

Lesson 5

Backup and Recovery

Objectives

- Archive log mode
- Backup the CDB and PDBs
- Restore and recovery
 - Complete
 - Incomplete
- Database flashback
- Data Guard and Multitenant

2

Configuring for recoverability

3

- The redo log is managed in the root container
- Configure archive logging and flashback in the root
- RMAN is aware of the multitenant environment
- All operations can be carried out from the root
 - Backup of the entire CDB or of a container
 - Restore of the CDB or a container
 - Complete or PIT recovery of the CDB or a container

RMAN is fully multitenant aware, with syntax extensions to manage multiple containers. As a general rule, from the root container one can do anything; from within a PDB, one can perform backup and recovery operations for that PDB only.

RMAN syntax for Multitenant

4

- No difference when connected to a PDB
- Use of TWO_TASK (or LOCAL) may help
- When connect to the root container:
 - The PLUGGABLE keyword nominates a container:
`backup pluggable database "CDB$ROOT" ;`
`restore pluggable database pdba ;`
`recover pluggable database pddb ;`
- A container name can qualify a tablespace name:
`backup tablespace pdbc:users ;`

When connected to a pluggable container as SYSDBA or SYSBACKUP (always using password file authentication) all RMAN commands will run as normal. From the root container, the entire environment can be backed up or recovered.

The restore and recover of an individual container is possible by leveraging the Tablespace Point In Time Recovery (TSPITR) mechanism. This is because there is an absolute guarantee that the tablespaces of one container are self-contained.

Complete recovery

5

- Controlfile recovery
 - In the root container, in nomount mode
 - Conclude with RESETLOGS
- Online logfile member: clear, or drop and create
- Datafile recovery: the four steps
 - Offline the damaged file(s)
 - Restore
 - Recover
 - Bring online
- Perform from the root or within the PDB

The usual rules for recovery apply.

If the controlfile is damaged, this must be restored in NOMOUNT mode. Damage to the online log does not require any downtime (assuming that the members are multiplexed).

The critical datafiles (SYSTEM or UNDO datafiles) necessitate mount mode for restore and recover. This will be the entire CDB if the root container is damaged, or the individual PDB if the damage is restricted to one container. Damage to a non-critical datafile can be repaired while the container is open.

Incomplete recovery

6

- Perform at the CDB or PDB level
- The four steps
 - Mount the container
 - Restore all datafiles
 - Recover UNTIL
 - Open RESETLOGS
- Method for PDB recovery depends on undo mode
 - Shared undo uses an auxiliary database
 - Local undo can be done in place
 - Leverages the TSPITR mechanism



A CDB may be supporting many pluggable containers. It is therefore important that it should be possible to take one back in time without affecting the others. This can be accomplished with zero downtime for the bystander containers.

The process leverages the tablespace point in time recovery (TSPITR) mechanism. A requirement for TSPITR is the set of tablespaces to be recovered should be "self contained", with no dependencies on any objects in other tablespaces. This is guaranteed by the Multitenant architecture: each container is isolated from the others. The exception is the undo tablespace, in shared undo mode. RMAN will be aware of this, and requires an auxiliary instance. For example:

```
RMAN> ALTER PLUGGABLE DATABASE JW1 CLOSE IMMEDIATE;
RMAN> RUN {
SET UNTIL TIME = 'SYSDATE - 12/24';
RESTORE pluggable DATABASE JW1;
RECOVER pluggable DATABASE JW1
AUXILIARY DESTINATION='/u01/auxdb';
ALTER PLUGGABLE DATABASE JW1 OPEN RESETLOGS;
}
```

Database Flashback

7

- Enable flashback in the root as normal
- Flashback can be the entire CDB
- Restore points can be created at both levels
- Flashback of a PDB (12.2)

- Should use local undo
 - Perform from the root
- ```
connect / as sysdba
alter pluggable database pdba close immediate;
flashback pluggable database pdba to
[timestamp... | SCN... | restore point...] ;
```

In release 12.1, a flashback operation (though not point in time recovery) applied to the whole CDB. From 12.2 it is possible to flashback one container if local undo has been enabled. Without local undo, it is not possible because the undo tablespace, being shared, cannot be taken back in time.

## Duplication With RMAN

8

- Use normal syntax to duplicate an entire CDB
- When duplicating to a new CDB
  - Start the auxiliary with `enable_pluggable_database=true`
  - The root and seed containers are always included
  - A single PDB or a set of PDBs can be duplicated
  - Tablespaces can also be excluded or included
- When duplicating to a different CDB
  - Duplicate one PDB at a time
  - Only active database duplication is possible
  - Duplication to a standby is not possible
  - TDE may cause issues



To duplicate to a new CDB:

Start the auxiliary instance, with `enable_pluggable_database=true`

Ensure that backups are visible to the auxiliary

Example duplicate commands:

```
duplicate database to cdb2 pluggable database pdba;
duplicate database to cdb2 pluggable database pdba,pdbb;
duplicate database to cdb2 skip pluggable database pdbc;
```

To duplicate to an existing CDB, connect to the source as a target and the destination as an auxiliary and use a command such as

```
duplicate pluggable database pdba as pdbb to cdb2
db_file_name_convert ('cdb1','cdb2')
from active database;
```

Note that it is not possible to duplicate a PDB from a backup of one CDB into another CDB; only active duplication is possible.



## Refreshable clones

9

- Leverages the hot cloning capability
- Clone must be opened read only
- Refreshed by applying redo
  - Manually on demand
  - By a scheduled job
  - Clone must be in mount mode for the refresh to succeed
  - Can be frequent – every minute is possible
- The roles of the source and clone can be switched
- Possible substitute for Data Guard in Standard Edition



To create a clone with manual refresh:

```
create pluggable database pdbe from pdba@cdb2
 refresh mode manual;

alter pluggable database pdbe open read only;
alter pluggable database pdbe close immediate;
alter session set container=pdbe;
alter pluggable database refresh;
alter pluggable database open read only;
```

Or for automatic refresh:

```
create pluggable database pdbe from pdba@cdb2
 refresh mode every 60 minutes;
```

A job will be required to close the database or the refresh will not succeed. The refresh is accomplished by use of redo. Any archive logs necessary must therefore be available.

To perform a role reversal (Enterprise Edition on Exadata or ODA only):

```
alter pluggable database refresh mode manual from pdba@cdb2
 switchover;
```

## Data Guard And Multitenant

10

- Conceptually the same as non-CDB
  - Redo propagation and apply
  - Fast start failover
  - Transient logical rolling upgrade
  - Active Data Guard
- Configured and managed in the root container
- Role transitions are at the CDB level only
- Recovery can be disabled for a nominated PDB



The mechanisms of redo generation, propagation, and apply are not changed in a Multitenant environment. The configuration can be managed by the Data Guard Broker in the usual fashion.

All PDBs in the standby will have the same role and open mode. It is not, for example, possible to have some containers opened as snapshot standbys while others are mounted and applying redo. Similarly, the protection mode is an attribute of the entire configuration, not of individual containers. In an ADG environment, individual containers can however be open or closed.

It is possible to disable recovery for one or more PDBs. Stop recovery in the root container:

```
alter database recover managed standby database cancel;
```

And in the pluggable container, disable recovery:

```
alter pluggable database pdba disable recovery;
```

Then restart the recovery process. The recovery status of each container is shown in `v$pdb$recovery_status`.

## Creating PDBs in a Data Guard Configuration

11

- Datafiles must be available on the standby
- Create from seed in the primary
  - Standby will copy its seed files to create the new PDB
  - Or to prevent the creation in the standby(s):  
`create pluggable database ... standbys = all|none ... ;`
- Cloning within the CDB
  - Standby will copy its PDB files to create the new clone
  - The source must be enabled for Data Guard
- Disabled PDBs can be restored to the standby
  - Use RMAN to backup and restore
  - Enable recovery for the container



When creating PDBs from seed, because the standby does have a copy of the seed container the standby can use this to create the new PDB. When making a local clone, that too can be accomplished automatically by the standby.

If a PDB is create with STANDBYS=NONE or if recovery is disabled for a PDB, the logical structure is maintained in the standby but its datafiles are marked as UNNAMED and all change vectors for them are discarded.

To enable a previously disabled container:

Backup the container on the primary

Copy the backupset to the standby

Stop redo apply

Catalog the backupset into the standby controlfile

Restore the PDB, using SET NEWNAME for files if necessary

Move your session to the PDB, and

```
alter pluggable database pdba enable recovery;
```

Start redo apply

## Automating standby datafile copy with ADG

12

- Local clones are not a problem
  - The standby already has a copy of the source
  - The new PDB is created by copying the local datafiles
- Remote clones or plug-ins need the datafiles
  - The standby will not have a copy of the source PDB
  - Files can be transferred through a database link
  - Or copied from a nominated directory
- The source must be open read only
- Active Data Guard licences are not needed
  - ADG is a requirement for standby datafile automation
  - The licence is waived if only the root is opened



In earlier releases, when creating a PDB via plugin or remote hot cloning over a database link the files had to be made visible to the standby in advance. By default, Oracle would look for them in the standby's OMF destination. If they were not there, redo apply would cease.

This issue can be avoided by setting `STANDBY_PDB_SOURCE_FILE_DBLINK`. The parameter is set to the name of the database link that primary will use to clone the remote PDB or non-CDB. The standby must be open read only, in order for it to be able to get the link details from its data dictionary.

Open the standby's root container read only, and set the parameter to point to the link. Then it should be possible to perform a remote hot clone: the datafiles will be copied to both primary and standby over the database link.

An alternative is to copy the datafiles to a directory on the standby server, create an Oracle directory pointing to this location, and set `STANDBY_PDB_SOURCE_FILE_DIRECTORY` to the directory. The files will be found there, and copied into the OMF destination. Using NFS may be the best way to do this: the directory can then point straight to the source (PDB or non-CDB) OMF location. Then plug in to primary using the .XML manifest file, and the standby will copy the datafiles into its OMF location.

Whether the plugin is by hot clone or XML file, the operation is not "hot" because the source PDB or non-CDB must be read only throughout the whole operation.

## Backup and recovery: summary

13

- Perform from the root or within a PDB
- Within a PDB, RMAN is unchanged
- Granularity can be the CDB, PDB, or datafile
- Incomplete recovery of one PDB is possible
- Flashback of one PDB may be possible
- Data Guard is Multitenant aware
- RMAN duplications can be at PDB level

## Exercise

- Backup the CDB and PDBs
- Complete recovery of a PDB
- Incomplete recovery of a PDB
- Database flashback
- Configure and manage physical standby

14

# SKILLBUILDERS



ORACLE DATABASE ADMINISTRATION

CLOUD ADMINISTRATION

ORACLE AND APEX HOSTING

APEX APPLICATION DESIGN AND DEVELOPMENT

TRAINING