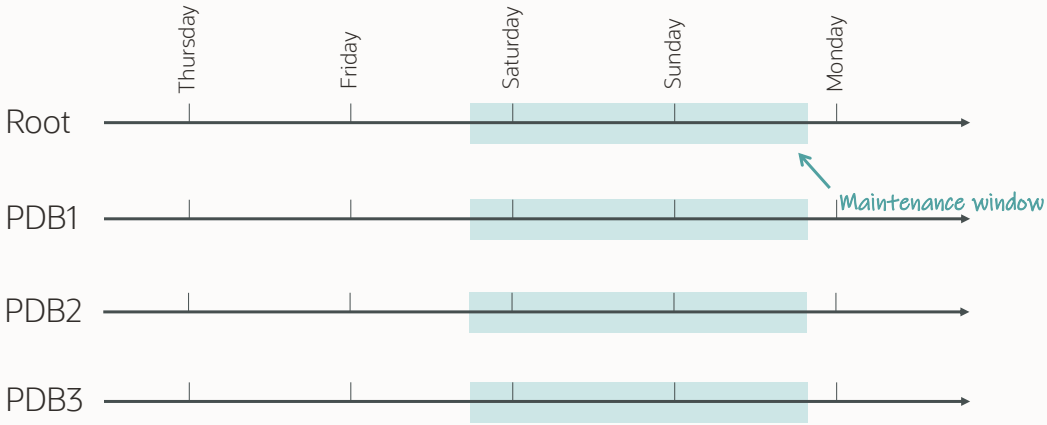
- `CPU_COUNT`
- `SGA_MAX_SIZE`

Inter-instance CPU resource manager:

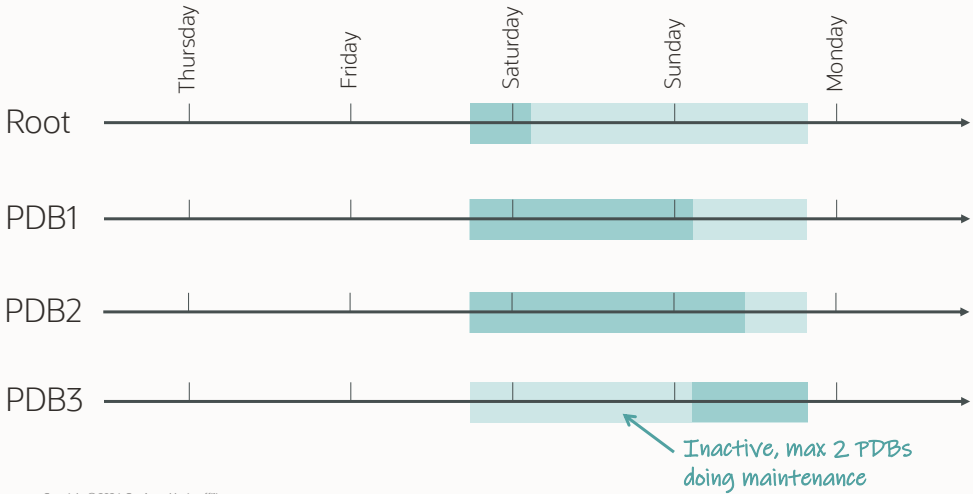
- Controls resource sharing using Linux c-groups
- Check `RESOURCE_MANAGER_CPU_SCOPE`
- Exadata Database Machine and Autonomous Database

A word about automated maintenance tasks

Automated Maintenance Tasks



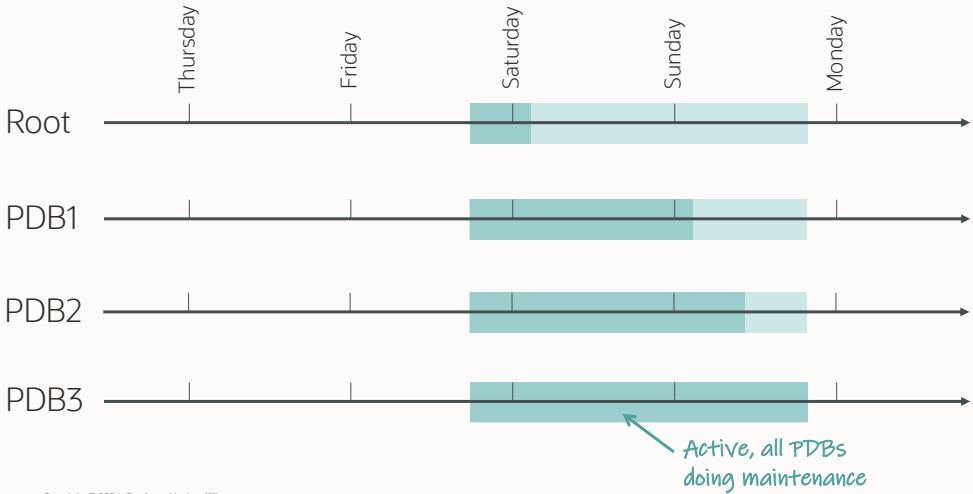
Automated Maintenance Tasks



```
--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;
```

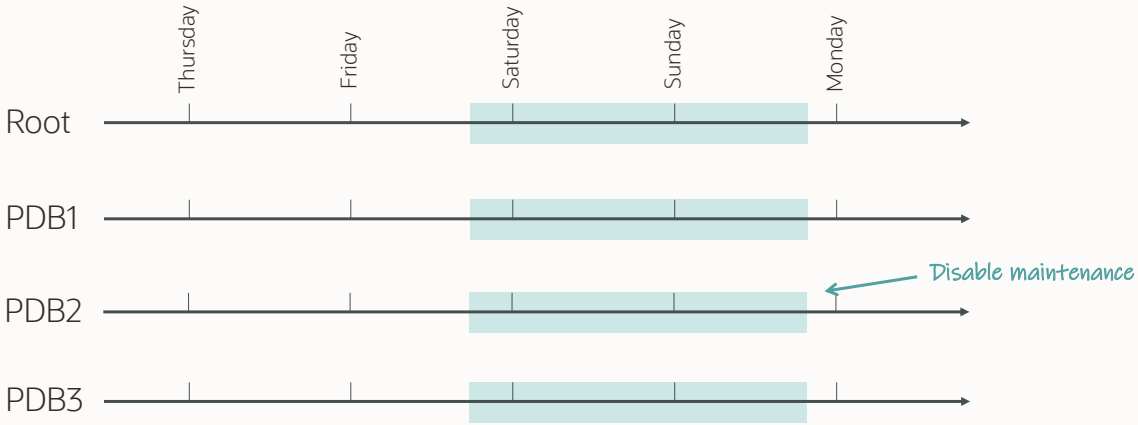

Automated Maintenance Tasks



```
--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;
```

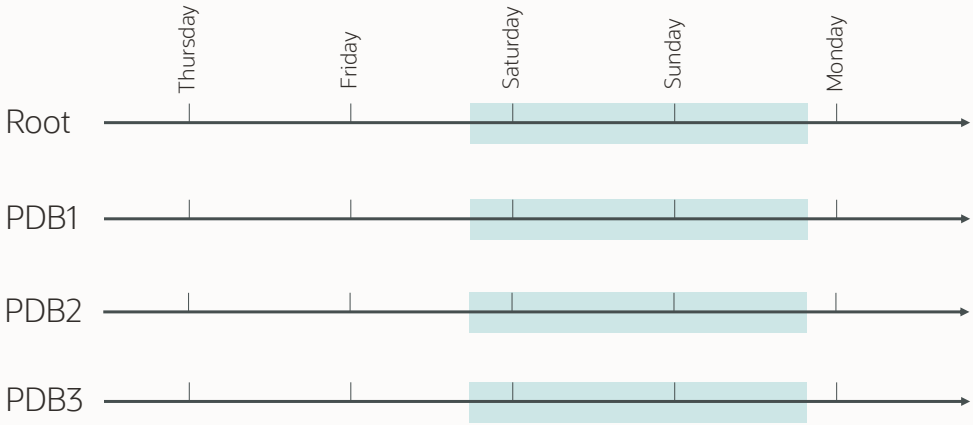

Automated Maintenance Tasks



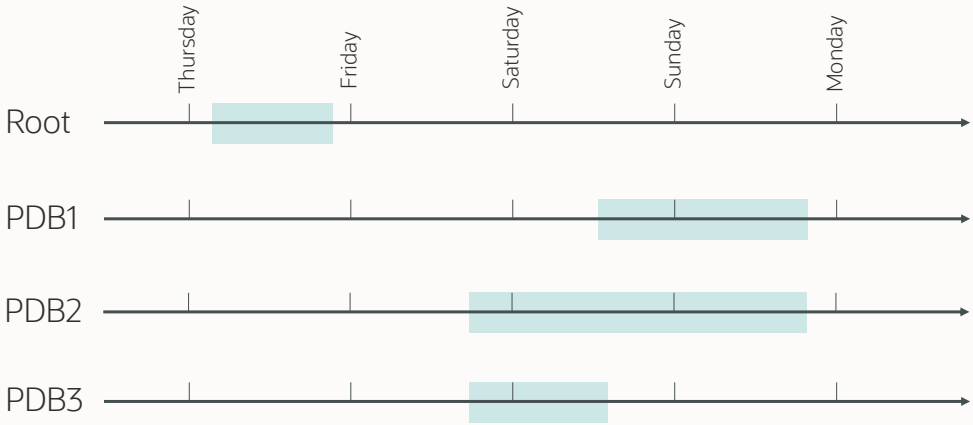
Shift maintenance windows

- 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

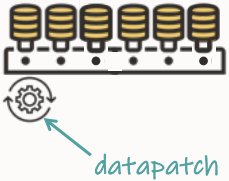
7 rows selected.
```

Patching

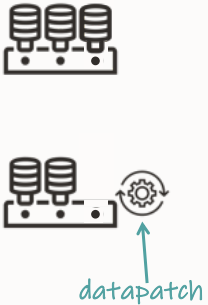
Patching Oracle home in a multitenant is the same as for non-CDB

Multitenant Patching Approaches

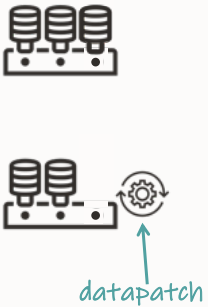
All at once



Unplug-plug



Refreshable clone



The database must be open
Only open PDBs will be patched

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

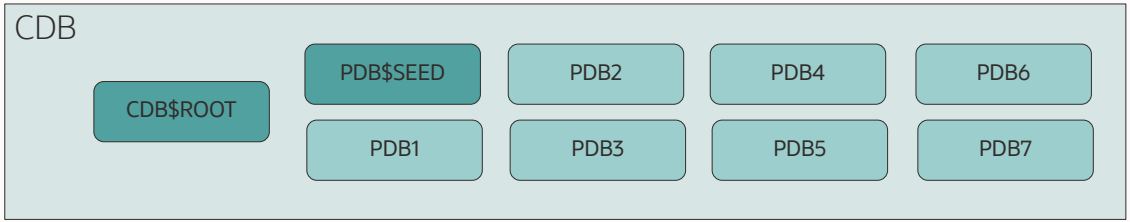
```
$ORACLE_BASE/cfgtoollogs/sqlpatch/.../sqlpatch_invocation.log
```

```
[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

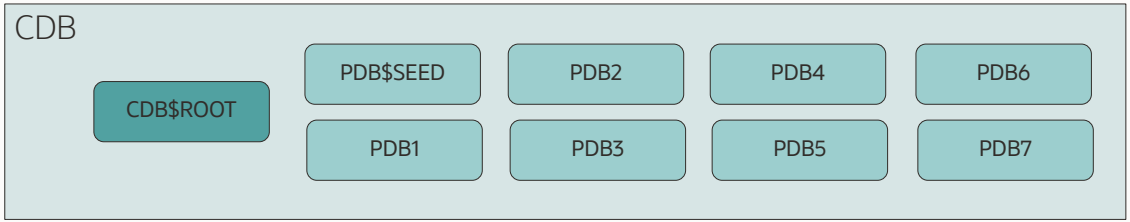
- Consider increasing the **PROCESSES** parameter

Datapatch



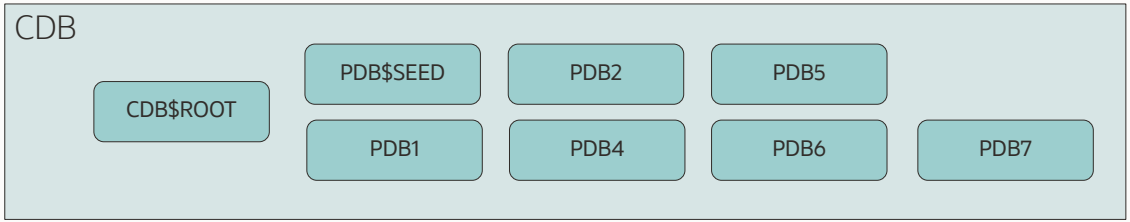
- Datapatch patches CDB$ROOT and PDB$SEED automatically

Datapatch



- Datapatch always patches CDB$ROOT first

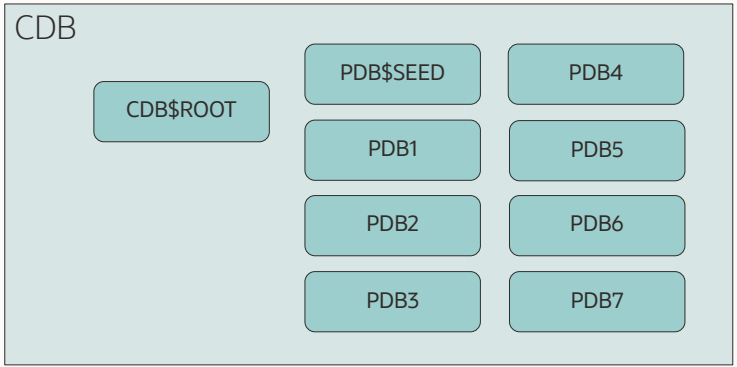
Datapatch



- Datapatch only patches open PDBs

PDB3

Datapatch

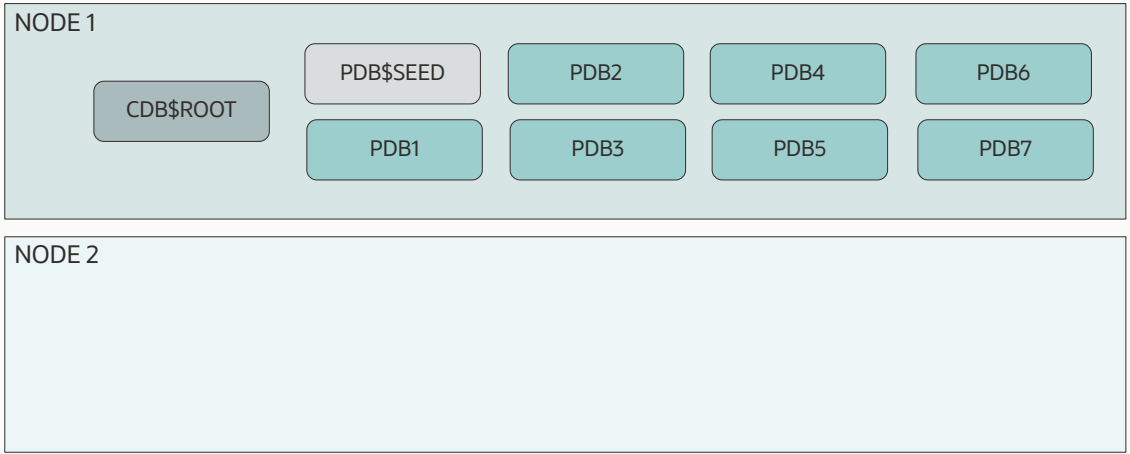


- Datapatch determines parallel degree based on CPU count

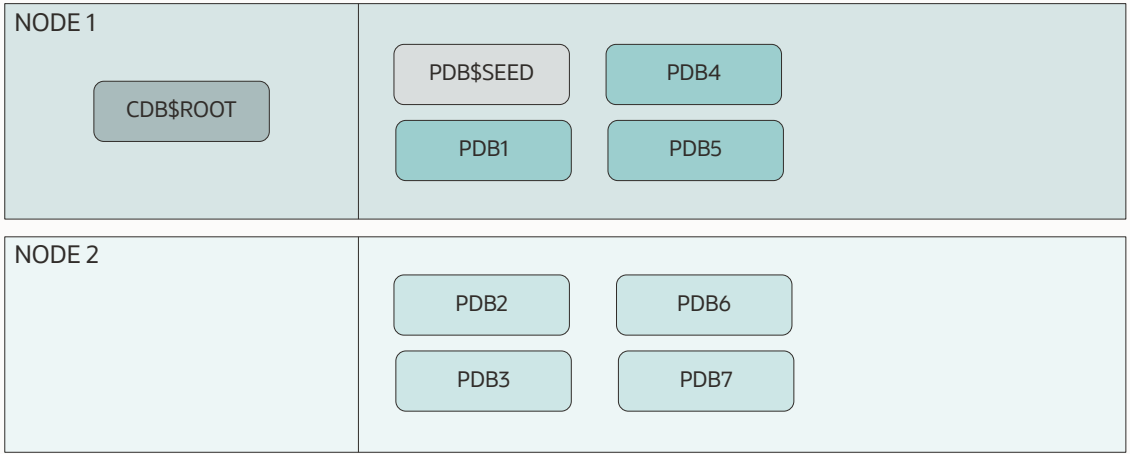
Significantly speed up patching using AutoUpgrade

- 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
```

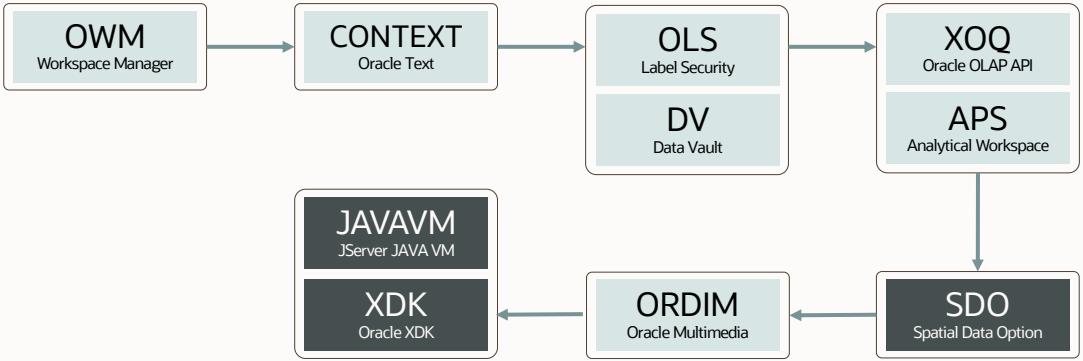
Less components, faster patching

Typical candidates:

- JAVAVM
- ORDIM
- SDO

Component Clean Up Order

If required, remove components before upgrade/plugin



Different Components in PDBs

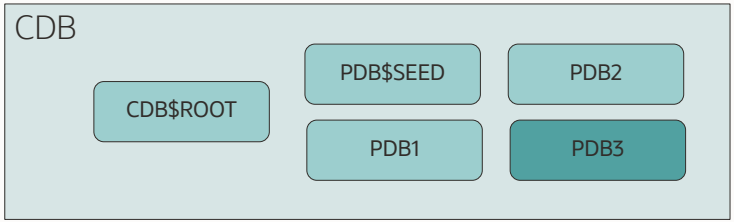
Having different components doesn't increase patching time

- 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

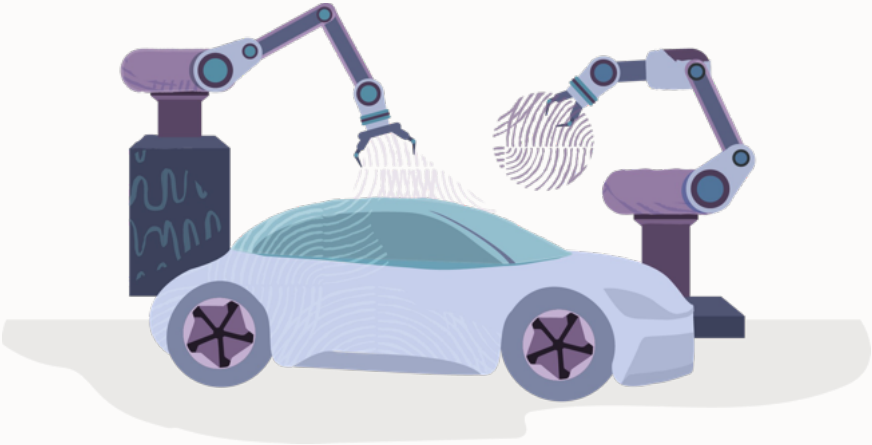
SQL> select name, open_mode, restricted from v$pdbs;

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



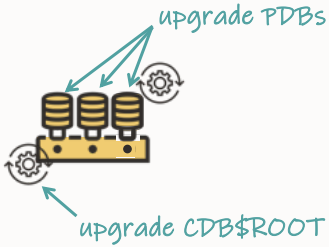
Multitenant Upgrade Approaches

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

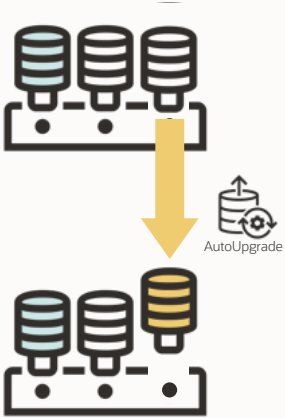
```
upg1.source_home=/u01/app/oracle/prod/19
upg1.target_home=/u01/app/oracle/prod/23
upg1.sid=CDB19
upg1.target_cdb=CDB23
upg1.pdbs=PDB2,PDB3
```

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

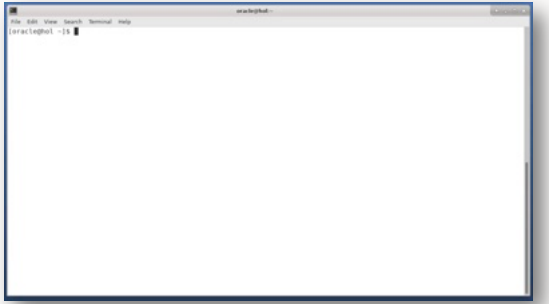
- Minimum 1
- Maximum 8
- Default 2
- You can override the default with:

```
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

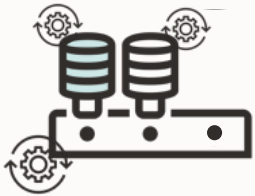
Parallelism calculation

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

- 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

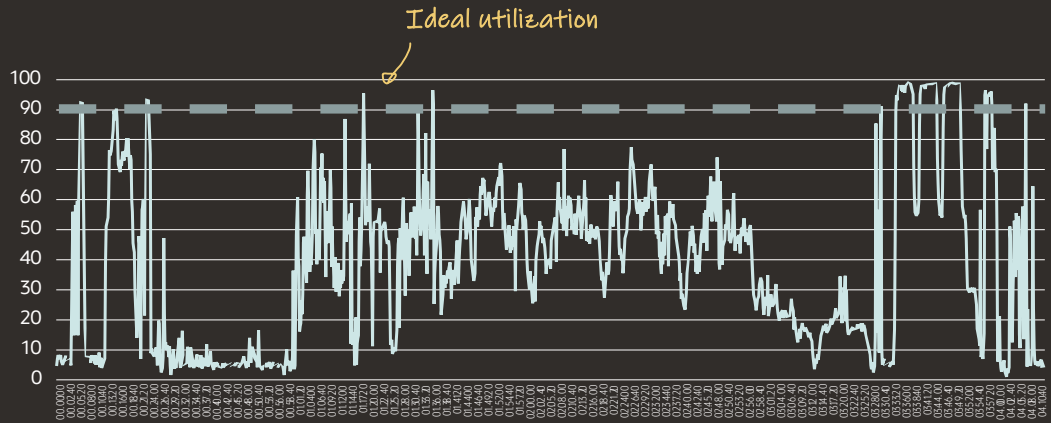
Oracle Database 12.1.0.2 to Oracle Database 19c

16 OPCUs
768 GB memory

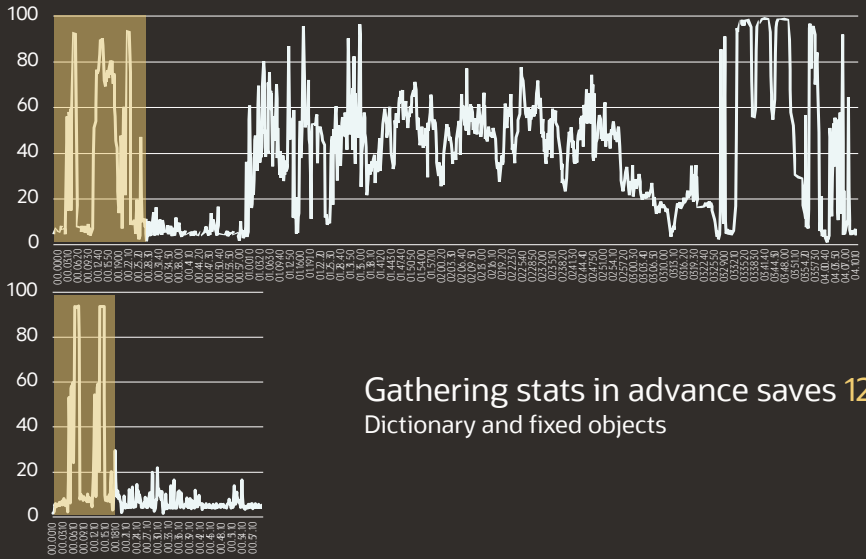
CDB with 52 PDBs
Many database components (17 in total)

`CPU_COUNT`	32
`SGA_TARGET`	80G
`PGA_AGGREGATE_TARGET`	20G

Upgrade Benchmark



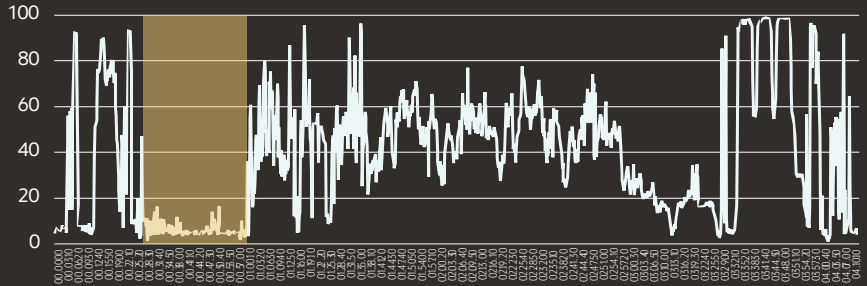
Upgrade Benchmark



Gathering stats in advance saves 12 minutes

Dictionary and fixed objects

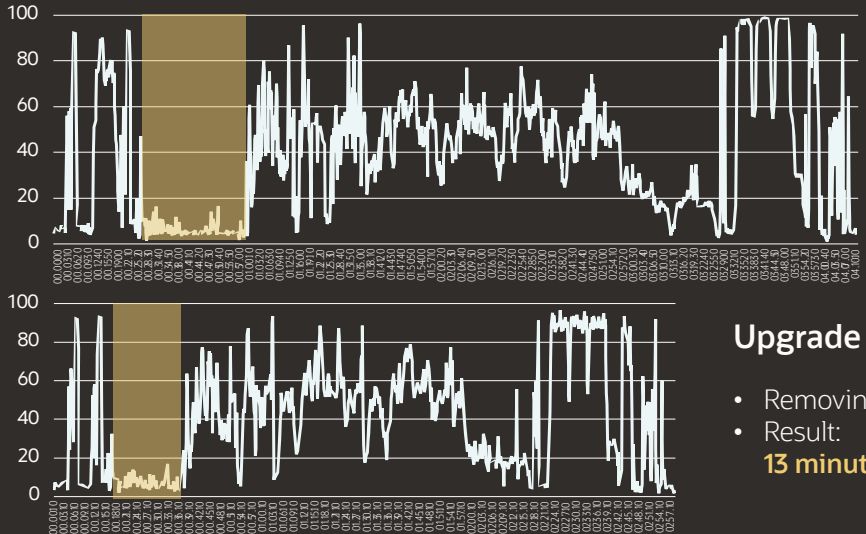
Upgrade Benchmark



Upgrade CDB$ROOT

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

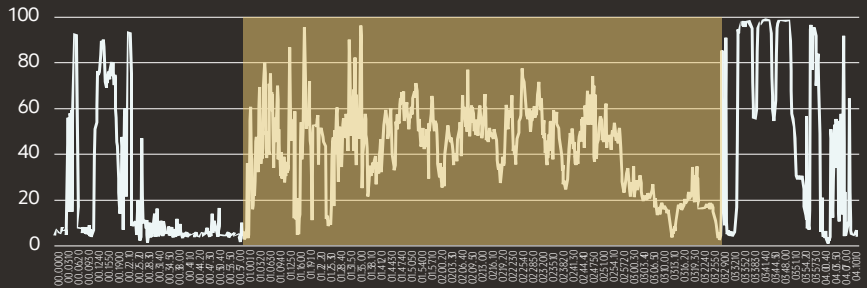
Upgrade Benchmark



Upgrade CDB$ROOT

- Removing all components
- Result:
  13 minutes faster

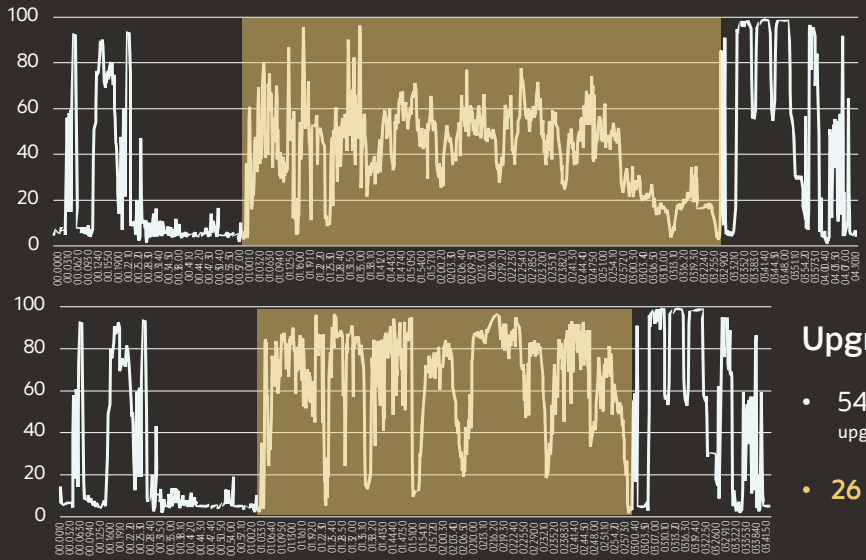
Upgrade Benchmark



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

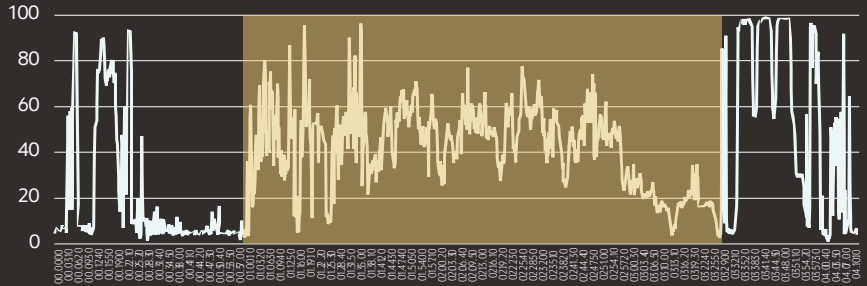
Upgrade Benchmark



Upgrade PDBs

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

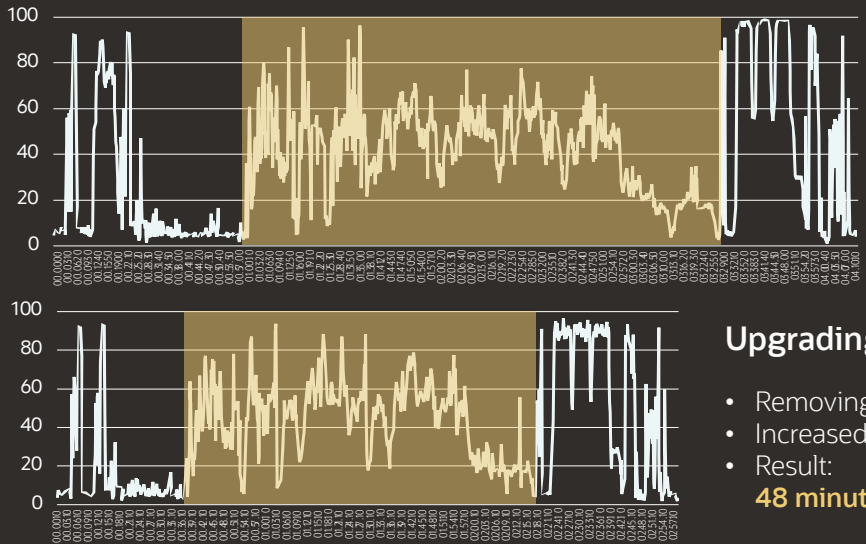
Upgrade Benchmark



Upgrading CDB$ROOT, PDB$SEED and 52 PDBs

- Speed up the upgrade? Consider removal of unused components

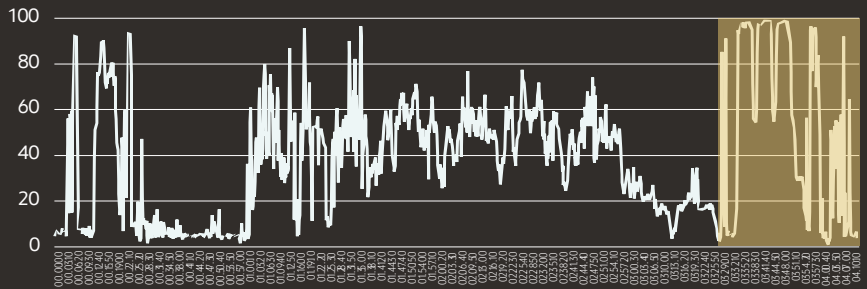
Upgrade Benchmark



Upgrading all containers

- Removing all components
- 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

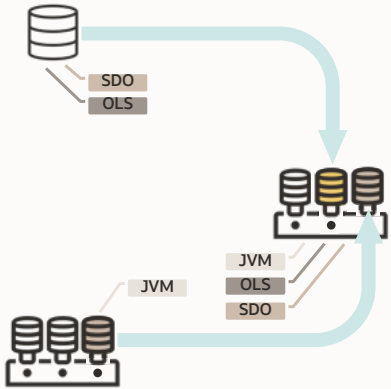
Increase number of PDBs upgraded in parallel

10% improvement

Implement all recommendations

32% improvement

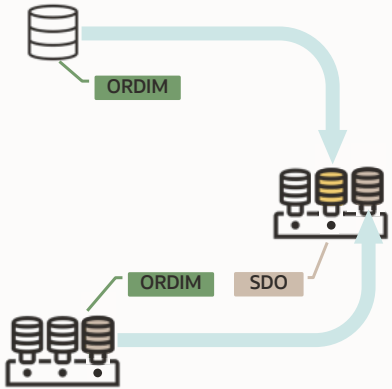
Component Considerations



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

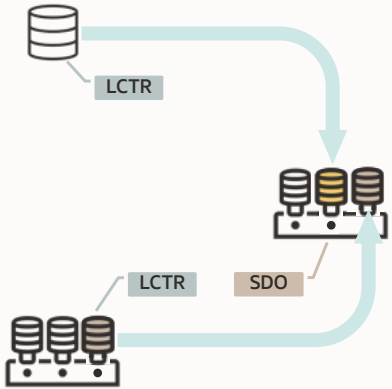
Special Case: Multimedia



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

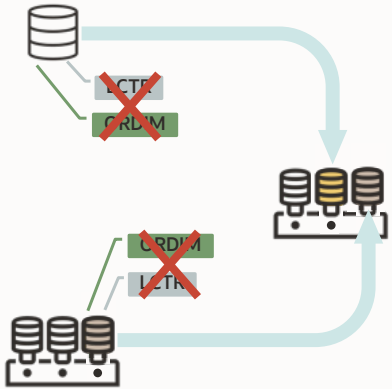
Special Case: Locator



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%';

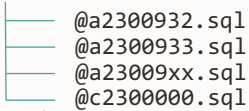
PROPERTY_NAME                  PROPERTY_VALUE
------------------------------ --------------------
CONVERT_NONCDB_ON_OPEN         true
UPGRADE_PDB_ON_OPEN            true
```


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



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

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

Before upgrade

- Asses the upgrade readiness and perform pre-upgrade fixups

```
export ORACLE_SID=CDB1
export ORACLE_HOME=/u01/app/oracle/product/19
export ORACLE_TARGET_HOME=/u01/app/oracle/product/23
java -jar autoupgrade.jar -config_values "pdbs=PDB1" -mode analyze

java -jar autoupgrade.jar -config_values "pdbs=PDB1" -mode fixups
```

Replay Upgrade

After upgrade

- Call Datapatch

```
$ORACLE_HOME/OPatch/datapatch -pdbs PDB1 -verbose
```

- Call AutoUpgrade

```
export ORACLE_SID=CDB23
export ORACLE_HOME=/u01/app/oracle/product/23
java -jar autoupgrade.jar \
             -preupgrade "dir=/tmp/aulog,inclusion_list=PDB1" \
             -mode postfixups
```

```
SQL> alter pluggable database pdb1 open;
alter pluggable database pdb1 open
*
ERROR at line 1:
ORA-60510: encountered an error during Replay Upgrade
```


Replay Upgrade – on failure?

If Replay Upgrade fails

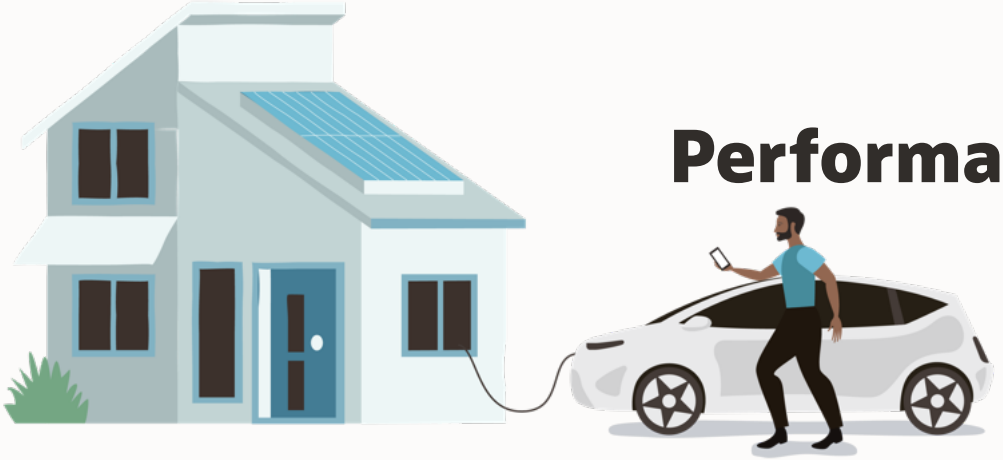
- Check for errors:
  - `SELECT * FROM DBA_REPLAY_UPGRADE_ERRORS`
  - `SELECT * FROM DBA_APP_ERRORS`
  - ```
    SELECT app_version, app_status
    FROM   DBA_APPLICATIONS
    WHERE  app_name='APP$CDB$CATALOG';
    ```
  - Check alert log
  - Trace files
- Revert to traditional upgrade
    ```

```
--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

- *prefix*.tune_setting=proactive_fixups=true

Proactive Fixups?

Performance feature

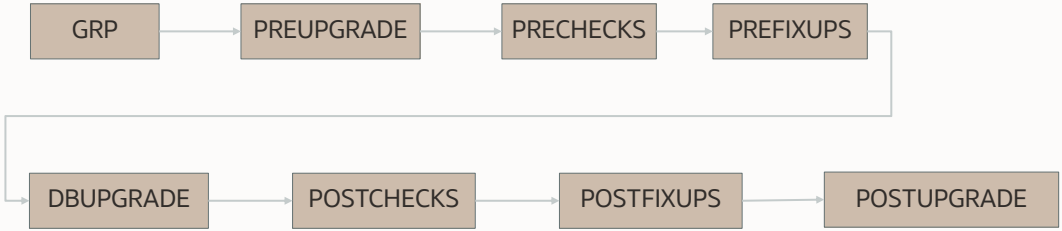
Start PDB post-upgrade tasks as soon as a PDB has been upgraded

- Independently of other PDBs

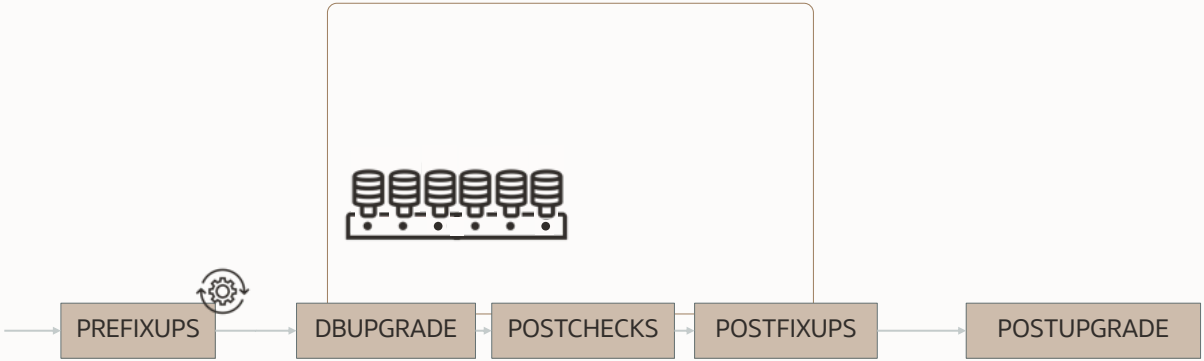
Isolates errors in PDBs

Valid for CDB upgrades only

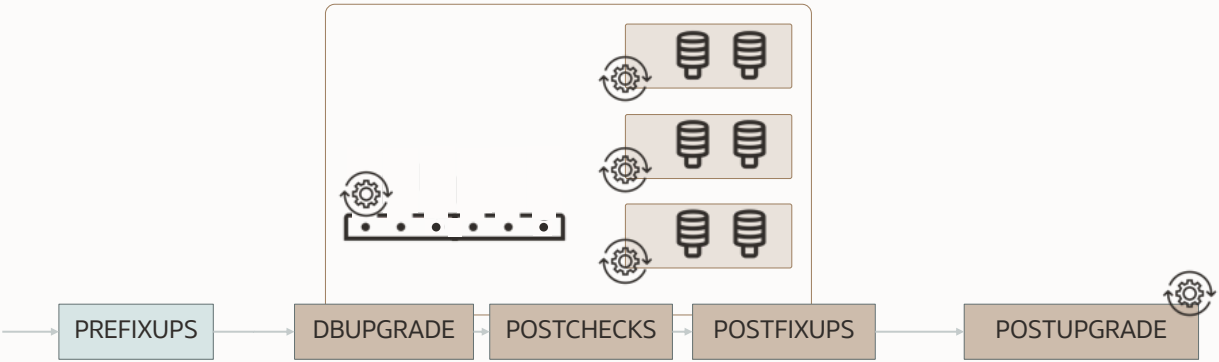
Classic Upgrade Flow



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

INFO	PREUPGRADE	<1 min
INFO	PRECHECKS	1 min
INFO	PREFIXUPS	8 min
INFO	DRAIN	<1 min
INFO	DBUPGRADE	143 min
INFO	POSTCHECKS	2 min
INFO	POSTFIXUPS	34 min
INFO	POSTUPGRADE	1 min
	TOTAL	**179 min**

Proactive Fixups

INFO	PREUPGRADE	<1 min
INFO	PRECHECKS	1 min
INFO	PREFIXUPS	7 min
INFO	DRAIN	<1 min
INFO	DBUPGRADE	130 min
INFO	POSTCHECKS	<1 min
INFO	POSTFIXUPS	<1 min
INFO	POSTUPGRADE	1 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

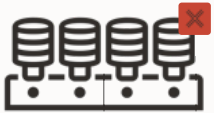
The more PDBs, the greater the benefit

Proactive Fixups isolate each PDB

- Errors in any given PDB don't effect others

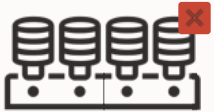
PDB Isolation

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.

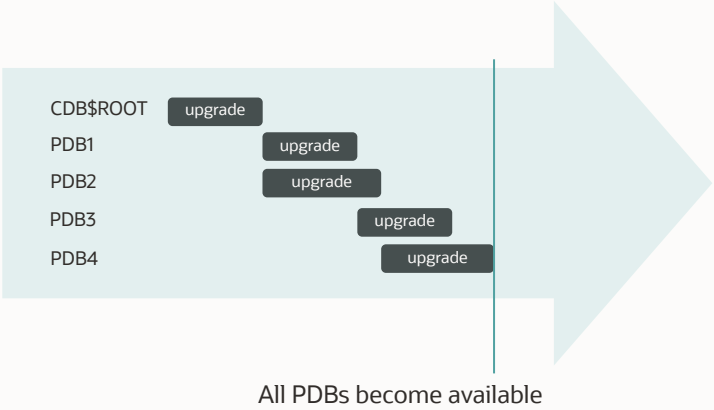
Some PDBs are more important

Control the order of the upgrade

Proactive Fixups Availability

DEFAULT

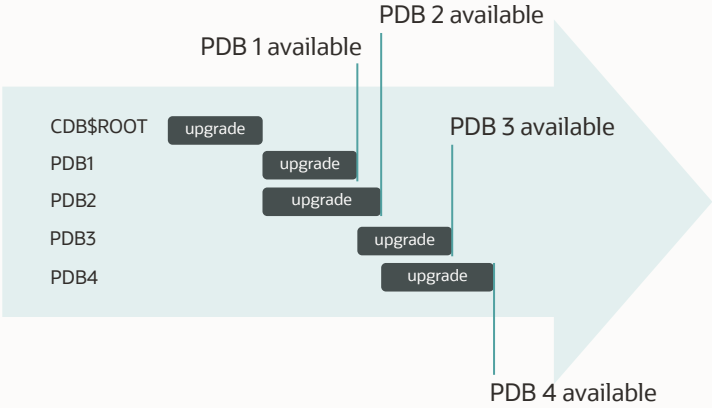
prefix.make_pdbs_available=false



Proactive Fixups Availability

IMMEDIATELY AVAILABLE

prefix.make_pdbs_available=true



Control the order of PDBs

- Make sure to specify all PDBs – otherwise, they'll be skipped, e.g.
  `*prefix*.pdbs=PDB1,PDB2,PDB3,PDB4`

```
alter pluggable database SALESPROD priority 1;

alter pluggable database SALESDEV priority 2;

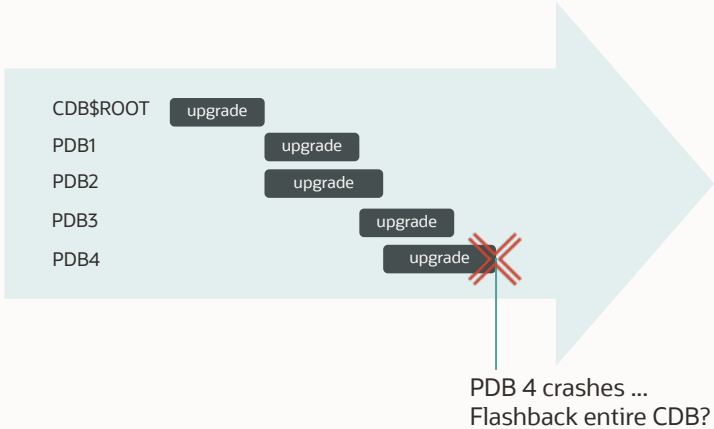
alter pluggable database SALESUAT priority 2;

alter pluggable database SALESTEST priority 3;
```


PDB Availability

IMMEDIATELY AVAILABLE

prefix.make_pdbs_available=true



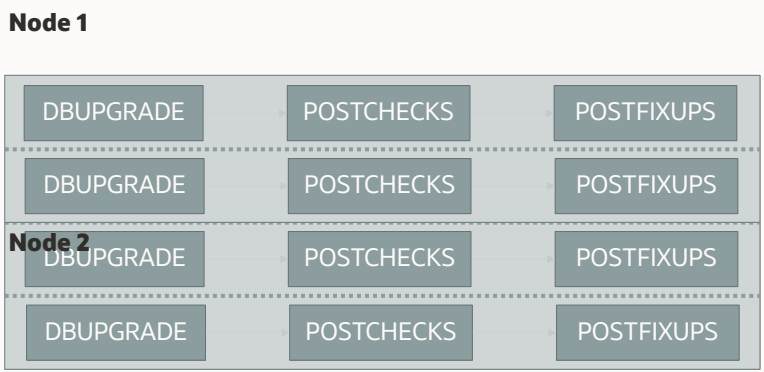
Distributed Upgrade

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

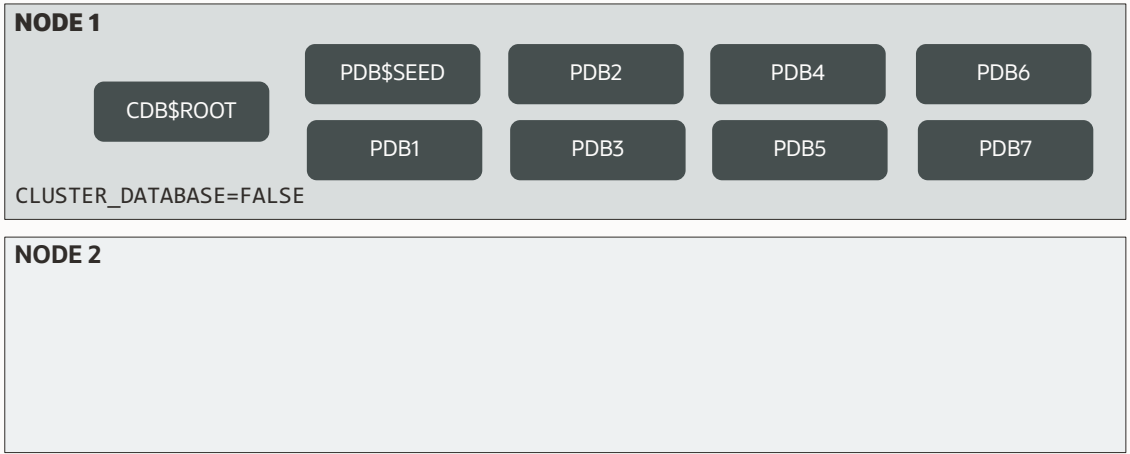
Distributed Upgrade Concept



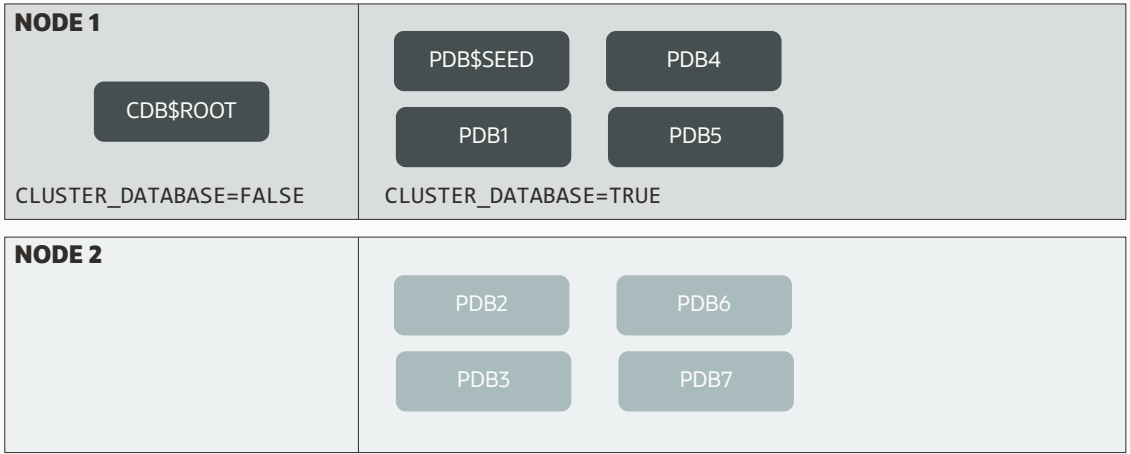
How does Distributed Upgrade work?

- 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 [redacted] as by default the time zone scripts won't touch the [redacted] when you execute them. And in addition, MOS Note:1509653.1 tells you, that the [redacted] 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 Mitropoulos 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

Words of Advice

1. Start with simple databases
2. Leave time to learn and adjust
3. Proceed with bigger databases

Further Reading

Oracle Support:

- Oracle Multitenant: Frequently Asked Questions (Doc ID 1511619.1)
- How to migrate a non pluggable database that uses TDE to pluggable database ? (Doc ID 1678525.1)

Blog posts:

- Database Migration from non-CDB to PDB – Typical Plugin Issues and Workarounds
- Upgrade & Plug In: With ASM, Data Guard, TDE and no Keystore Password

There are many benefits to explore

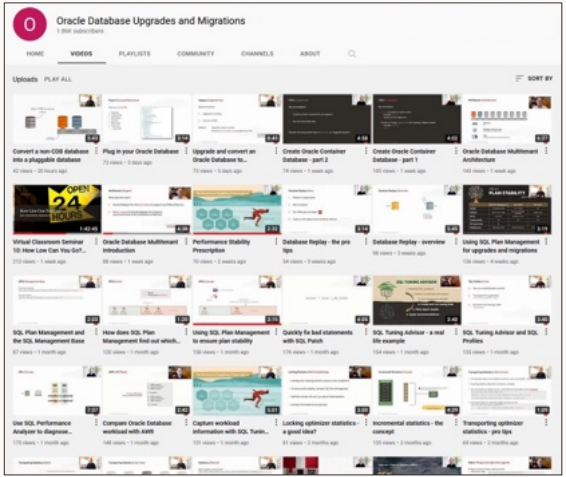
- Application containers
- Faster cloning
- Faster provisioning
- Faster redeployment
- Sparse clones
- Resource consolidation
- Save resources
- Improved functionality
- Better management
- More secure
- Separation of duties
- Easier operations
- Many-as-one
- Faster updates
- Faster patching
- Enables self-service

It's better to fail in our lab, than in production

Try multitenant migration in our Hands-On Lab

For free using Oracle LiveLabs

YouTube | @UpgradeNow



Link

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

SCAN ME

Thank You