Blink and You'll Miss It: Migrating, Cloning and Recovering Oracle 12*c* Databases At Warp Speed

Jim Czuprynski
Zero Defect Computing, Inc.

My Credentials

ORACLE Certified Professional

ORACLE ACE Director

- 30+ years of database-centric IT experience
- Oracle DBA since 2001
- Oracle 9i, 10g, 11g OCP and Oracle ACE Director
- > 100 articles on databasejournal.com and ioug.org
- Teach core Oracle DBA courses (Grid + RAC, Exadata, Performance Tuning, Data Guard)
- Regular speaker at Oracle OpenWorld, IOUG COLLABORATE, OUG Norway, and Hotsos
- Oracle-centric blog (*Generally, It Depends*)

Our Agenda

- Fresher on Multi-Tenancy Databases: CDBs, PDBs, and PDB Migration Methods
- Cloning a New PDB from “Scratch”
- Cloning New PDB from Existing PDBs
- “Replugging” Existing PDBs
- Migrating a Non-CDB to a PDB
- RMAN Enhancements
- Q+A

Upgrading to 12*c*: What's the Rush?

- 12*c* Release 1 … is actually 12*c* Release 2
- 12.1.0.2 offers significant enhancements
  - PDB Enhancements
  - Big Table and Full Database Caching
  - In-Memory Aggregation
  - In-Memory Column Store

- Support for 11gR2 expires as of 12-2015
  - 11gR2 Database issue resolution costs will escalate dramatically

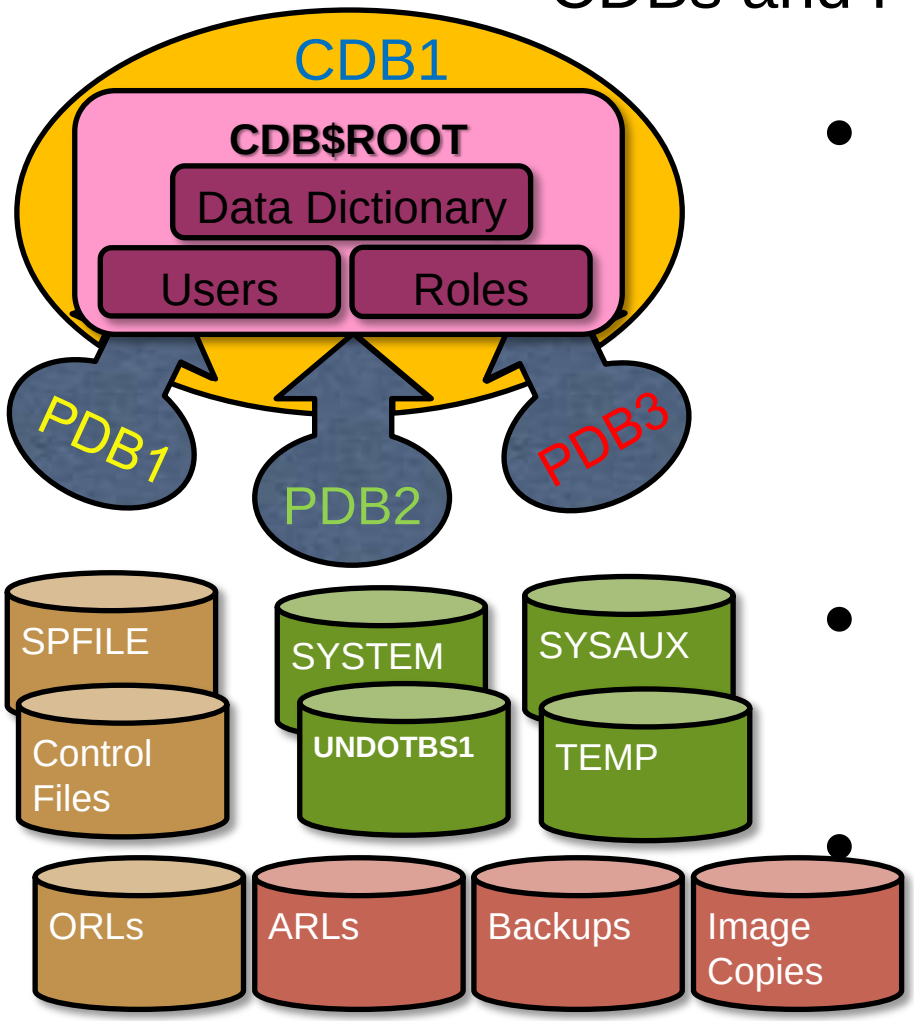
Multi-Tenancy: CDBs and PDBs

Oracle 12*c* offers a completely new multi-tenancy architecture for databases and instances:

- A Container Database (CDBs) comprises one or more Pluggable Databases (PDBs)
- CDBs are databases that contain common elements shared with PDBs
- PDBs comparable to traditional databases in prior releases …
- …but PDBs offer extreme flexibility for cloning, upgrading, and application workload localization

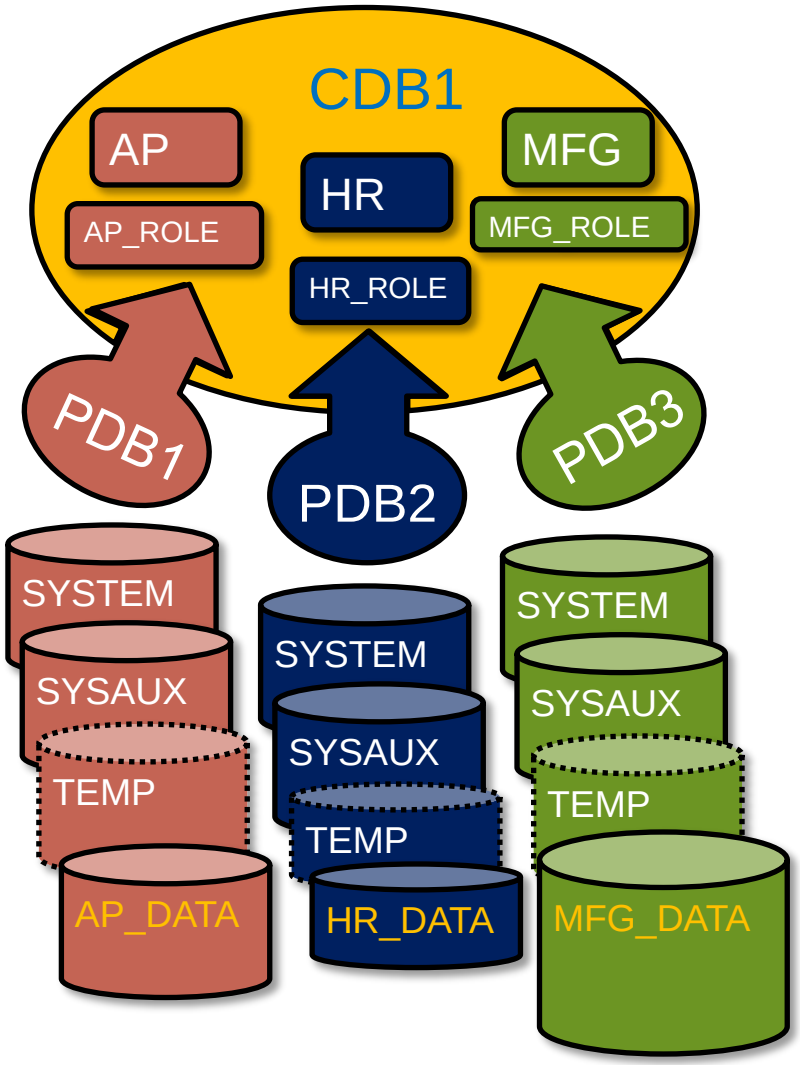
CDBs and Common Objects

CDBs and PDBs share *common* objects



- A CDB owns in common:
  - Control files and SPFILE
  - Online and archived redo logs
  - Backup sets and image copies
- Each CDB has *one* SYSTEM, SYSAUX, UNDO, and TEMP tablespace
- Oracle-supplied *data dictionary objects, users, and roles* are shared globally between CDB and *all* PDBs

PDBs and Local Objects

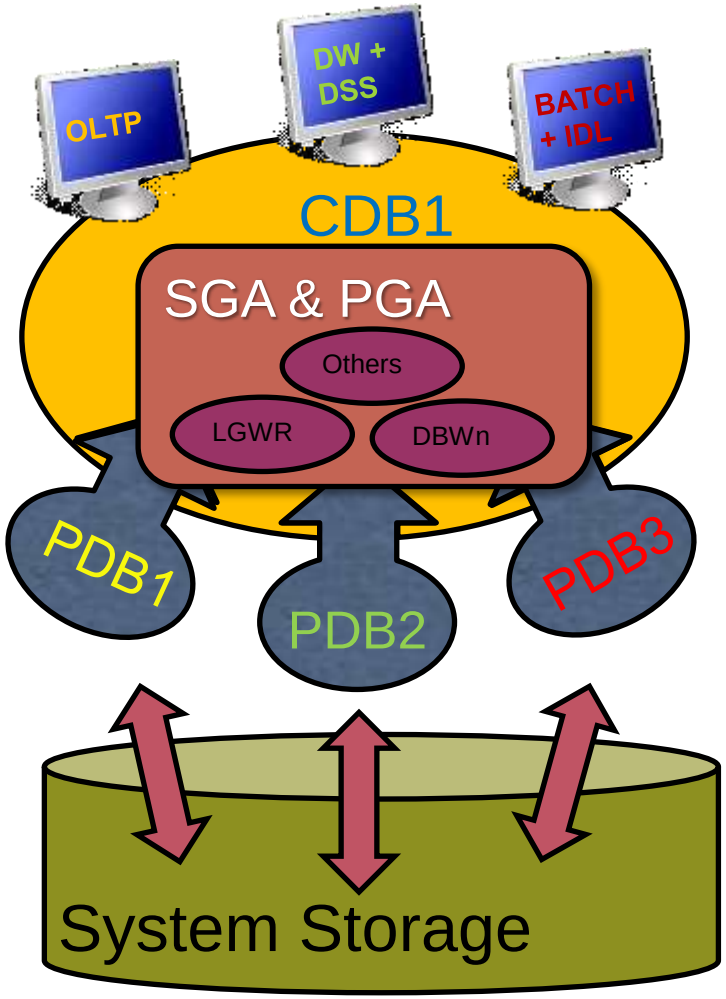


PDBs also own *local* objects

- PDBs have a local SYSTEM and SYSAUX tablespace
- PDBs may have their own local TEMP tablespace
- PDBs can own one or more application schemas:
  - Local tablespaces
  - Local users and roles
- PDBs own all application objects within their schemas

By default, PDBs can only see their own objects

Shared Memory and Processes



CDBs and PDBs also share common *memory* and *background processes*

- *All* PDBs share same SGA and PGA
- *All* PDBs share same background processes
  - OLTP: Intense random reads and writes (DBWn and LGWR)
  - DW/DSS: Intense sequential reads and/or logical I/O
  - Batch and Data Loading: Intense sequential physical reads and physical writes

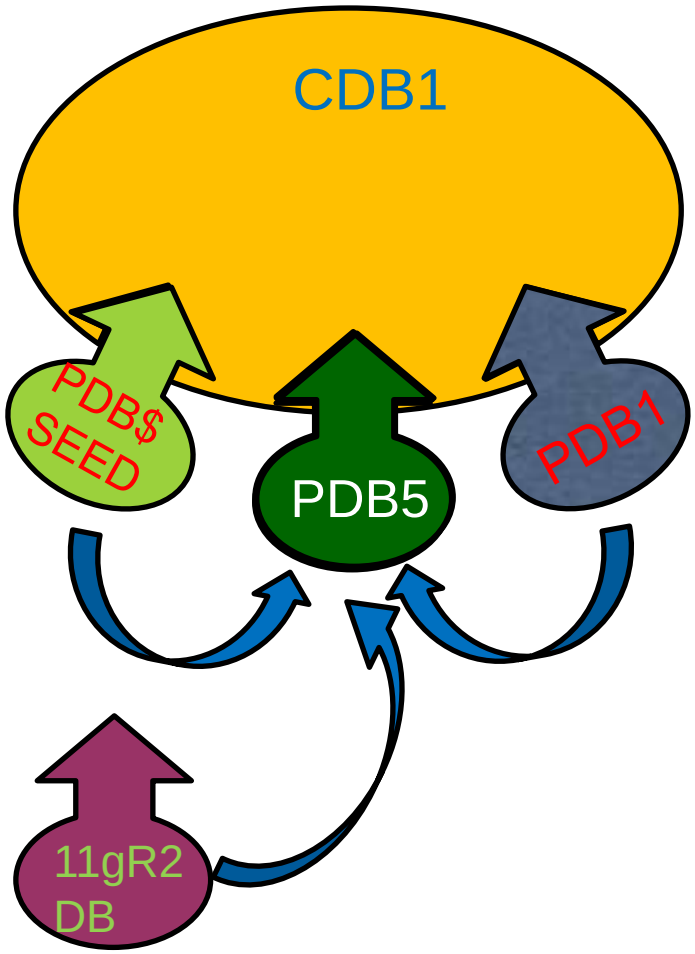
Sharing: It's a Good Thing!

Sharing common resources - when it makes sense - tends to reduce contention as well as needless resource over-allocation:

- Not all PDBs demand high CPU cycles
- Not all PDBs have same memory demands
- Not all PDBs have same I/O bandwidth needs
  - DSS/DW: MBPS
  - OLTP: IOPS and Latency

Result: More instances with less hardware

PDBs: Ultra-Fast Provisioning



Four ways to provision PDBs:

1. Clone from PDB$SEED
2. Clone from existing PDB
3. “Replugging” previously “unplugged” PDB
4. Plug in non-CDB as new PDB

All PDBs already plugged into CDB stay alive during these operations!

Cloning From PDB$SEED

Prerequisites to Oracle 12*c*R1 PDB Cloning

- A valid *Container Database* (CDB) must *already* exist
- The CDB must *permit* pluggable databases
- Sufficient space for new PDB's database files must *exist*

Defining PDB Database File Destinations

- Declare the new PDB's destination

Parameter	Used During
FILE_NAME_CONVERT	Inline during cloning
DB_CREATE_FILE_DEST	In parameter file during cloning and database creation
PDB_FILE_NAME_CONVERT	In parameter file during cloning of PDBs only

```
CREATE PLUGGABLE DATABASE dev_ap
  ADMIN USER dev_ap_adm
  IDENTIFIED BY "P@5$w0rD"
 ROLES=(CONNECT)
  FILE_NAME_CONVERT =
   ('/u01/app/oracle/oradata/pdbseed',
    '/u01/app/oracle/oradata/dev_ap');
```

Cloning From the PDB Seed Database

A new PDB is cloned in a matter of *seconds*

From CDB1 instance's alert log:

```
CREATE PLUGGABLE DATABASE dev_ap
    ADMIN USER dev_ap_admin
    IDENTIFIED BY *    ROLES=(CONNECT)

Tue Apr 08 16:50:18 2014
****************************************************************
Pluggable Database DEV_AP with pdb id - 4 is created as UNUSABLE.
If any errors are encountered before the pdb is marked as NEW,
then the pdb must be dropped
****************************************************************
Deleting old file#5 from file$
Deleting old file#7 from file$
Adding new file#45 to file$(old file#5)
Adding new file#46 to file$(old file#7)
Successfully created internal service dev_ap at open
ALTER SYSTEM: Flushing buffer cache inst=0 container=4 local
****************************************************************
Post plug operations are now complete.
Pluggable database DEV_AP with pdb id - 4 is now marked as NEW.
****************************************************************
Completed: CREATE PLUGGABLE DATABASE dev_ap
    ADMIN USER dev_ap_admin
    IDENTIFIED BY *    ROLES=(CONNECT)
```

Completing PDB Cloning Operations

Once the PDB$SEED tablespaces are cloned, the new PDB must be opened in READ WRITE mode:

2

```
SQL> ALTER PLUGGABLE DATABASE dev_ap OPEN;
```

From CDB1 instance's alert log:

```
alter pluggable database dev_ap open
Tue Apr 08 16:51:39 2014
Pluggable database DEV_AP dictionary check beginning
Pluggable Database DEV_AP Dictionary check complete
Due to limited space in shared pool (need 6094848 bytes, have 3981120
bytes), limiting Resource Manager entities from 2048 to 32
Opening pdb DEV_AP (4) with no Resource Manager plan active
Tue Apr 08 16:51:56 2014
XDB installed.
XDB initialized.
Pluggable database DEV_AP opened read write
Completed: alter pluggable database dev_ap open
```

Cloning from Existing PDBs

Cloning a New PDB From Another PDB

Declare the new PDB's destination directory:

1

```
*.DB_CREATE_FILE_DEST =
  '/u01/app/oracle/oradata/qa_ap'
… or …
*.PDB_FILE_NAME_CONVERT_DEST =
  '/u01/app/oracle/oradata/qa_ap'
```

Connect as CDB$ROOT and quiesce the source PDB in READ ONLY mode:

2

```
SQL> CONNECT / AS SYSDBA;
SQL> ALTER PLUGGABLE DATABASE prod_ap
     CLOSE IMMEDIATE;
SQL> ALTER PLUGGABLE DATABASE prod_ap
     READ ONLY;
```

Clone the target PDB:

3

```
SQL> CREATE PLUGGABLE DATABASE qa_ap
     FROM prod_ap;
```

Cloning From the PDB Seed Database

From CDB1 instance's alert log:

```
Mon Mar 31
ALTER SYST
Pluggable
Completed
ALTER PLU
Mon Mar 31
Due to lim
bytes), li
Opening pd
Pluggable
Completed
CREATE PL
Mon Mar 31
**********
Pluggable
If any err
then the p
**********
Deleting o
Deleting o
Deleting o
. . .
Deleting o
Deleting o
```

<< continued >>

```
Adding new file#23 to file$(old file#8)
Adding new file#24 to file$(old file#9)
. . .
Adding new file#28 to file$(old file#14)
Marking tablespace #7 invalid since it is not present
in the describe file
Marking tablespace #8 invalid since it is not present
in the describe file
Marking tablespace #9 invalid since it is not present
in the describe file
. . .
Marking tablespace #12 invalid since it is not present
in the describe file
Marking tablespace #13 invalid since it is not present
in the describe file
Marking tablespace #14 invalid since it is not present
in the describe file
Successfully created internal service qa_ap at open
ALTER SYSTEM: Flushing buffer cache inst=0 container=4 local
**********************************************************
Post plug operations are now complete.
Pluggable database QA_AP with pdb id - 4 is now marked as NEW.
**********************************************************
Completed: CREATE PLUGGABLE DATABASE qa_ap FROM prod_ap
```

Completing PDB Cloning Operations

Once the QA_AP database has been cloned, it must be opened in READ WRITE mode:

4

```
SQL> ALTER PLUGGABLE DATABASE qa_ap OPEN;
```

From CDB1 instance's alert log:

```
. . .
alter pluggable database qa_ap open
Mon Mar 31 08:11:47 2014
Pluggable database QA_AP dictionary check beginning
Pluggable Database QA_AP Dictionary check complete
Due to limited space in shared pool (need 6094848 bytes, have 3981120
bytes), limiting Resource Manager entities from 2048 to 32
Opening pdb QA_AP (4) with no Resource Manager plan active
Mon Mar 31 08:11:59 2014
XDB installed.
XDB initialized.
Pluggable database QA_AP opened read write
Completed: alter pluggable database qa_ap open
```

4

“Replugging” an Existing PDB

“Unplugging” An Existing PDB

Connect as CDB$ROOT on **CDB1**, then shut down the *source* PDB:

1

```
SQL> CONNECT / AS SYSDBA;
SQL> ALTER PLUGGABLE DATABASE qa_ap
     CLOSE IMMEDIATE;
```

“Unplug” the existing PDB from its *current* CDB:

2

```
SQL> ALTER PLUGGABLE DATABASE qa_ap
     UNPLUG INTO '/home/oracle/qa_ap.xml';
```

Drop the unplugged PDB from its *current* CDB:

3

```
SQL> DROP PLUGGABLE DATABASE qa_ap;
```

“Replugging” An Existing PDB

Connect as CDB$ROOT at **CDB2**:

1

```
SQL> CONNECT / AS SYSDBA;
```

2

```
SQL> CREATE PLUGGABLE DATABASE qa_ap
     USING ‘/home/oracle/qa_ap.xml’ NOCOPY;
```

“Replug” the *existing* PDB into its *new* CDB

Check PDB’s compatibility with **CDB2**:

3

```
SET SERVEROUTPUT ON
DECLARE
   compat BOOLEAN;
BEGIN
   compat :=
      DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
         pdb_descr_file => '/home/oracle/qa_ap.xml',
        ,pdb_name => ‘qa_ap');
   DBMS_OUTPUT.PUT_LINE('PDB compatible? ‘ || compat);
END;
/
```

4

```
SQL> ALTER PLUGGABLE DATABASE
     qa_ap READ WRITE;
```

Open the replugged PDB in READ WRITE mode

New PDB Features in Release 12.1.0.2

PDBs: Accessibility & Management

- CONTAINERS Clause
  - Queries can be executed in a CDB across *identically-named* objects in *different* PDBs
- OMF File Placement
  - CREATE_FILE_DEST controls default location of all new files in PDB (*really* useful for shared storage!)
- Improved State Management on CDB Restart
  - SAVE STATE: PDB automatically *reopened*
  - DISCARD STATE: PDB left in *default* state (MOUNT)
- LOGGING Clause
  - Controls whether any future tablespaces are created in LOGGING or NOLOGGING mode

PDBs: Cloning Enhancements

- Subset Cloning
  - USER_TABLESPACES clause captures only the *desired tablespaces* during PDB cloning from non-CDB or PDB
- Metadata-Only Cloning
  - NO DATA clause captures *only the data dictionary definitions* – but not the *application data*
- Remote Cloning
  - Allows cloning of a PDB or non-CDB on a *remote server* via a *database link*
- Cloning from 3rd Party Snapshots
  - SNAPSHOT COPY clause enables cloning a PDB *directly from snapshot copies* stored on supported file systems (e.g. ACFS or Direct NFS)

“Upgrating” To 12cR1: Plugging In Non-CDB As PDB

Upgrating* a Non-CDB To a PDB

A pre-12*c*R1 database can be *upgrated** to a 12*c*R1 PDB

- Either …
  - Upgrade the *source* database to 12*c*R1 non-CDB
  - Plug upgraded non-CDB into *existing* CDB as *new* PDB
- … or:
  - Clone *new* empty PDB into existing CDB from PDB$SEED
  - Migrate *data* from source database to *newly-cloned* PDB

**WARNING: As a member of POEM, I am qualified to make up words. For your own safety, please do not try this without a certified POEM member present; poor grammar and misspelling may result.*

Migrating Data From Previous Releases

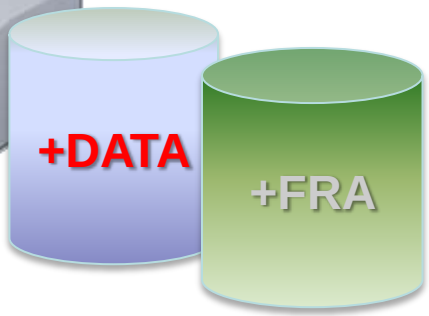
Depending on application downtime requirements:

- Oracle GoldenGate 12*c*
  - Low downtime
  - Separate (*perhaps* expensive) licensing required
- *Cross-Platform* Transportable Tablespaces (XTTS)
  - 12cR1 integrates XTTS operations with DataPump metadata migration *automatically*
- DataPump *Full Tablespace Export* (FTE)
  - 12cR1 integrates DataPump full export metadata migration with TTS *automatically*

DataPump: Full Tablespace Export (FTE)

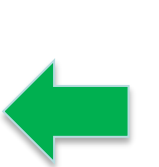
- ✓ **ARCHIVELOG ON**
- ✓ **COMPATIBLE >= 11.2.0.3**
- ✓ **OPEN READ WRITE**

Oracle DBA decides to transform data from **11.2.0.3** to **12cR1** using DataPump FTE :

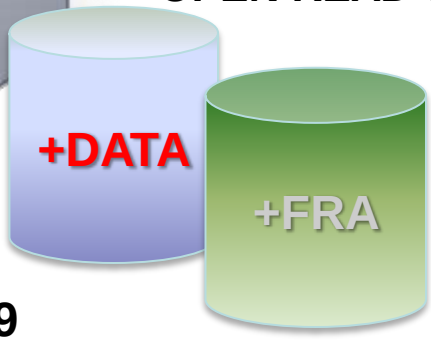


RMAN> BACKUP **FOR TRANSPORT**
FORMAT '+DATA'
DATAPUMP FORMAT '/home/oracle/dp_fte.dmp'
TABLESPACE ap_hot_data;

DBMS_FILE_TRANSFER



- ✓ **ARCHIVELOG ON**
- ✓ **COMPATIBLE = 12.0**
- ✓ **OPEN READ WRITE**



RMAN> RESTORE FOREIGN TABLESPACE ap_hot_data
FORMAT '+DATA'
FROM BACKUPSET '+FRA'
DUMP FILE FROM BACKUPSET '/home/oracle/dp_fte.dmp';

Uber-Fast Database Recovery: 12cR1 Recovery Manager (RMAN) Enhancements

Backup, Restore, and Recover Non-CDBs, CDBs, and PDBs

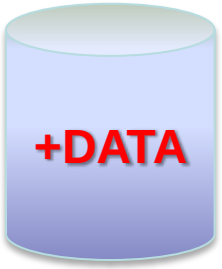
- Image copy backups now support multi-section, multi-channel BACKUP operations
  - SECTION SIZE directive fully supported
  - Faster image copy file creation
- What is backed up in multiple section(s) …
- …can be restored in multi-channel fashion more quickly
- Backups for TTS can now be taken with tablespace set open in READ WRITE mode

BACKUP AS COPY … SECTION SIZE

```
RMAN> # Back up just one tablespace set with SECTION SIZE
BACKUP
  AS COPY
  SECTION SIZE 100M
  TABLESPACE ap_data, ap_idx;
Starting backup at 2014-04-07 13:11:33
using channel ORA_DISK_1
using channel ORA_DISK_2
<< continued >>
channel ORA_DISK_1: starting datafile copy
input datafile file number=00015 name=+DATA/NCDB121/DATAFILE/ap_idx.290.836526787
backing up blocks 12801 through 25600
output file name=+FRA/NCDB121/DATAFILE/ap_idx.298.844261895 tag=TAG20140407T131133
channel ORA_DISK_2: datafile copy complete, elapsed time: 00:00:05
channel ORA_DISK_2: starting datafile copy
input datafile file number=00015 name=+DATA/NCDB121/DATAFILE/ap_idx.290.836526787
backing up blocks 25601 through 38400
output file name=+FRA/NCDB121/DATAFILE/ap_data.270.844261895 tag=TAG20140407T131133
channel ORA_DISK_3: datafile copy complete, elapsed time: 00:00:05
output file name=+FRA/NCDB121/DATAFILE/ap_data.270.844261895 tag=TAG20140407T131133
channel ORA_DISK_4: datafile copy complete, elapsed time: 00:00:06
output file name=+FRA/NCDB121/DATAFILE/ap_idx.298.844261895 tag=TAG20140407T131133
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:05
output file name=+FRA/NCDB121/DATAFILE/ap_idx.298.844261895 tag=TAG20140407T131133
channel ORA_DISK_2: datafile copy complete, elapsed time: 00:00:04
Finished backup at 2014-04-07 13:11:44 << or about 11 seconds total time!
```

RMAN: Table-Level Recovery

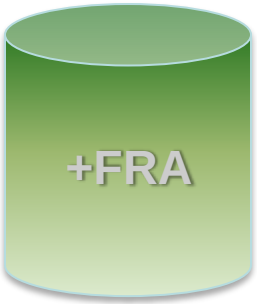
- ✓ **ARCHIVELOG ON**
- ✓ **COMPATIBLE = 12.0**
- ✓ **OPEN READ WRITE**



Oracle DBA decides to roll back table **AP.VENDORS** to prior point in time (e.g. 24 hours ago), but:

- ➢ FLASHBACK VERSIONS Query, FLASHBACK Query, or FLASHBACK TABLE can't rewind far enough because UNDOTBS is exhausted
- ➢ FLASHBACK … TO BEFORE DROP impossible
- ➢ FLASHBACK DATABASE impractical

RMAN> RECOVER TABLE ap.vendors
UNTIL TIME 'SYSDATE – 1'
AUXILIARY DESTINATION '+AUX';



- ✓ Appropriate RMAN backup files located on **+FRA**
- ✓ RMAN creates auxiliary destination on **+AUX**
- ✓ Tablespace(s) for **AP.VENDORS** restored + recovered to prior TSPITR
- ✓ DataPump *exports* table (**AP.VENDORS**) into dump set in **+AUX**
- ✓ DataPump *imports* recovered data back into **+DATA** from **+AUX**

Customizing Table-Level Recovery

Table-level recovery is customizable:

- NOTABLEIMPORT tells RMAN to stop before recovered objects are imported into the target database
- REMAP TABLE renames recovered tables and table partitions during IMPORT
- REMAP TABLESPACE permits remapping of table partitions into different tablespaces

In-Memory Column Store: A Revolution in SQL Query Performance

In-Memory Column Store

- A new way to store data *in addition to* DBC
  - Can be enabled for specific *tablespaces*, *tables*, and *materialized views*
- Significant performance improvement for queries that:
  - Filter a *large number of rows* (=, <, >, IN)
  - Select a *small* number of columns from a table with *many* columns
  - Join a *small* table to a *larger* table
  - Perform *aggregation* (SUM, MAX, MIN, COUNT)
- Eliminates need for *multi-column* indexes

In-Memory Column Store: Setup

Allocate memory for the In-Memory Column Store, then bounce the database instance:

1

```
SQL> ALTER SYSTEM
       SET inmemory_size = 128M
       SCOPE=SPFILE;

SQL> SHUTDOWN IMMEDIATE;

SQL> STARTUP;
```

Add the desired table to the In-Memory Column Store:

2

```
SQL> CONNECT / AS SYSDBA;

SQL> ALTER TABLE ap.randomized_sorted
       INMEMORY
       MEMCOMPRESS FOR QUERY HIGH
       PRIORITY HIGH;
```

In-Memory Column Store: Results