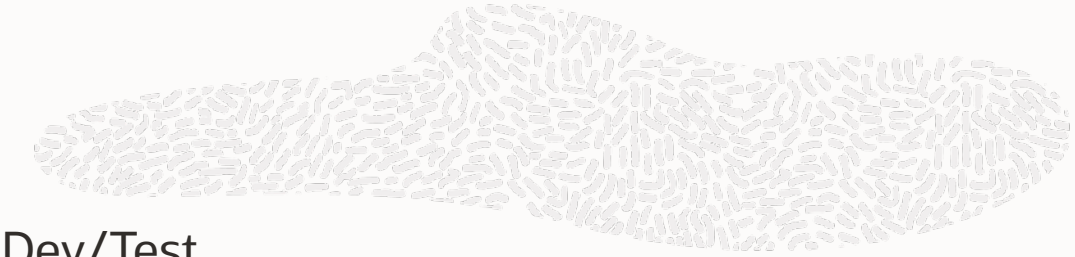
ORACLE

Exadata ACFS Snapshots & Sparse Clones

Database Clones for Development and Test

March 2022

Database Clones on Exadata



- Many organizations use Exadata for Production, DR & Dev/Test
- Single solution platform for all production and test/dev databases use cases
- Exadata is the best platform to run Oracle Database

Test/Dev Use Cases	Oracle's Solution
Full End-to-End performance testing	Non-Sparse Exadata Identical or comparable system as primary
Testing with simple snapshot use cases and Exadata smart features	Exadata Sparse
Advanced snapshot capabilities similar to third party copy-on-write but no Exadata offload features required	ACFS Snapshots on Exadata

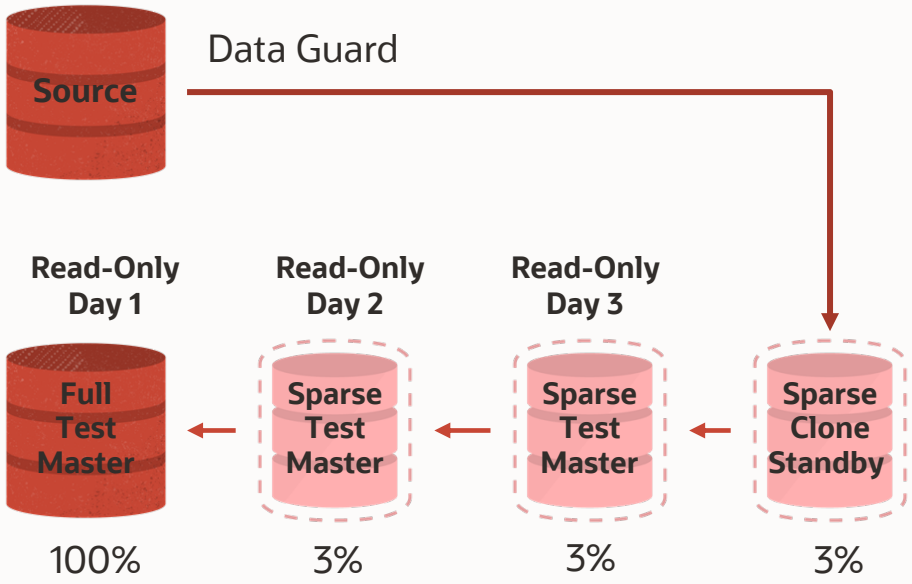
Comparing Sparse Clones vs. Storage Snapshots

Sparse Clone (copy-on-write)

The master of a sparse clone is read-only

Sparse clones contain changed blocks

Blocks accumulate as the clone changes

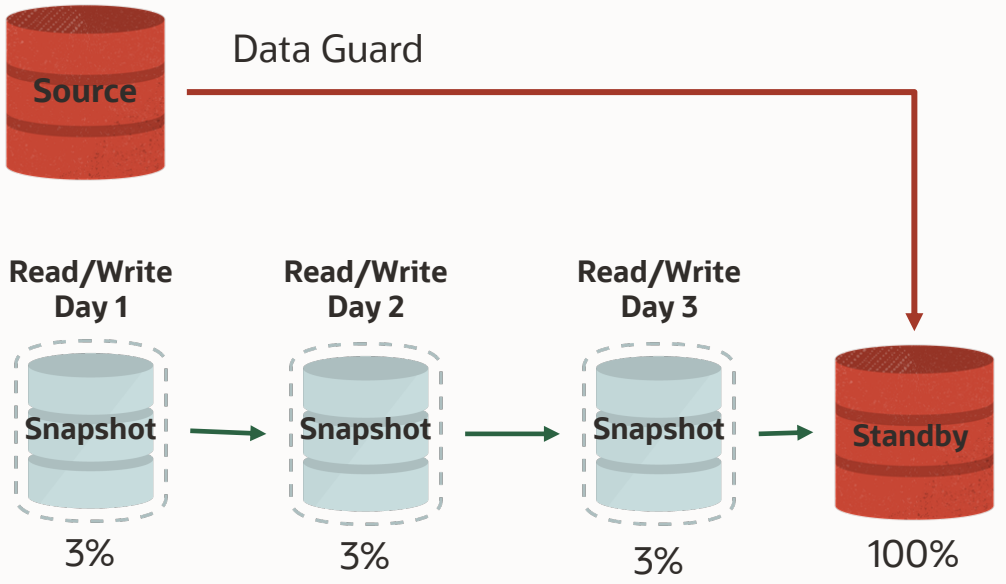


Snapshot (preserve prior block versions)

The master of a snapshot is read/write

Snapshots preserve older block versions

Blocks accumulate as the master changes



Exadata Sparse Clones

Integral Part of Exadata

- Fully compatible with Exadata storage features (SQL offload, I/O prioritization, etc.)

Space Efficient Sparse Clones

- Uses copy-on-write technology internally
- Each snapshot only contains changes to data based on a master copy, including sparse masters

Support Range-of-Data Timeline

- Cascade Masters from Data Guard Feeds
- Choose any point in a timeline of data changes (hour, day, month, year)

Space Efficient Sparse Backups

- Backups are also space-efficient, backing up only the changes in a sparse clone

Data Obfuscation Capabilities

- Oracle Data Masking, Subsetting, Data Redaction, Virtual Private Database

ACFS Snapshots on Exadata

Space Efficient Filesystem Snapshots

- Uses copy-on-write technology internally
- Snapshot contains changes made from parent snapshot

Support Multiple Timelines

- Cascade Masters from Data Guard Feeds
- Choose any point in a timeline of data changes (hour, day, month, year)
- Supports up to 1023 Snapshots per Filesystem

Data Obfuscation Capabilities

- Oracle Data Masking, Subsetting, Data Redaction, Virtual Private Database

Exadata Feature Support

- Supports Exadata Smart Flash Cache only

Agenda

Exadata Sparse Clones

- Features
- Hierarchical Snapshots
- Sparse Test Masters
- Monitoring and Statistics
- Resources

ACFS Snapshots

Summary

Agenda

Exadata Sparse Clones

- Features
- Hierarchical Snapshots
- Sparse Test Masters
- Monitoring and Statistics
- Resources

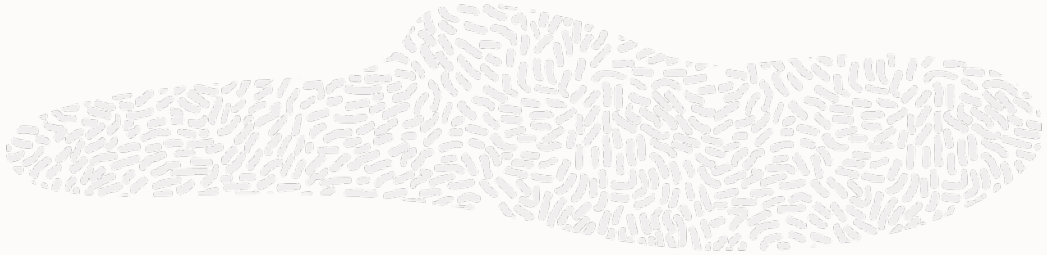
ACFS Snapshots

Summary

Exadata Sparse Snapshots

- Ability to quickly create space efficient test/dev databases on Exadata
- Sparse snapshot test/dev databases can use all Exadata Smart features including smart offload capabilities so applications can evaluate using Exadata features
- Sparse snapshot test/dev databases are NOT full copies resulting in storage and cost savings
- HCC storage compression works transparently providing additional storage savings

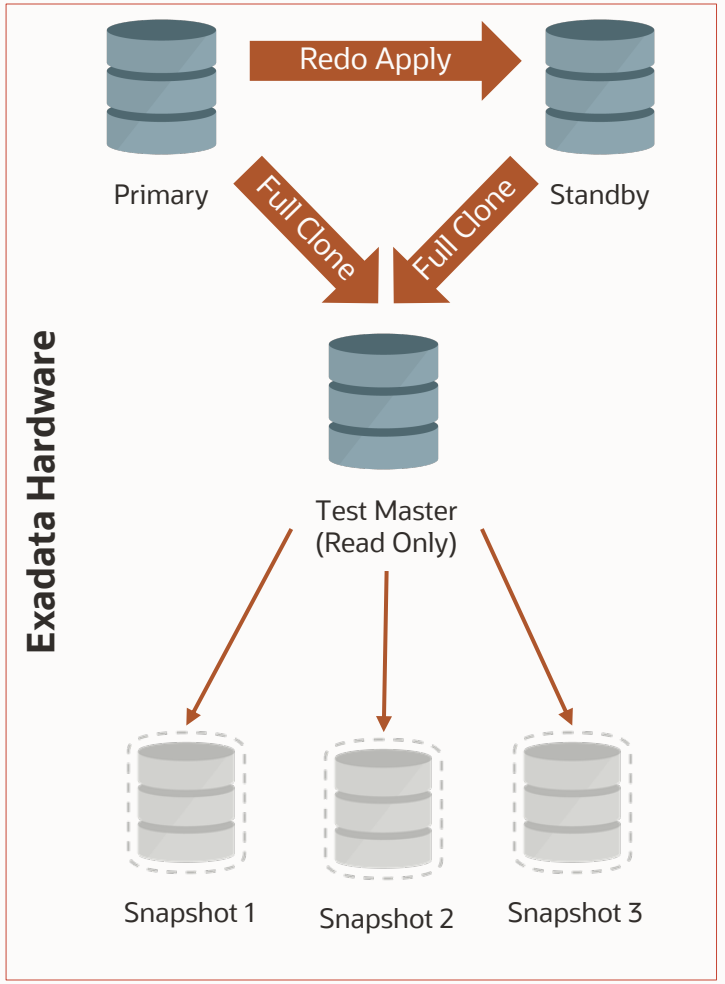
Terminology



- Sparse Griddisk
  - Virtual size
  - Physical size
- Sparse Diskgroup
- Sparse Data Files
- Exadata Snapshots
- Exadata Test Master
- Exadata Sparse Test Master

Test/Dev Lifecycle using Exadata Snapshots

- Production Database runs on Exadata
- Full Cloned Test Master database created from Standby or Production on Exadata
  - Optionally mask sensitive data in Test Master
  - Space efficient Test/Dev databases created from here with one-command for PDBs
  - Exadata Smart features (query offload; storage index; smart log; smart flash cache; HCC; etc) available on snapshots
  - Challenge – Refreshing the Test Master invalidates existing snapshots; must create new full Test Master to create new refreshed Exadata snapshots



Sparse Database/File/GridDisk

Sparse Database

- Only the datafiles in a sparse database are sparse
- controlfile/online redo logs/tempfiles; etc are not sparse

Sparse File

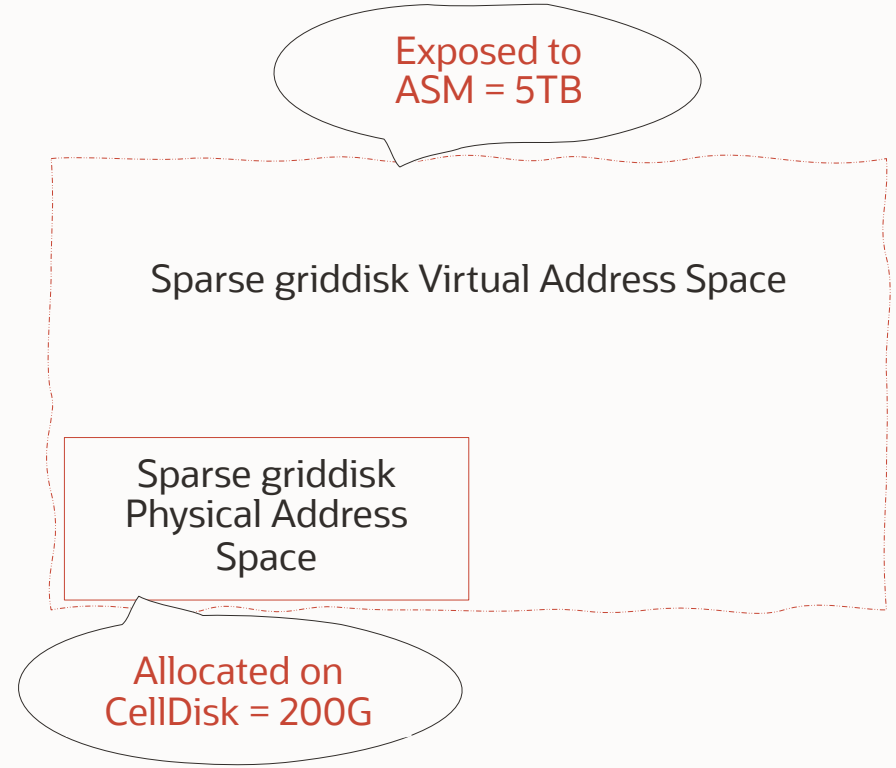
- Sparse datafile points back to Test Master database datafile
- Only allocates blocks on-demand during writes

Sparse Griddisk

- Exposes a virtual size in addition to a physical size
- Max. allowed virtual size/disk = 100TB
- Max. Allowed aggregate physical size/disk = 4TB

Example syntax:

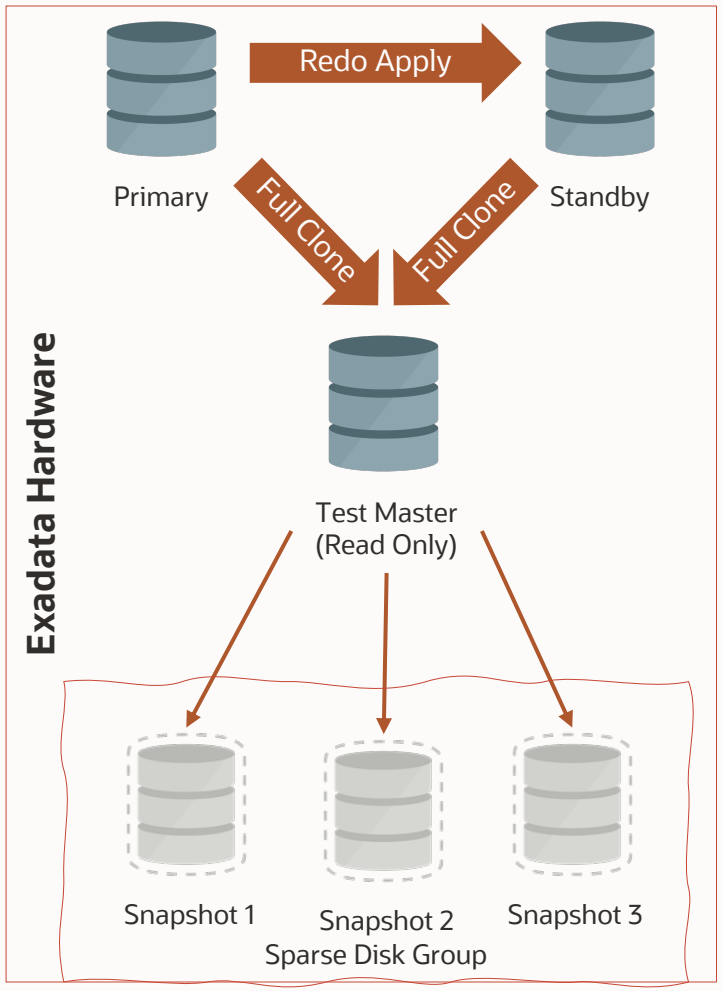
```
Cellcli> create griddisk all harddisk
prefix=SPARSE size=200G,
virtualsize=5TB
```



Sparse Diskgroup

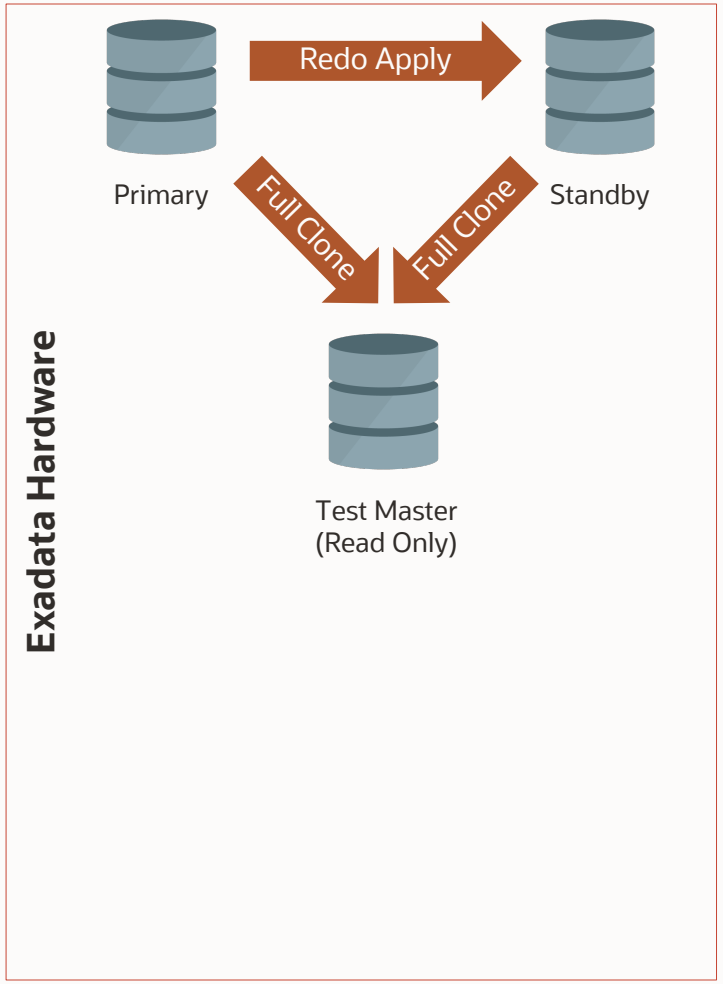
- Sparse Files can only be created in a sparse diskgroup
  - **cell.sparse_dg** is a new attribute that must be set to **allsparse** for a sparse diskgroup
  - Must have **compatible.asm** and **compatible.rdbms** set to 12.1.0.2 or higher
  - Uses 16X extent size for 4M AU; each extent is 64M
  - Sparse diskgroups use Virtually Allocated Metadata

```
SQL> create diskgroup SPARSEDG
  normal redundancy
  disk 'o/*/SPARSE_*'
  attribute
    'compatible.asm'          = '19.0.0.0',
    'compatible.rdbms'        = '12.2.0.2',
    'cell.smart_scan_capable' = 'true',
    'cell.sparse_dg'          = 'allsparse',
    'au_size'                 = '4M';
```



Prepare Exadata for Snapshot Database

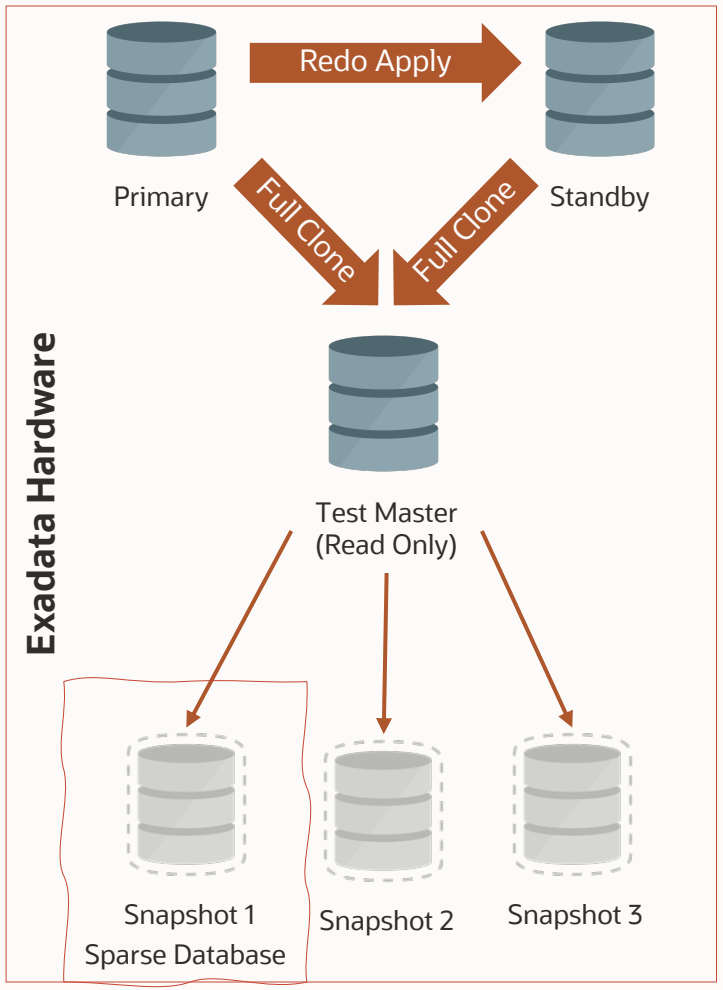
- Create sparse griddisks on the storage cells
- Create sparse diskgroup on the ASM instance
- Setup Test Master Database
  - Enable ASM ACCESS_CONTROL on the Test Master Diskgroup
  - Create full clone and mask; OR
  - Convert existing full clone on Exadata to a Test Master; OR
  - Convert standby database to a Test Master



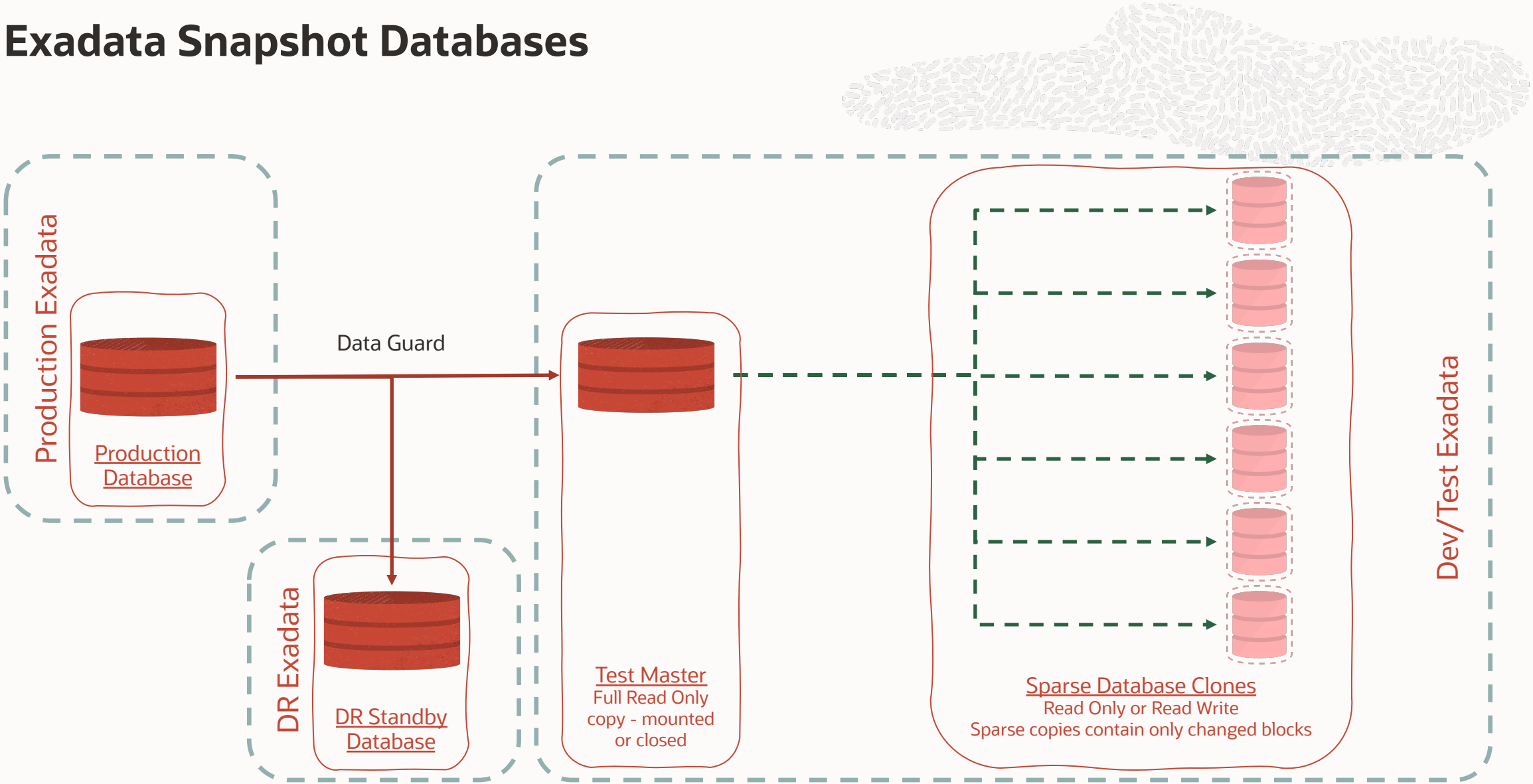
Create Exadata Snapshot Database

- Create Sparse/Snapshot Database
  - **create_file_dest** must be set to a SPARSE diskgroup
  - Pluggable database.. Using SNAPSHOT COPY
  - Non-container database… using DBMS_DNFS.CLONEDB_RENAMEFILE

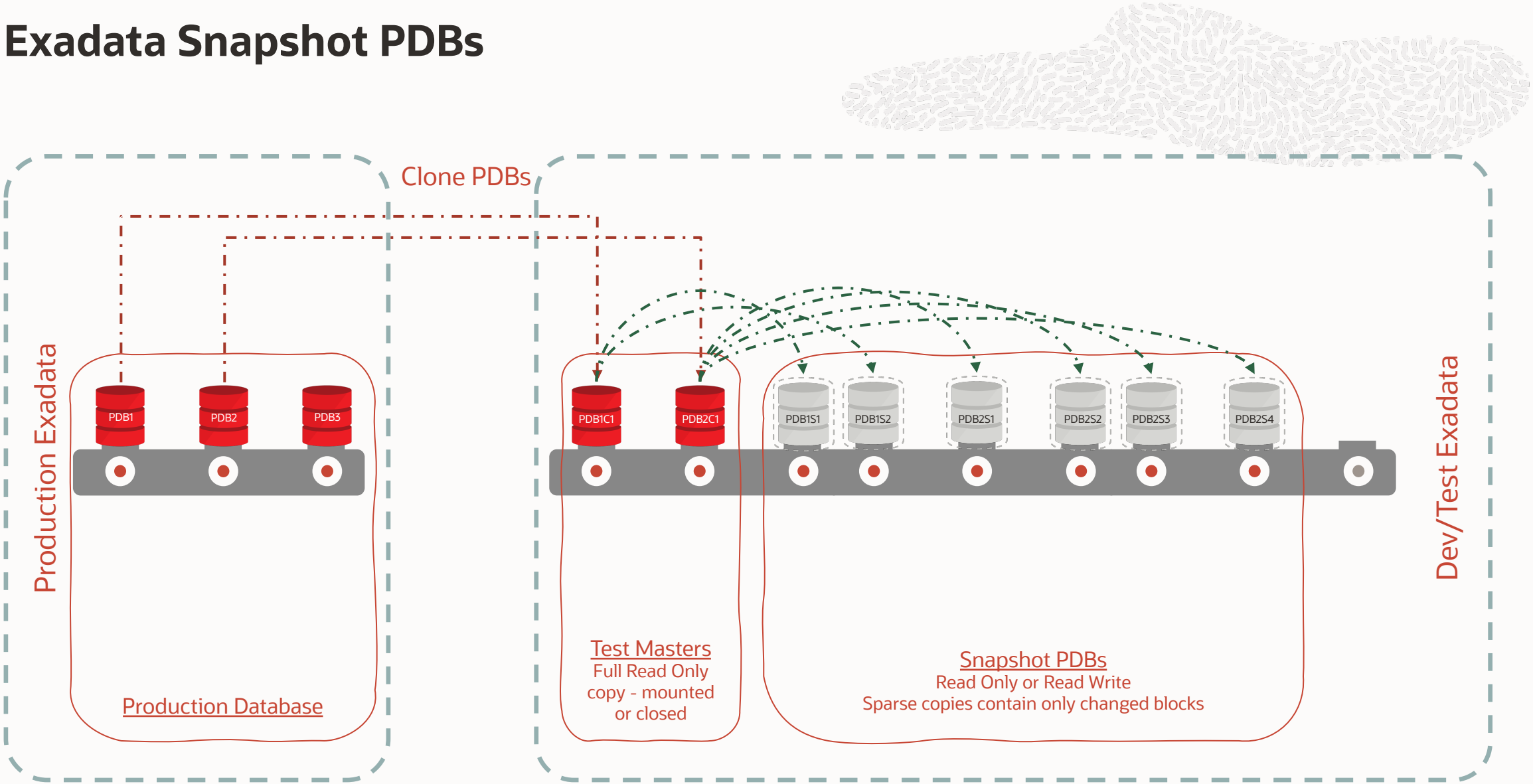
```
SQL> alter pluggable database
TestMaster open read only;

SQL> create pluggable database
JohnTest from TestMaster
create_file_dest='+SPARSEDG'
SNAPSHOT COPY;
```



Exadata Snapshot Databases



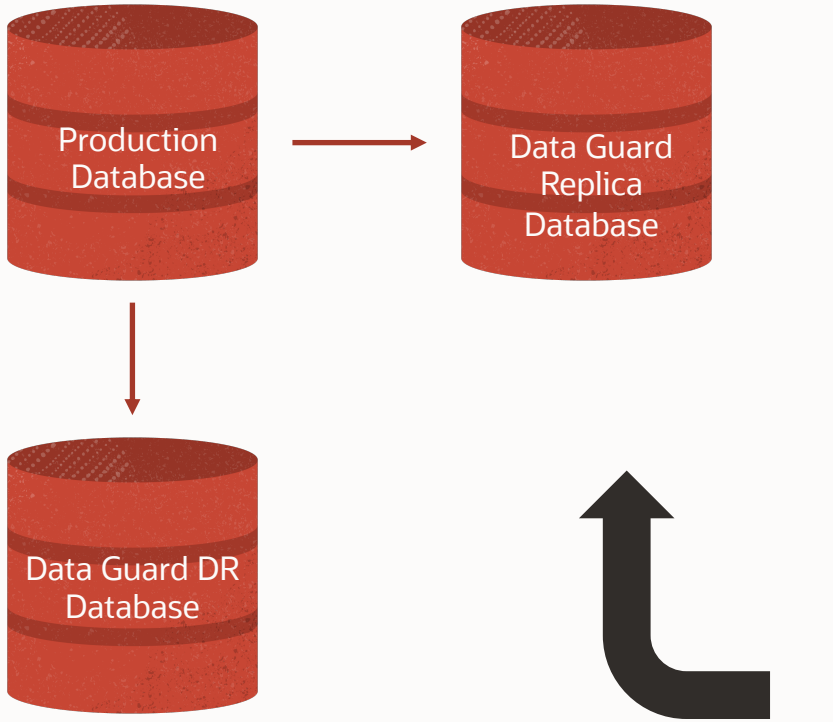
Exadata Snapshot PDBs



Use Standby Database as Test Master

- Standby database cannot be running redo apply while serving as Test Master
- Must have ASM ACCESS_CONTROL enabled + ownership set
- Periodically refresh
  - Drop all snapshots including datafiles
  - Make all Test Master datafiles read write
  - Refresh Test Master from production via
    - DATAGUARD REDO APPLY; OR
    - RMAN RECOVER … FROM SERVICE
  - Close Test Master and make all TM datafiles RO
  - Create new test snapshots for next week's testing

Refresh Standby Test Master Database

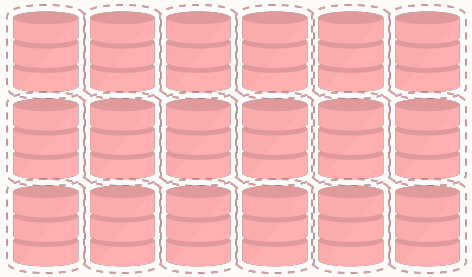


Step 1: Convert to test master database

- Defer redo transport
- Convert to Data Guard snapshot standby
- Prepare to be test master database
- Close database & open read-only

Step 2: Create Snapshots

- Create snapshots and use for dev/test



Step 3: Refresh test master database

- Drop snapshots
- Convert Data Guard snapshot database to original state
- Apply RMAN incremental from production to refresh Data Guard replica
- Enable redo transport to complete refresh
- Repeat Step 1

Efficient Sparse Database Backup & Recovery

Option to create a L0 Sparse backups as Backup Set or Image Copy

```
BACKUP AS [NON]SPARSE {BACKUPSET | COPY} ...
```

Choose Sparse backup option per device

```
CONFIGURE DEVICE TYPE {SBT|DISK}..SPARSE ON|OFF;
```

Restore a Sparse Database as Sparse instead of full

```
RESTORE .. FROM {SPARSE|NONSPARSE} ..
```

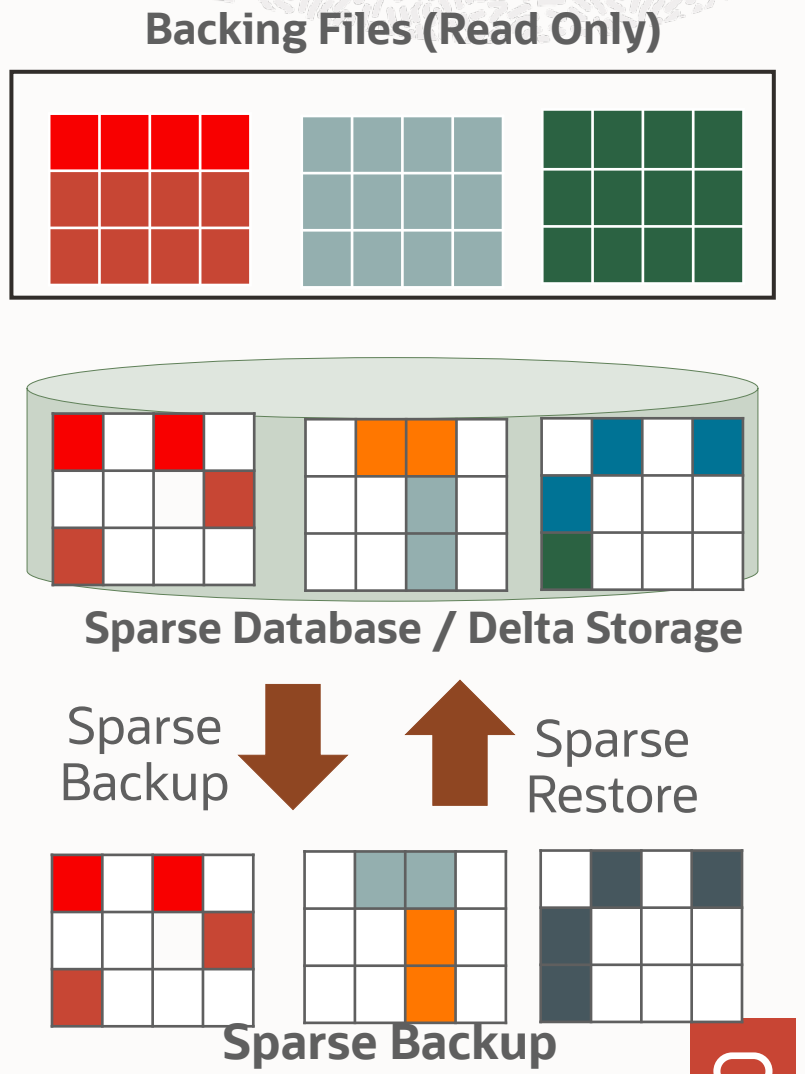
Support for regular RMAN operations

- TSPITR, DUPLICATE, DELETE, LIST, SHOW, CATALOG etc.
- Sparse Backup Database, Tablespace, Datafiles, CDB, PDB
- Duplicate as Sparse Database instead of complete database

```
DUPLICATE DATABASE DB1 AS DB2 FROM SPARSE ...
```

COMPATIBLE parameter to be set to 12.2 or higher

Recommend 18c or later for sparse backup usage



Agenda

Exadata Sparse Clones

- Features
- Hierarchical Snapshots
- Sparse Test Masters
- Monitoring and Statistics
- Resources

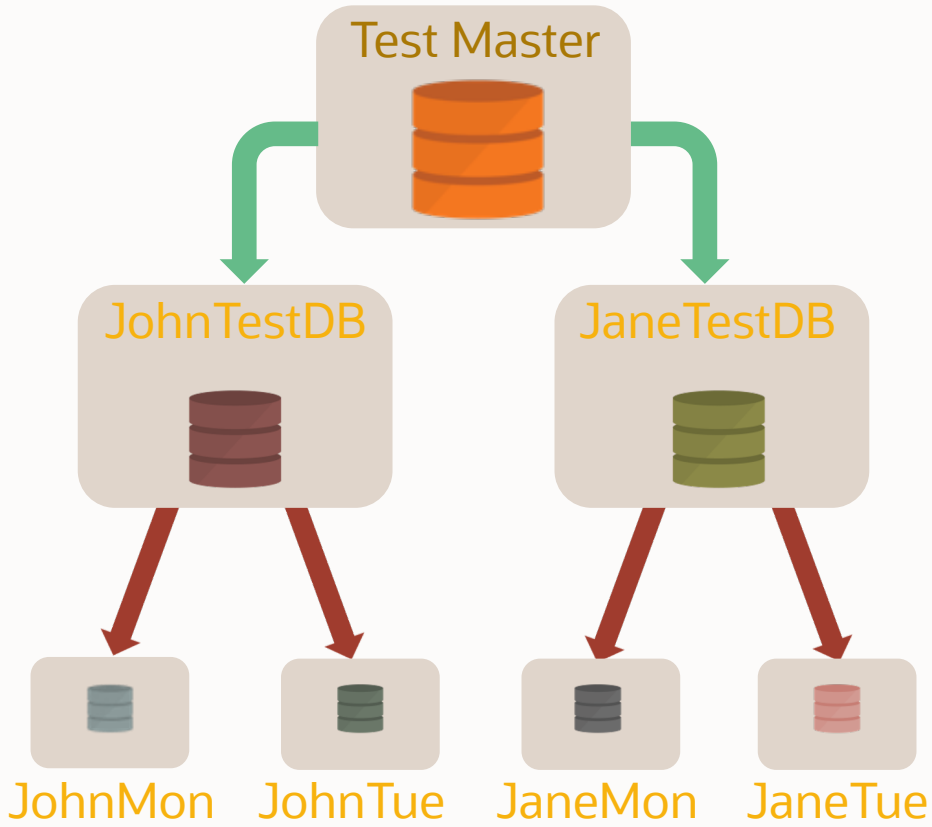
ACFS Snapshots

Summary

Hierarchical Snapshots

Architecture

- Create Snapshots of databases from previously created Snapshots
  - `CREATE PLUGGABLE DATABASE JOHNMon from JOHNTESTDB create_file_dest='+SPARSE' SNAPSHOT COPY;`
- Syntax and technology remain unchanged
- Works with pluggable and non-pluggable databases
- Use case example
  - Development releases nightly build of the database
  - Tester creates a snapshot for himself and finds a bug
  - Tester creates a snapshot of his snapshot
  - Tester provides the new copy back to development for analysis
- Recommend snapshot tree depth <10 for performance



Agenda

Exadata Sparse Clones

- Features
- Hierarchical Snapshots
- Sparse Test Masters
- Monitoring and Statistics
- Resources

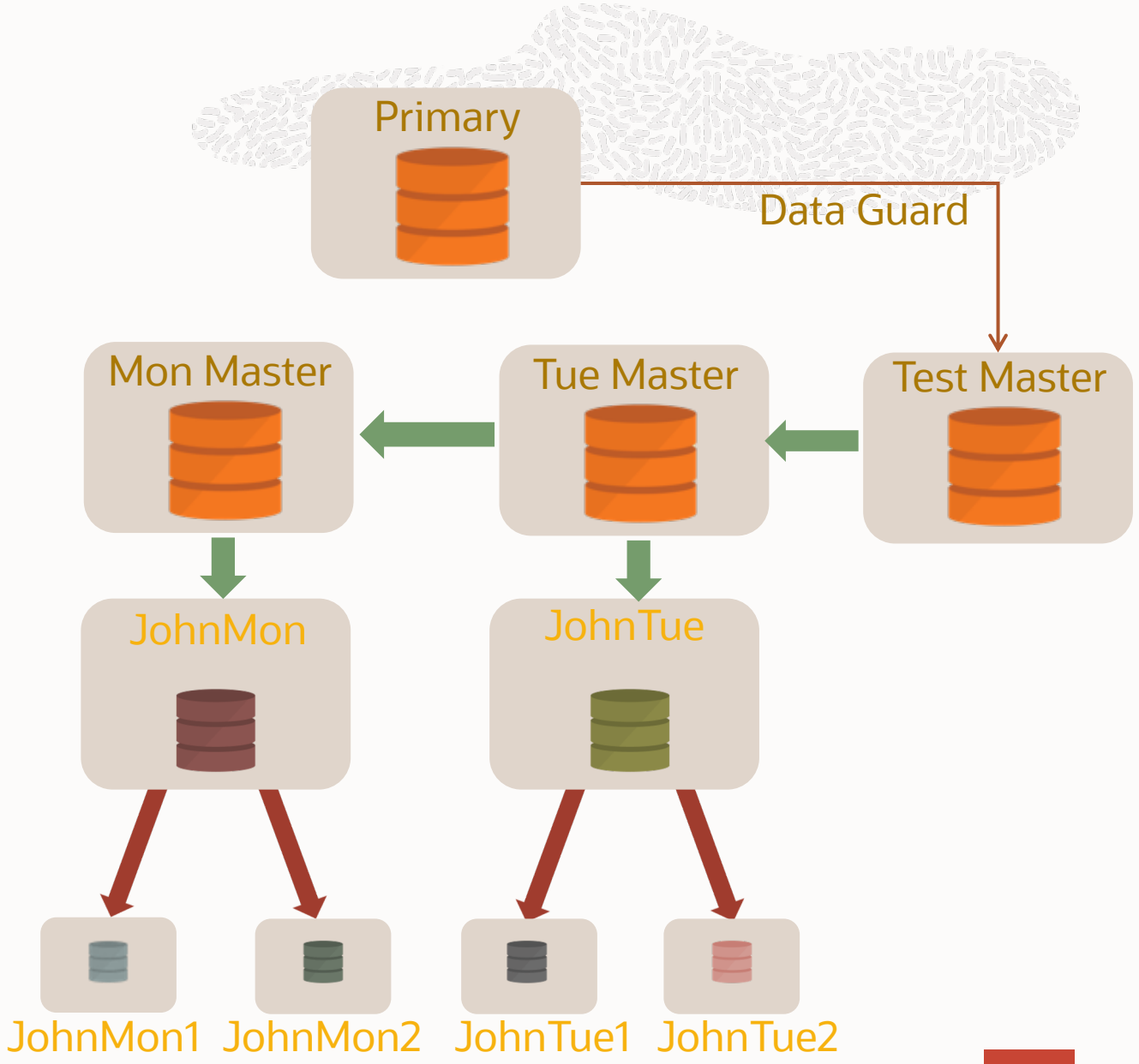
ACFS Snapshots

Summary

Sparse Test Masters

Architecture

- Use cases
  - Ability to create snapshots at different points of time; without using up full space for each point of time.
  - Test Master can be a writable Data Guard Target
- Steps
  - At any point on the Test Master; stop data guard redo apply; and create a ‘Mon Master’.
  - Create a new Test Master which is space efficient snapshot of ‘Mon Master’
  - ‘Mon Master’ is now read only and can be treated as a parent to create other test/dev snapshots.
  - Repeat the step on Test Master to create ‘Tue Master’ (it is space efficient).
  - ‘Tue Master’ is a sparse test master

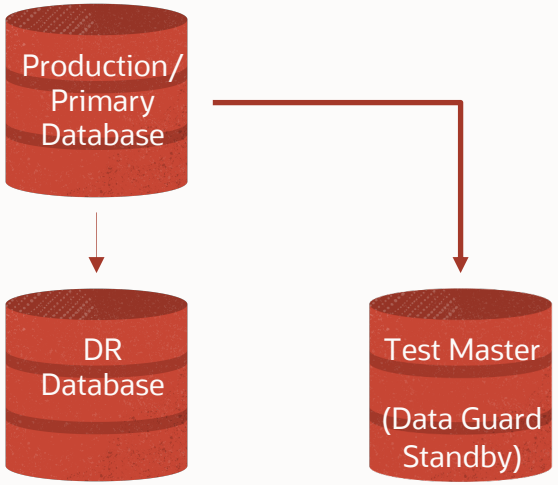


Sparse Test Masters

Lifecycle Start (Create Test Master)

Create new Data Guard or RMAN full sized Test Master (TM) – can be in sparse or non-sparse diskgroup

- Sparse Test Master files must be in sparse diskgroup

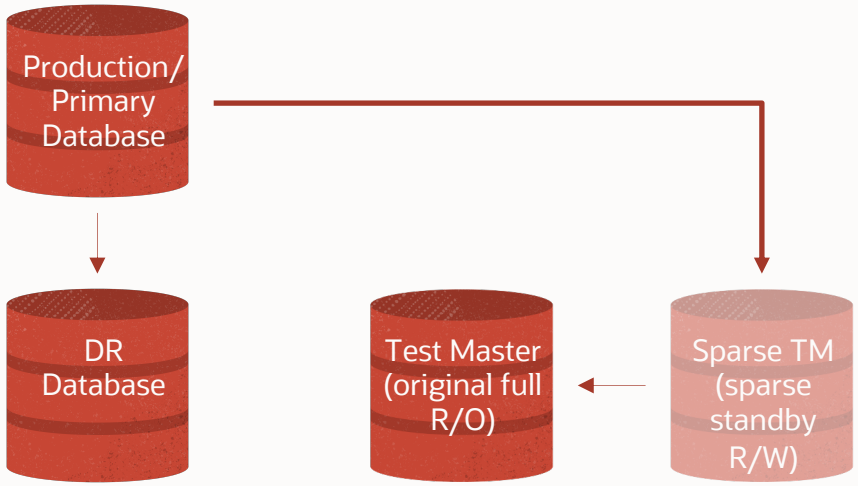


Stop Data Guard Redo Apply to original Test Master

- Make it a Read Only Test Master

Create new Sparse Test Master (for Standby)

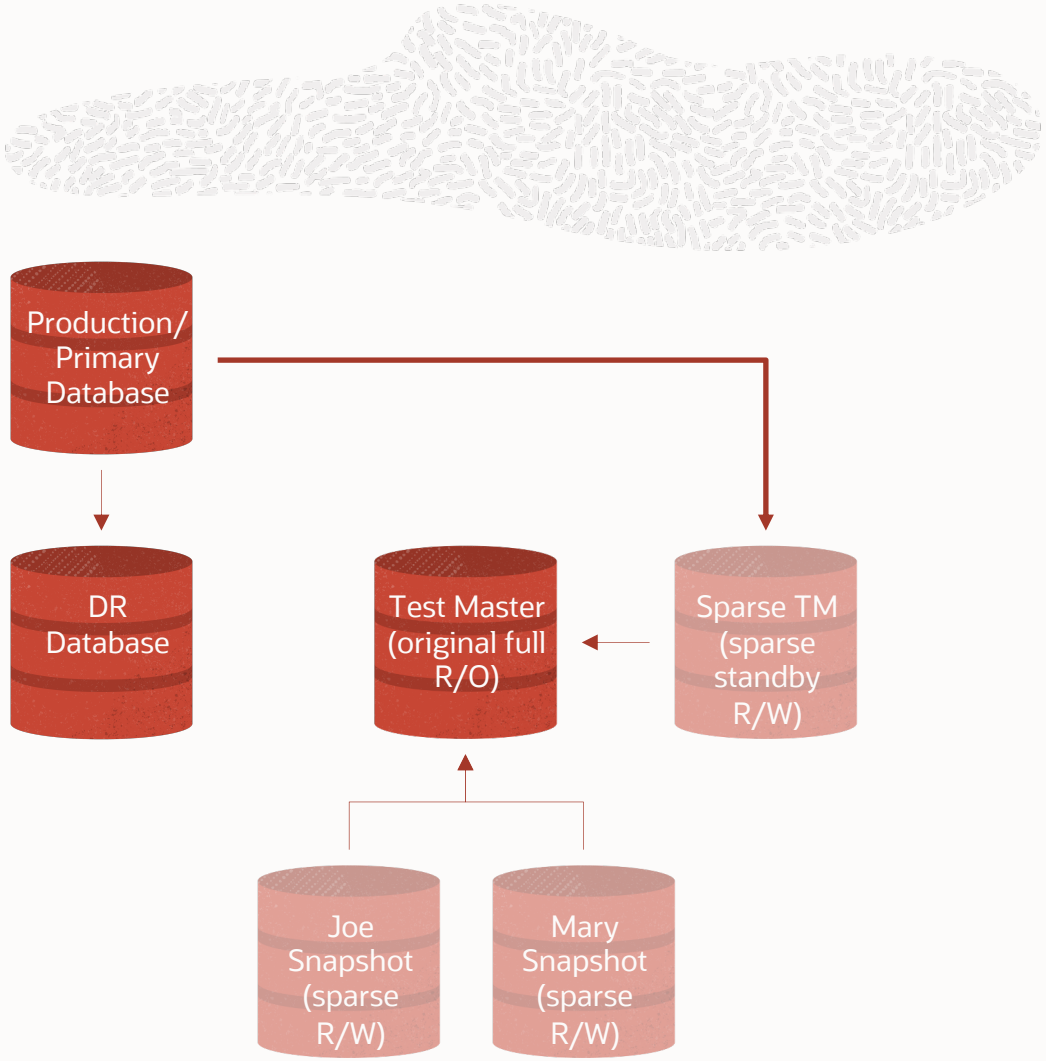
- Start Data Guard Redo Apply to Refresh



Sparse Test Masters

Create Snapshots

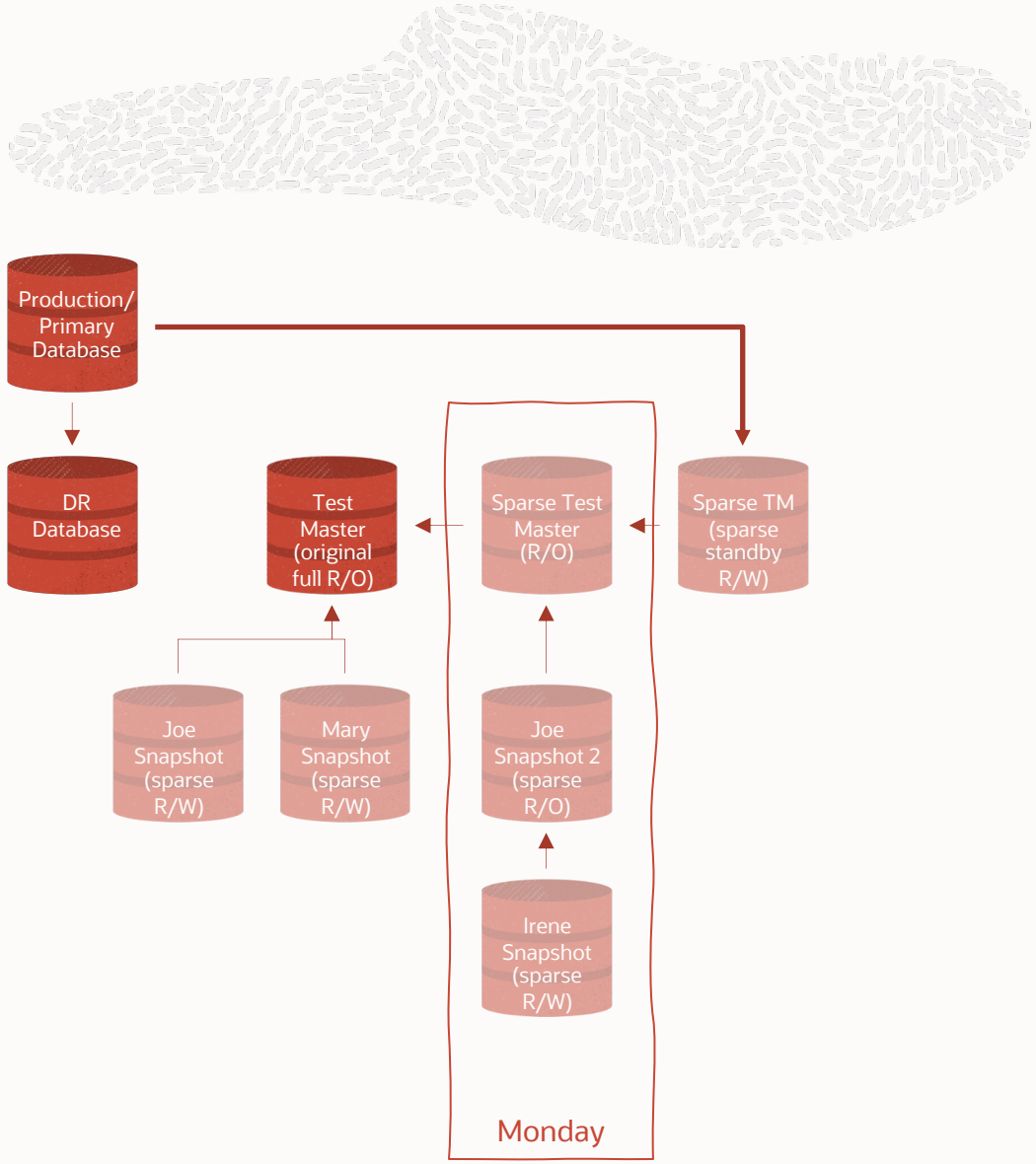
- Test Master original is a Read Only Test Master
- Create Exadata snapshots from it
  - Joe Snap
  - Mary Snap
- Sparse Test Master for Standby receives redo from primary and writes to sparse datafiles



Sparse Test Masters

Refresh Test Master + Create New Snapshots

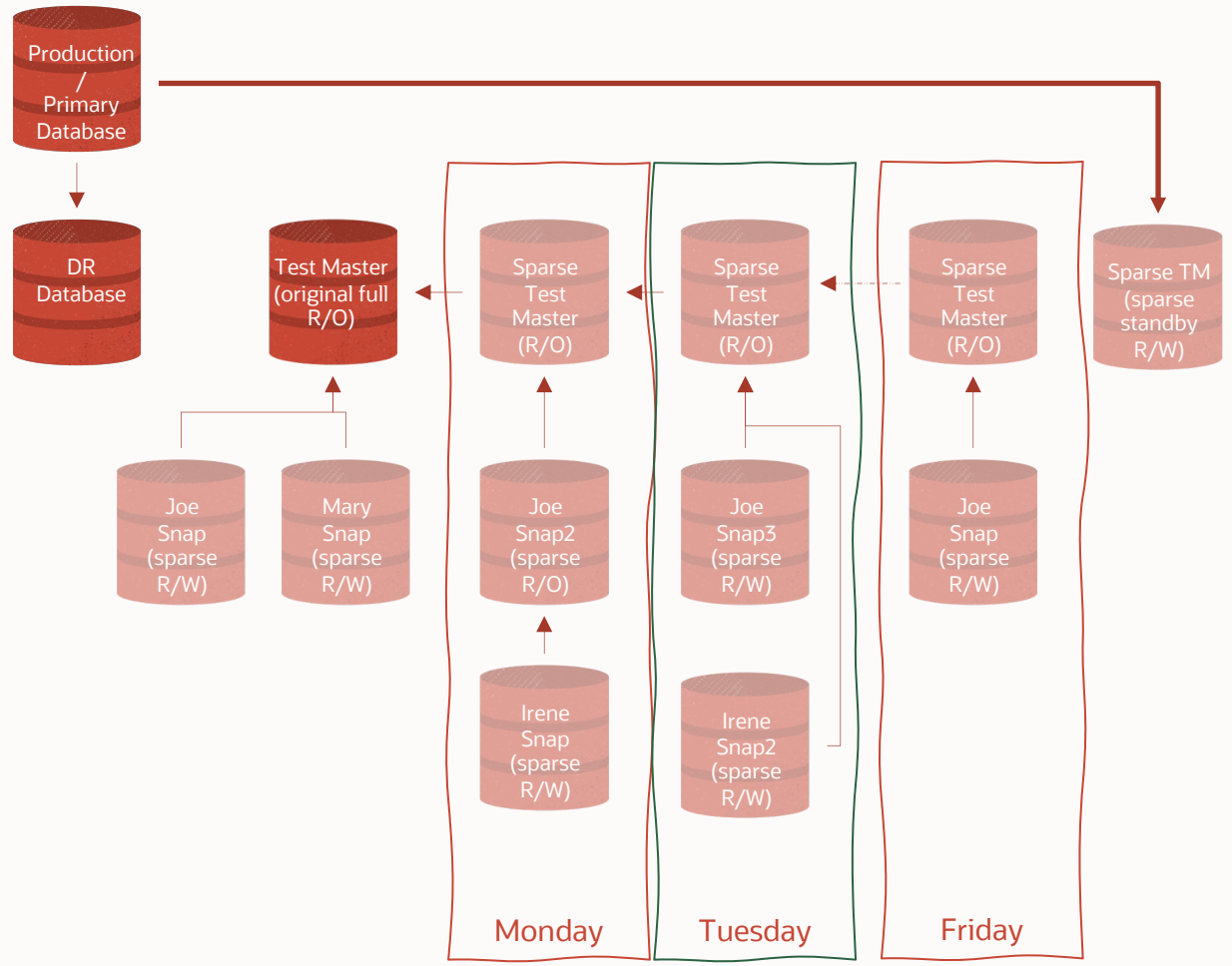
- Repeat process to create new Exadata Snapshots while keeping prior Exadata snapshots
- All Sparse Test Masters and Snapshots are sparse sized



Sparse Test Masters

Additional Test Master Refreshes

- We recommend a maximum of 10 levels – 9 Sparse Test Masters
  - For performance reasons we like to limit the hierarchical tree depth
  - Inclusive of levels of Test Masters and levels beneath a Test Master
- Beyond 10 Sparse Test Master (width)
  - Drop all snapshots and sync TM original current to start over – OR -
  - Space permitting create new full Test Master



Agenda

Exadata Sparse Clones

- Features
- Hierarchical Snapshots
- Sparse Test Masters
- Monitoring and Statistics
- Resources

ACFS Snapshots

Summary

Monitoring and Statistics

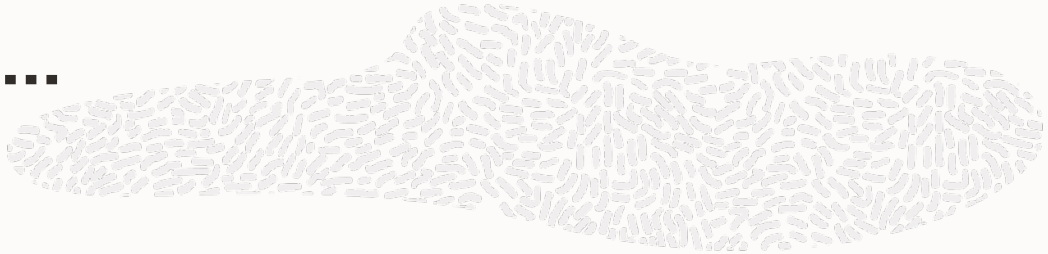
- Sparse IO stats in RDBMS
- Wait events in RDBMS
- v$ views
    - v$asm_disk_sparse
    - v$asm_diskgroup_sparse
    - v$clonedfile

RDBMS stats (v$sysstat; v$mystat)

New stats

Name	Meaning
Physical read snap IO request no data	No physical IO done for these (i.e. wasted roundtrips)
Physical read snap IO request base	Number of physical IOs on base level
Physical read snap IO request copy	Number of physical IOs on any snap hierarchy
Physical read snap bytes base	Number of bytes read from the base
Physical read snap bytes copy	Number of bytes read from the snap

RDBMS stats (v$sysstat; v$mystat) contd…



Updated stats

Name	Meaning
Physical read total IO requests	Number of physical IOs submitted by user
Physical read total multi block requests	Number of multi block IOs submitted by user

- Only the IOs that lead to a real physical IO will be counted here, using the same logic as described in wait events to omit completely sparse IOs.

RDBMS wait events

- Following wait events are monitored for 0 byte reads returned; i.e. sparse buffers
  - cell single block physical read
  - cell multi block physical read
  - cell list of blocks physical read
- List of blocks wait events are also tracked
- Then, we change the wait event to "**cell sparse block physical read**"
  - this wait event is significantly faster since there is no IO involved and if the request is large in size, then even network transfer is significantly faster because of packing

ASM Sparse Disk (v$asm_disk_sparse)

Name	Meaning
GROUP_NUMBER	Number of the diskgroup containing the disk
DISK_NUMBER	Number assigned to the disk within this diskgroup
INCARNATION	Incarnation number for the disk
ALLOCATED_MAT_MB	Total used physical capacity on this disk
TOTAL_MAT_MB	Total physical capacity on this disk
SPARSE_READS	Number of read requests on sparse regions of this disk
SPARSE_BYTES_READ	Bytes read from sparse regions of this disk
SPARSE_READ_TIME	Time taken by sparse read IOs

v$asm_disk_sparse

```
SQL> select
     DISK_NUMBER           dsk_num,
     ALLOCATED_MAT_MB      alloc,
     TOTAL_MAT_MB          total
from V$ASM_DISK_SPARSE
where GROUP_NUMBER = 5;

DSK_NUM     ALLOC       TOTAL
---------- ---------- ----------
         0       5536     204774
         1       5424     204774
         2       5532     204774
         3       5424     204774
         4       5424     204774
```


ASM Sparse Diskgroup (v$asm_diskgroup_sparse)

Name	Meaning
GROUP_NUMBER	Cluster wide number assigned to the diskgroup
ALLOCATED_MAT_MB	Total used physical capacity of the diskgroup
TOTAL_MAT_MB	Total physical capacity of the diskgroup

```
SQL> select
       ALLOCATED_MAT_MB    alloc,
       TOTAL_MAT_MB        total
from V$ASM_DISKGROUP_SPARSE
where GROUP_NUMBER = 5;

ALLOC      TOTAL
---------- ----------
    197208    7371864
```

v$clonedfile

- Only works on mounted databases/files
- Can be run in either database instance or in ASM
  - In snapshot instance will display parent files for that snapshot
  - In ASM instance, possible to see parent/child relationships for all open/mounted snapshots

```
SQL> select FILENUMBER
     , SNAPSHOTFILENAME
     , CLONEFILENAME
  from V$CLONEDFILE;

FILENUMBER SNAPSHOTFILENAME
---------- --------------------------
CLONEFILENAME
--------------------------------------------------------------------------------
16         +DATA/TESTMASTER/09D05108AB70216BE053D6CBF00AA040/DATAFILE/system.257.865863315
+SPARSEDG/JOHNTEST/09D05108AB70216BE053D6CBF00AA041/DATAFILE/system.257.865863315
17         +DATA/TESTMASTER/09D05108AB70216BE053D6CBF00AA040/DATAFILE/sysaux.258.865863317
+SPARSEDG/JOHNTEST/09D05108AB70216BE053D6CBF00AA041/DATAFILE/sysaux.258.865863317
```

Agenda

Exadata Sparse Clones

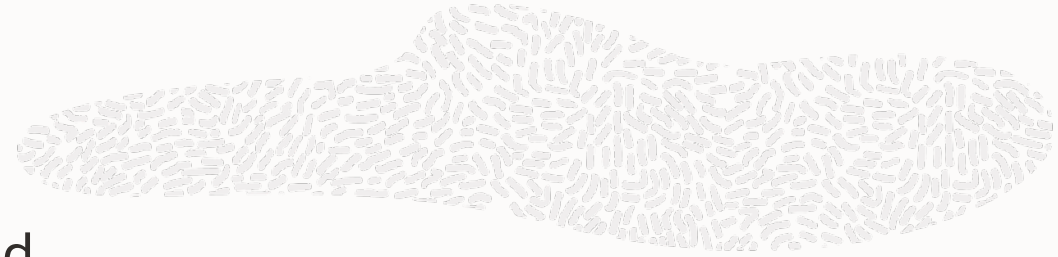
- Features
- Hierarchical Snapshots
- Sparse Test Masters
- Monitoring and Statistics
- Resources

ACFS Snapshots

Summary

Resize operations

Resizing



- Virtual or physical space of sparse can be changed
  - Remember the per disk virtual size limit of 100Tb and physical size limit of 4Tb
- To Modify Virtual Space
  - To increase
    - First alter the cell disks
      CellCLI> alter griddisk SPARSE_CD_00_CELL01,SPARSE_CD_01_CELL01,....,SPARSE_CD_11_CELL01 virtualSize=newBiggerSize;
    - Then alter the disk group in an ASM instance
      SQL> alter diskgroup SPARSE resize all size newBiggerSize;
  - To decrease
    - Ensure you have the free space to reduce virtual size
    - First alter the diskgroup in ASM
      SQL> alter diskgroup SPARSE resize all size newSmallerSize;
    - Then alter the cell disks
      CellCLI> alter griddisk SPARSE_CD_00_CELL01,SPARSE_CD_01_CELL01,....,SPARSE_CD_11_CELL01 virtualSize=newSmallerSize;

Resize operations

Resizing

- Physical space increases may require resizing of other disk groups/grid disks
  - See 3.3.3 Resizing Grid Disks in Oracle Exadata System Software
- Once physical space is available
  - To increase
    - First alter the cell disks
      CellCLI> alter griddisk SPARSE_CD_00_CELL01,SPARSE_CD_01_CELL01,....,SPARSE_CD_11_CELL01 Size=newBiggerSize;
    - No changes need to be done in ASM
  - To decrease
    - Ensure you have the free space to reduce physical size by checking physical usage in ASM
      SQL> SELECT sum(allocated_mat_mb) FROM v$asm_disk_sparse
    - WHERE group_number = group_number_of_diskgrp_to_shrink;
    - Alter the cell disks
      CellCLI> alter griddisk SPARSE_CD_00_CELL01,SPARSE_CD_01_CELL01,....,SPARSE_CD_11_CELL01 Size=newSmallerSize;
    - No changes need to be done in ASM

ASMCMD sparse operations

Sparse File COPY

- Datafiles sometimes need to be copied
  - from one diskgroup to another; OR
  - one hardware to another
- Copy a sparse file in a space efficient manner to a new destination
  - asmcmd> cp --sparse <src_sparse_file_list> <tgt_file_or_dir>
  - Need sparse copy to prevent exploding the file size at the destination
  - Destination file or directory must be on a sparse diskgroup
  - We also copy the OSD header of the source file to the destination file
  - If a sparse copy is done on a local ASM instance; the parent file is set during the copy
  - If a sparse copy is done on a remote ASM instance; parent file must exist in same state and should be explicitly set by the user
- The command accepts a set of input source files and copies it to a destination directory

ASMCMD sparse operations

Sparse File setSparseParent

- When the parent file is moved; copied to a different diskgroup/hardware
  - Must update child file's parent info
- Sets the parent of a sparse file
  - `asmcmd> setsparseparent <sparse_child_file> <parent_file>`
  - Child file must be a sparse file
  - Parent file may be a sparse or non-sparse file
- Parent and child must have a valid relationship
  - child's block 0 information must match with parent's block 1 information
  - Indicates parent is at a precise point in time when the child was created (SCN; timestamp; etc)
- Parent and child files must be on the same ASM instance
- Most common usage after an RMAN restore with SET NEWNAME

Misc Improvements

- Sparse File in block 0 stores information about parent file's block 1 → SCN, timestamp
  - On an open of a sparse file we also open the parent, and at this point we validate the child and parent are still at a valid point
  - Prevents **setsparseparent** from assigning an incorrect parent
  - Protects against opening the child if someone incorrectly wrote to the parent file
- Better errors when attempting a write on a read-only parent file
  - ORA-17528: A read only file or a file opened read only cannot be written to: +DATAFILE/dbs/cdb1_pdb1_ax.f

Release

All Exadata Snapshot features

- Available
  - Database software → 12.2.0.1
  - Grid software → 12.2.0.1
  - Cell software → 12.2.1.1.0
  - RMAN sparse backups → 18.1.0.0
- Recommended 19c or later

Documentation

- Exadata Storage Server Software User's Guide
  - Chapter 10 → Setting up Oracle Exadata Storage Snapshots
- https://docs.oracle.com/en/engineered-systems/exadata-database-machine/sagug/exadata-storage-server-snapshots.html#GUID-78F67DD0-93C8-4944-A8F0-900D910A06A0

Agenda

Exadata Sparse Clones

- Features
- Hierarchical Snapshots
- Sparse Test Masters
- Monitoring and Statistics
- Resources

ACFS Snapshots

Summary

ACFS Snapshots

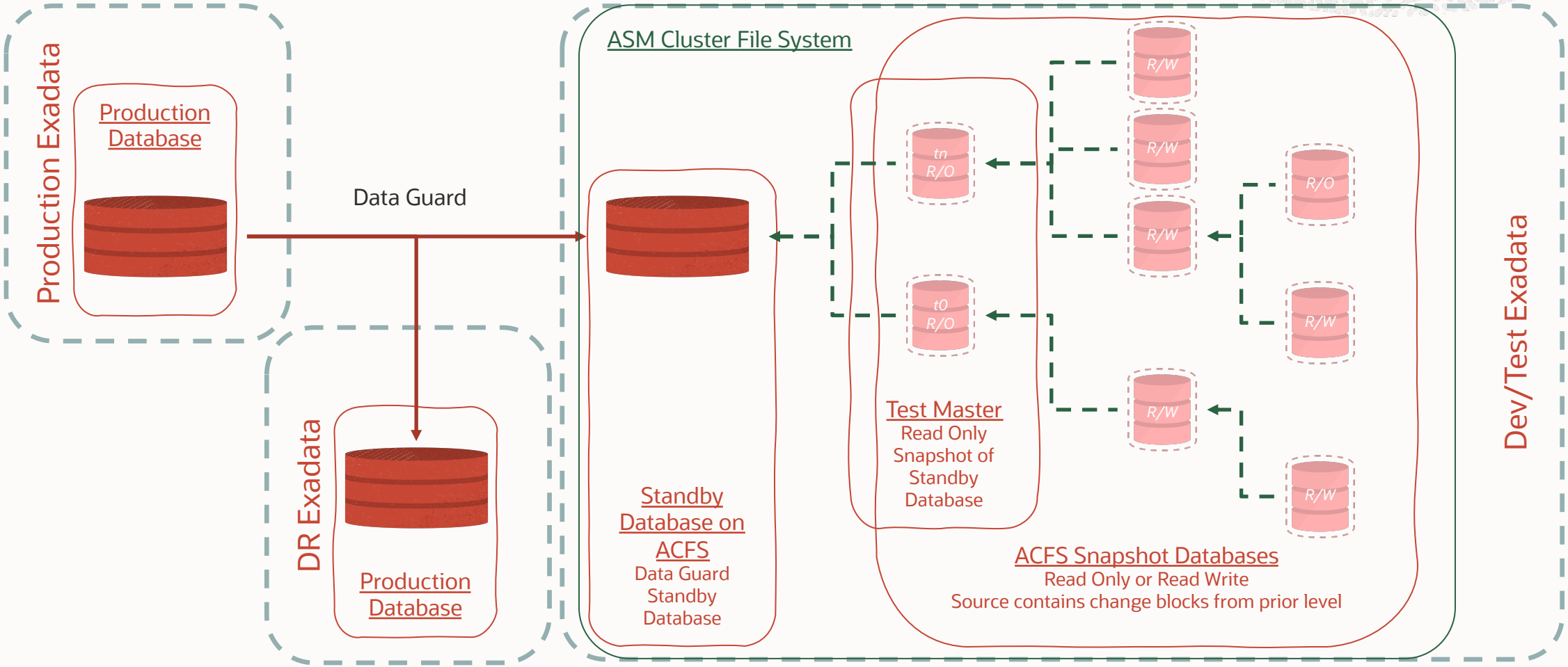
- Prerequisites
  - 19.10 (plus performance & stability patches) or later Grid Infrastructure
  - Set COMPATIBLE.ASM and COMPATIBLE.ADVM to match GI version for latest functionality access
- Benefits
  - ACFS on Exadata exists today
  - Solution has features and functionality similar to third party copy-on-write snapshots
  - Supports database versions 11gR2 (11.2.0.4) and higher
- Limitations
  - No Exadata smart offload features other than Flash Cache
  - ACFS Encryption for Oracle database files is not supported

ACFS Exadata Snapshot Use Cases

- Supports all Oracle databases
  - Singleton PDBs
  - Full non-CDBs
  - Full CDBs
- Primary benefit from read-write test master with multiple timelines
  - Single full copy of source data (e.g. physical standby database)
  - Periodically create a read only snapshot
    - Serves as test master for that timeline
  - Full copy source data continues to be updated
  - Ability to create up to 1023 snapshots

Exadata ACFS Database Snapshots

Conceptual View

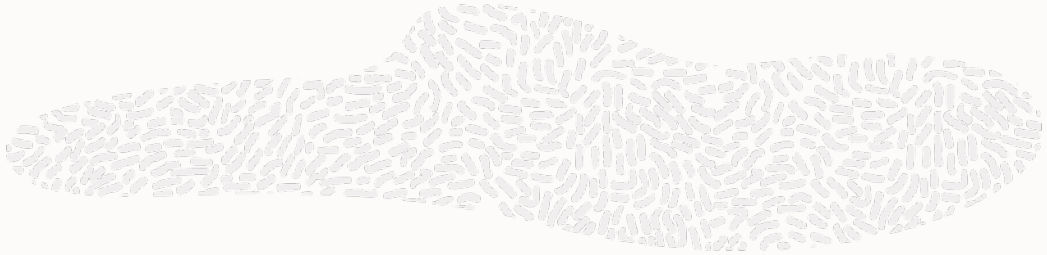


Lifecycle Operations

High Availability and File Placement

- Use ACFS on Exadata for test/dev databases only
- Test Master Database (Standby Database) should not be the same as the DR Standby Database
- Always recommend using high redundancy disk group for best storage protection and high availability during storage rolling updates
- ACFS “DATA” filesystem will be created on +DATA and should contains your database files, online logs, controlfiles and spfile
- ACFS “RECO” filesystem will be created on +RECO and should contains your archive files and flashback logs
- Do not use Exadata ACFS for Oracle Home, diagnostic destination, audit destination or security/encryption wallets
- Do NOT co-locate test/dev databases with production databases on the same RAC or VM cluster

Lifecycle Operations



Backup/Restore

- ACFS Snapshots looks like a normal database file to RMAN instead of a “sparse” database. RMAN not ACFS Snapshot aware.

Software updates

- Same options for Software and Database Updates as for non-ACFS Databases

Creating, Resizing and Dropping ACFS file systems is simple. Refer to Doc ID 2761360.1

ACFS Read/Write Test Master

- Create ACFS Filesystem
- Create a physical standby database on ACFS
  - Use RMAN Duplicate for Standby – OR –
  - gDBClone

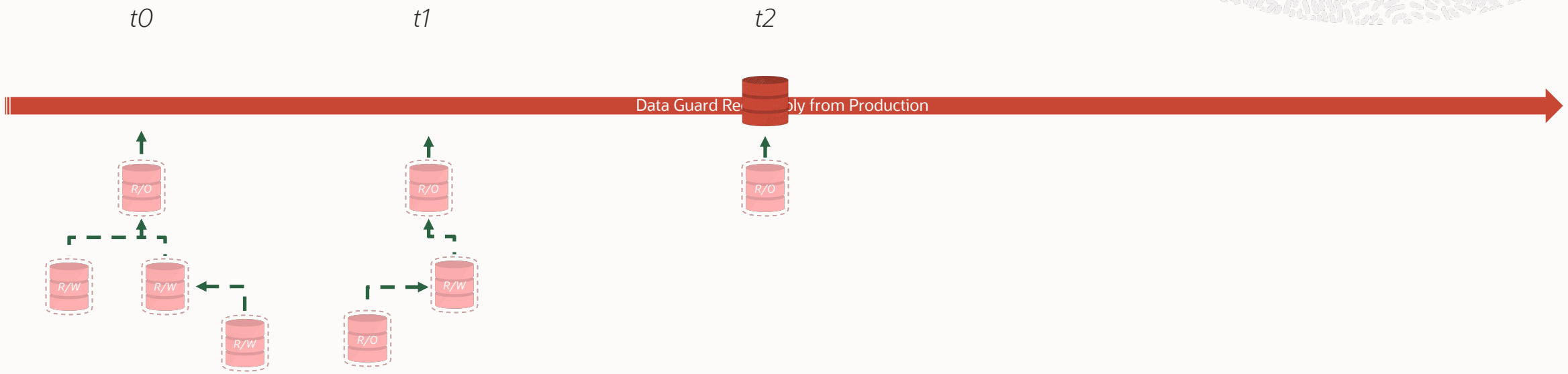
ACFS Read/Write Test Master

- Create first timeline
  - Stop redo apply
  - Create Read-Only ACFS snapshot as the base
  - Restart redo apply
  - Create additional Read-Write or Read-Only ACFS snapshots as required for test use cases

ACFS Read/Write Test Master

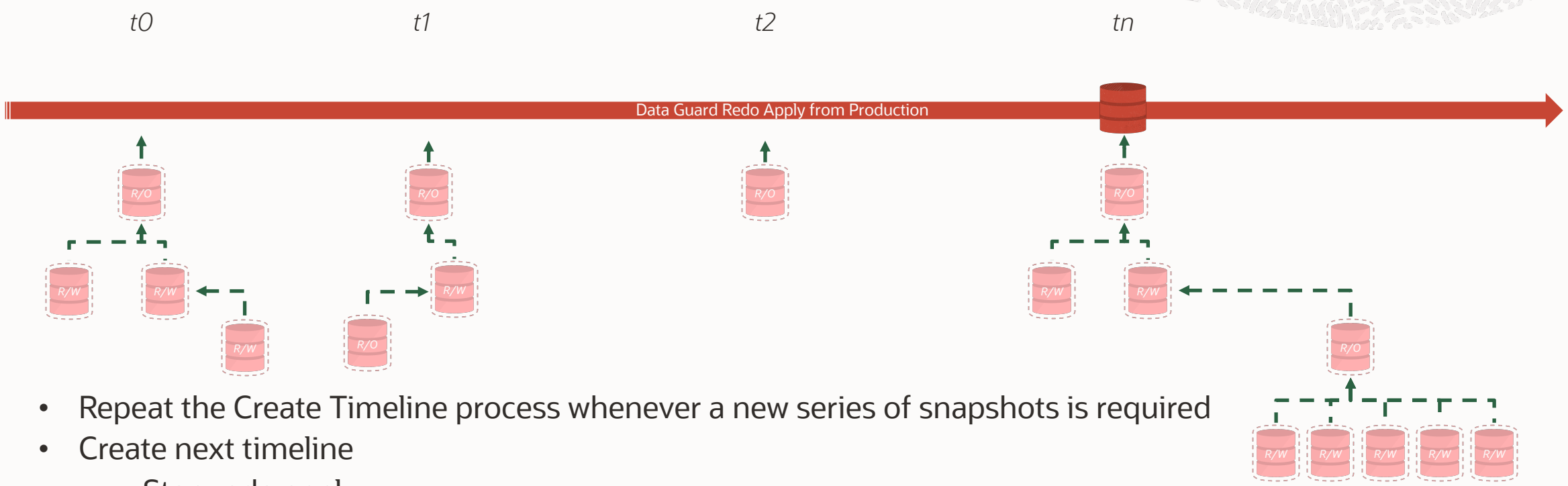
- Repeat the Create Timeline process whenever a new series of snapshots is required
- Create next timeline
  - Stop redo apply
  - Create Read-Only ACFS snapshot as the base
  - Restart redo apply
  - Create additional Read-Write or Read-Only ACFS snapshots as required for test use cases

ACFS Read/Write Test Master



- Repeat the Create Timeline process whenever a new series of snapshots is required
- Create next timeline
  - Stop redo apply
  - Create Read-Only ACFS snapshot as the base
  - Restart redo apply
  - Create additional Read-Write or Read-Only ACFS snapshots as required for test use cases

ACFS Read/Write Test Master



- Repeat the Create Timeline process whenever a new series of snapshots is required
- Create next timeline
  - Stop redo apply
  - Create Read-Only ACFS snapshot as the base
  - Restart redo apply
  - Create additional Read-Write or Read-Only ACFS snapshots as required for test use cases

ACFS Snapshot References

- Oracle ACFS Support on Oracle Exadata Database Machine (Linux only) (Doc ID 1929629.1)
- Oracle ACFS Snapshot Use Cases on Exadata (Doc ID 2761360.1)
- Oracle Automatic Storage Management Cluster File System - Administrator's Guide (https://docs.oracle.com/en/database/oracle/oracle-database/19/ostmg/index.html)
  - Creating an Oracle ACFS File
  - Managing Oracle ACFS Snapshots
  - How to Clone a Master Database with ACFS Snapshots
- Oracle System Software – User's Guide for Exadata Sparse to compare (https://docs.oracle.com/en/engineered-systems/exadata-database-machine/sagug/index.html)
  - Setting up Oracle Exadata Storage Snapshots

Agenda

Exadata Sparse Clones

- Features
- Hierarchical Snapshots
- Sparse Test Masters
- Monitoring and Statistics
- Resources

ACFS Snapshots

Summary

Exadata Database Clones & Snapshots

Feature/Requirement	Exadata Sparse Clones	ACFS Snapshots on Exadata
Simple to Use	Yes	Yes
Exadata Performance Features	All Exadata Features available	Exadata Smart Flash Cache
Space Efficient Dev/Test Database Clones	Yes	Yes
CDB & PDB Support	Yes	Yes
Non-CDB Support	Yes	Yes
Enterprise Manager Support	Yes	No
Hierarchical Snapshots (aka Snapshot of snapshot)	Yes	Yes
Space efficient backups	Yes	No

Thank you


ORACLE

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.