Oracle RAC, Oracle Data Guard, and Pluggable Databases: When MAA Meets Oracle Multitenant

Ludovico Caldara, Trivadis AG

This whitepaper describes how Pluggable Databases work in a Maximum Availability Architecture and how to deal with PDB creation and services. It is targeted at database administrators and architects who want to know more about Oracle Multitenant in HA environments.
Text

Executive Summary

After reading this white-paper, you should know:

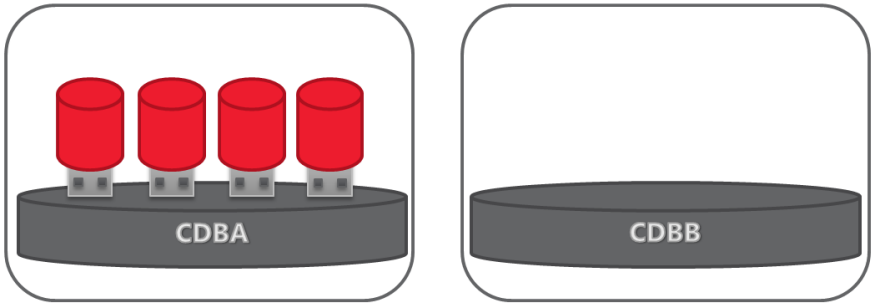
- about one of the most complex and effective solutions for database consolidation
- how to recognize the benefits of Oracle RAC in association with Multitenant
- how pluggable databases (PDBs) react in a Data Guard environment
- how to identify the main limitations and strengths of the whole architecture

RAC and Multitenant

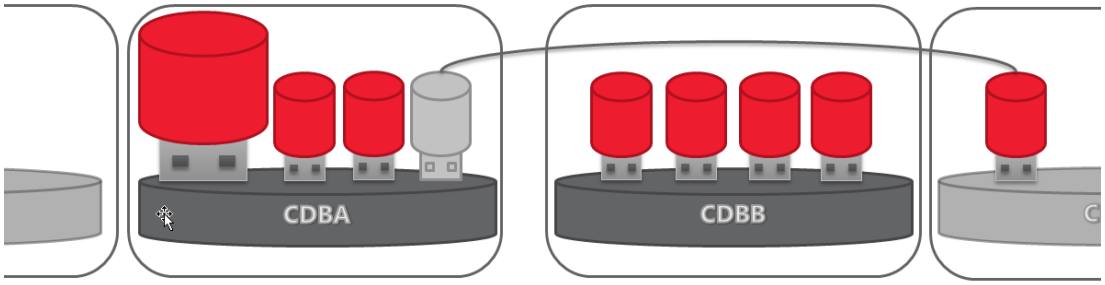
Why and How

Oracle has released the Oracle Multitenant option upon the release of Oracle Database 12c. Since the beginning, Oracle has marketed this option as “the” solution for consolidation and foundation for private cloud solutions.

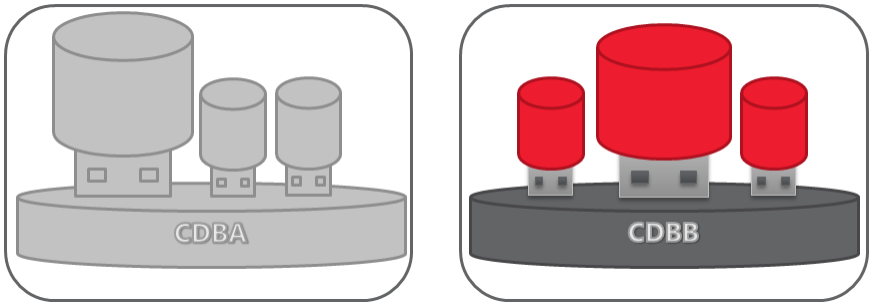
However, consolidating many pluggable databases on standalone instances has some limitations. First of all, the scalability of the server: once the Oracle CDB Instance fills up the entire host resources, customers need to continue the consolidation on newly created CDBs, on different servers:



This kind of consolidation may soon lead to CDB over provisioning, forcing customers to move PDBs to different CDBs and to react to increasing resource consumption.



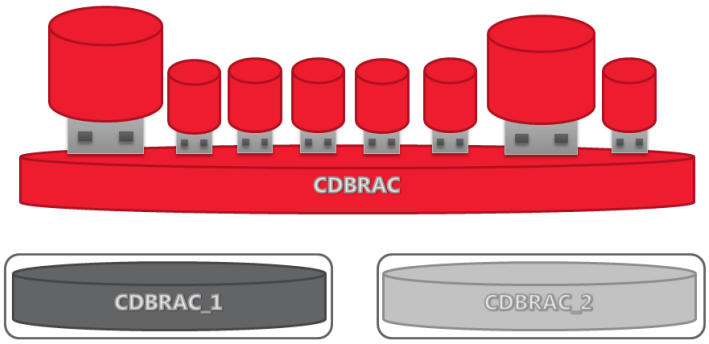
Moreover, there’s a problem of availability: when customers need to change static parameters, or need to patch or maintain the software or hardware, having many PDBs consolidated on one standalone CDB can make downtime harder to schedule, because different applications may need different maintenance windows.



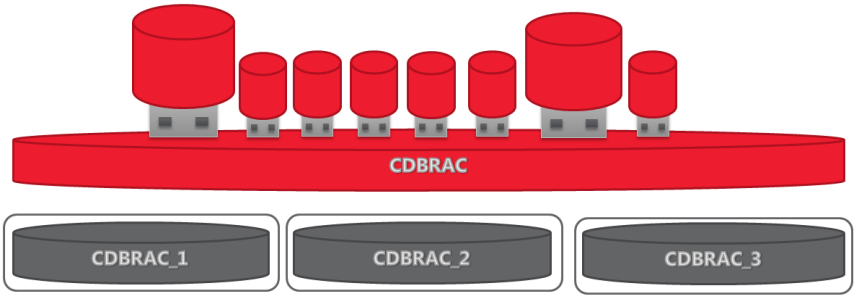
Briefly said, implementing Multitenant on standalone instances does not resolve the silo paradigm, it just move the problem at the CDB level rather than at the database level.

Having Oracle Real Application Clusters as backend solution can make the difference, because it provides both availability and scalability.

At availability level, the CDB runs on several instances: one instance failure does not affect the applications.



At scalability level, it's possible to add new instances to accommodate more PDBs as soon as new resources are needed.



The many PDBs don't compete for resources, because each PDB may be opened selectively on just a subset of instances, making possible to spread the resource consumption over all the instances. Which instances are opened on which server, it depends on the services defined at the cluster level.

Dealing with PDBs and Services

The pluggable databases, once created, are *mounted* by default. In the following example, a pluggable database named MAAZ is mounted on both instances in a two-node RAC CDB.

```
SYS@CDBATL_2> select INST_ID, CON_ID, name, OPEN_MODE
 2  from gv$pdbs where con_id!=2 order by name, inst_id;

   INST_ID     CON_ID NAME                           OPEN_MODE
---------- ---------- ------------------------------ ----------
         1          3 MAAZ                           MOUNTED
         2          3 MAAZ                           MOUNTED
```

But when a service exists for the PDB and it's started, the start of the service will open the PDB:

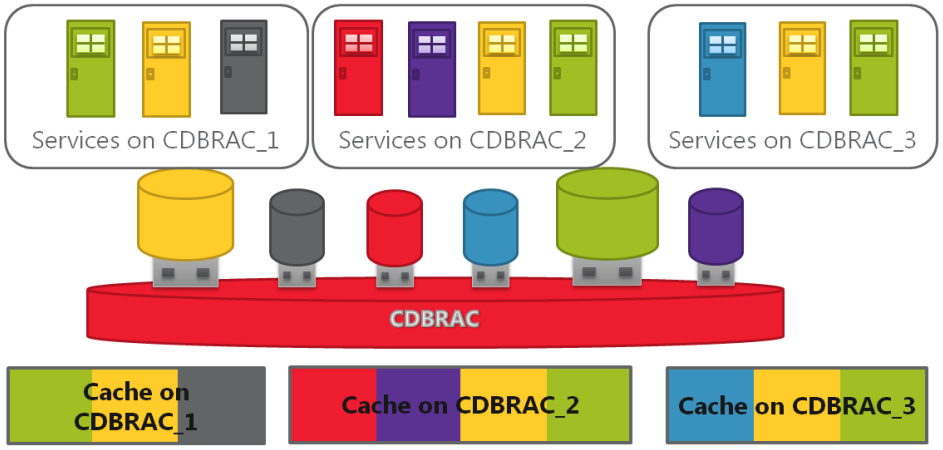
```
$ srvctl add service -db CDBATL -service maazapp -serverpool CDBPOOL
-cardinality singleton -role primary -failovertype select -
failovermethod basic -policy automatic -failoverdelay 2 -
failoverretry 180 -pdb maaz

$ srvctl start service -db CDBATL -service maazapp -instance
CDBATL_1
```

We can see that the srvctl commands have a new switch "-pdb" that let specify to which PDB we want to assign the newly created service. In this case the service has been defined as singleton: the service runs on one instance only, and by consequence the PDB is opened on that instance only:

```
   INST_ID      CON_ID NAME                           OPEN_MODE
---------- ---------- ------------------------------ ----------
         1          3 MAAZ                           READ WRITE
         2          3 MAAZ                           MOUNTED
```

By opposite, a stop of the service on an instance will not close the PDB: all sessions stay connected even if the service is stopped. The DBAs need to close the PDB manually if he wants to clean up the instance.

It's possible to open PDBs selectively on a specific number of instances by setting different services with different cardinalities. This permits a complete separation of resources and a much higher scalability coupled with multitenant, because every instance has its own redo thread, undo tablespace and especially its own buffer cache: one instance buffer cache will be populated only with blocks belonging to the PDBs active on that instance.



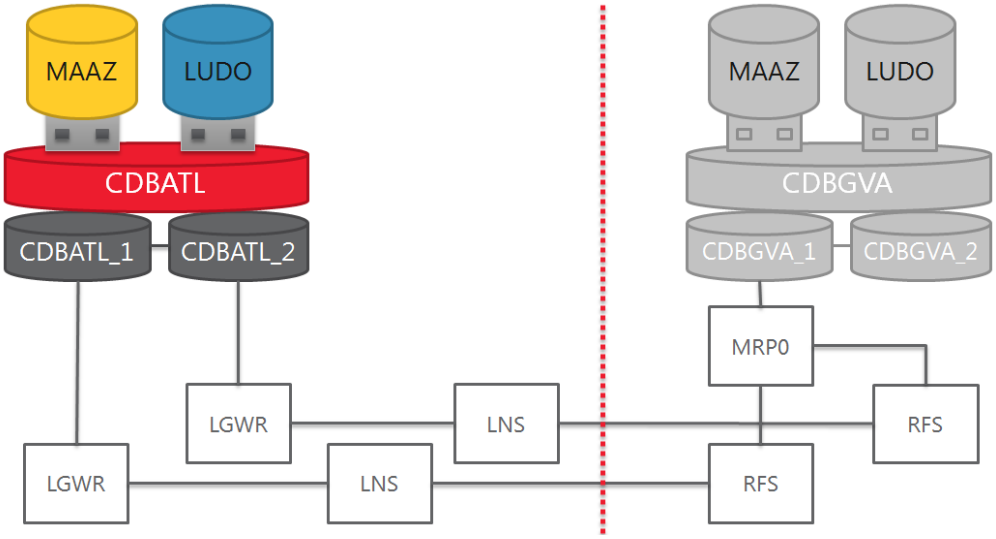
Thanks to this separation of resources, it becomes evident that RAC is an important technology enabler that not only provides higher availability to a Multitenant architecture, but also boosts the scalability: in case of resource starvation, it's possible to add one or more nodes to the cluster and rebalance the services/PDBs to the newly created instances.

The main limitation is the maximum number of services that can be specified: no more than 512 services can be created on a single CDB; after that a new CDB is required. The number of services becomes an important factor for a successful consolidation strategy in big environments.

Another important point is that if a node crashes, before the sessions can failover, the PDB need to be open: if the service is singleton and no other instances have the PDB open, the sessions will need to wait for a new instance to be ready. Opening critical PDBs on more than one instance can overcome this problem.

Why and How

Oracle Multitenant in an Oracle Data Guard environment brings an essential benefit: because the redo threads are shared amongst all PDBs, there's the need of just one stand-by CDB where all the PDBs are replicated. This implies the existence of only one Data Guard configuration and simplified maintenance, thus reducing enormously the administrative effort.

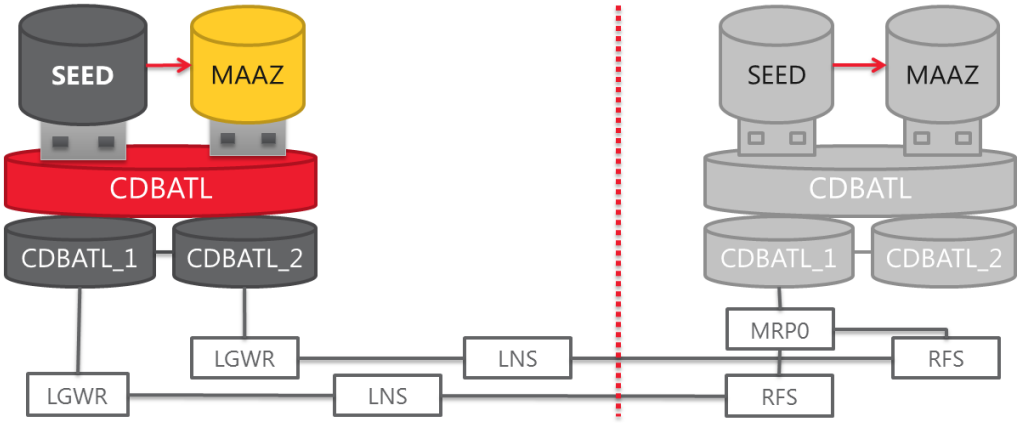


The side effect of this architecture is that the granularity of the switchover and failover operations is at CDB level, meaning that all the PDBs in a CDB will have the same role of their CDB: all primary or all stand-by. From a consolidation perspective is important to choose which PDBs need to be consolidated on which CDBs so that you can preserve some flexibility for the most critical PDBs.

PDB Creation and Cloning in a MAA environment

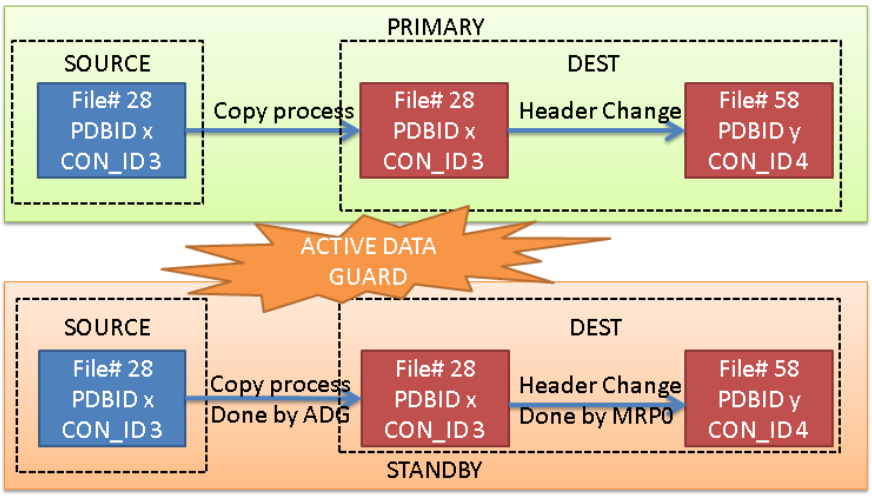
Upon the creation of a new pluggable database from PDB$SEED (the "standard" creation), the datafiles of the PDB$SEED database are copied and assigned to the newly created PDB. At the stand-by side, the Data Guard technology also copies the PDB$SEED datafiles from the local (stand-by) seed database, making the creation process transparent and allowing the recovery process to continue with the recovery of the new datafiles.

SQL> create pluggable database MAAZ;




```
Recovery created pluggable database MAAZ
Recovery copied files for tablespace SYSTEM
Recovery successfully copied file +DATA/CDBGVA/.../DATAFILE/system.435.856973955
  from +DATA/CDBGVA/.../DATAFILE/system.280.855055053
Successfully added datafile 24 to media recovery
```

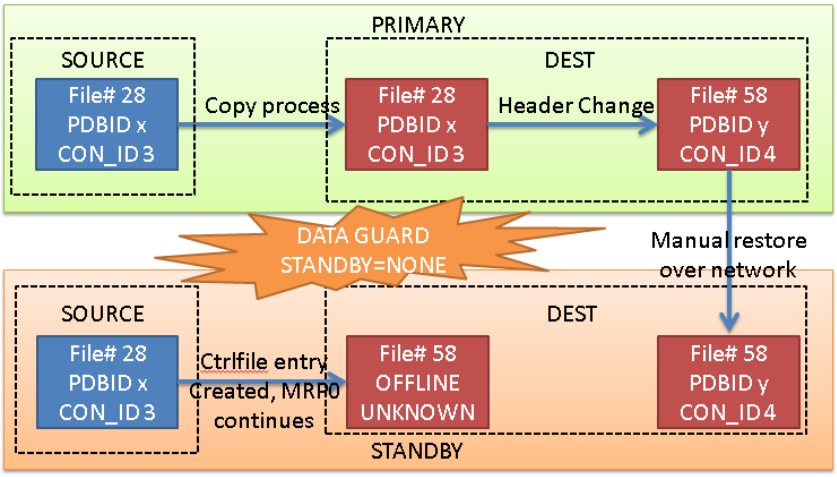
If the pluggable database is created by cloning an existent pluggable database (create pluggable database A from B;) the source database is copied on both sides (primary and stand-by) only if the stand-by database is open read-only, implying the need of Oracle Active Data Guard license.



If Active Data Guard is not licensed, the documentation says that it's necessary to copy the datafiles on the standby before the clone:

> *If you plan to create a PDB as a clone from a different PDB, then copy the data files that belong to the source PDB over to the standby database. (This step is not necessary in an Active Data Guard environment because the data files are copied automatically when the PDB is created on the standby database.)*
> http://docs.oracle.com/database/121/SBYDB/create_ps.htm#SBYDB5260

But in a RAC environment, the datafiles are managed by ASM and there's no way to copy the datafiles with the same path specified in the controlfiles. A manual restore of the new PDB is required on the standby CDB after the cloning occurs.



Oracle has published a MOS Note: Making Use of the STANDBYS=NONE Feature with Oracle Multitenant (Doc ID 1916648.1) that describes the best way to deal with pluggable database creation in a Data Guard environment, starting with release 12.1.0.2.

Dealing with PDBs and Services

In a Data Guard environment, the way to create new services doesn't change. If we create a read-only service on the stand-by database (Active Data Guard) we need to specify the correct role (physical_standby).

The only difference is that we need to specify the correct pluggable database with the –pdb switch:

```
$ srvctl add service -db db_unique_name  -service ro_service_name \
 -serverpool server_pool  -cardinality uniform -role
physical_standby \
 -failovertype select -failovermethod basic -policy automatic \
 -failoverdelay 2  -failoverretry 180 -pdb pluggable_database
```

Again, it's sensible to keep in mind that there's an overall limit of 512 services in a CDB. In an Active Data Guard + Multitenant environment we'll have, for each PDB:

- default service with the name of the PDB
- read-write service (more than recommended!)
- read-only service for the stand-by database

The maximum number of PDBs per CDB will then be 512/3=170 PDBs, way below the limit of 252 PDBs per CDB. The more services we plan to create per PDB, the less PDB we'll be able to consolidate in the same CDB.

The connection description also doesn't change from a traditional MAA connection string, except that the SERVICE_NAME must point to a service assigned to a specific PDB:

```
LUDOAPP =  (DESCRIPTION_LIST=
    (LOAD_BALANCE=off) (FAILOVER=on)
    (DESCRIPTION = (CONNECT_TIMEOUT=5)
      (TRANSPORT_CONNECT_TIMEOUT=3) (RETRY_COUNT=3)
      (ADDRESS_LIST=  (LOAD_BALANCE=on)
        (ADDRESS = (PROTOCOL = TCP)(HOST = raca-scan)(PORT = 1521)))
        (CONNECT_DATA = (SERVICE_NAME = LUDOAPP))     )
    (DESCRIPTION =   (CONNECT_TIMEOUT=5)
```

```
(TRANSPORT_CONNECT_TIMEOUT=3)  (RETRY_COUNT=3)
(ADDRESS_LIST=    (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST = racb-scan)(PORT = 1521)))
  (CONNECT_DATA = (SERVICE_NAME = LUDOAPP)) )  )
```

Always from a consolidation perspective, having a highly available OID for naming resolution is highly recommended.

My conclusion

Active Data Guard has become a more and more appealing option, but with the PDB cloning limitation on non-active standbys, it's not just a nice-to-have. ADG is the best solution to a concrete Data Guard+Multitenant limitation. A consolidation with Multitenant requires ideally all three options (Multitenant, Real Application Cluster, Active Data Guard) that are not easily affordable by customers, despite the multiple benefits that this solution provides.

From a consolidation perspective, the MAA+Multitenant architecture is the best choice in the direction of a cloud infrastructure because it has all the requirements in term of consolidation density, scalability, availability and ease of administration, so I would not hesitate to recommend it. Moreover, Oracle has recently announced the deprecation of the non-CDB architecture: a move that make customers considering the new architecture as a durable choice.