Exadata Database Machine Administration Workshop

Activity Guide – Volume I

D73668GC21
Edition 2.1
February 2014
D85492

ORACLE

Author

Peter Fusek

Technical Contributors and Reviewers

Akshay Shah, Alex Tsukerman, Amit Ganesh, Andrew Babb, Aslam Edah-Tally, Barb Lundhild, Bharat Baddepudi, Bill Hodak, Boris Erlikhman, Branislav Valny, Bruce Kyro, Caroline Johnston, Christian Craft, Dan Norris, Darryl Presley, Dave Winter, David Hernandez Mendoza, David Hitchcock, Deba Chatterjee, Douglas Utzig, Ed Gilowski, Eric Siglin, Georg Schmidt, Harald van Breederode, Ira Singer, James He, James Womack, Jean-Francois Verrier, Jia Shi,Jignesh Patel, Jim Hall, Jim Spiller, Jim Viscusi, Joel Goodman, Juan Loaiza, Kam Shergill,Kevin Jernigan, Kodi Umamageswaran, Krishnanjani Chitta, Lachlan Williams, Larry Justice, Lawrence To, Louis Nagode, Mahesh Subramaniam, Maria Billings, Maria Colgan, Mark Fuller, Mark Scardina, Mark Van de Wiel, Marshall Presser, Martin Jensen, Michael Cebulla, Michael Nowak, Naoki Kato, Nilesh Choudhury, Ravindra Dani, Raymond Dutcher, Richard Exley, Robert Carlin, Robert Pastijn, Roger Hansen, Sabyasachi Banerjee, Sean Kim, Selcuk Aya, Scott Gossett, Sriram Palapudi, Steven Lemme, Sue Lee, Sugam Pandey, Sumeet Lahorani, Sundararaman Sridharan, Tim Shelter, Umesh Panchaksharaiah, Uwe Hesse, Varun Malhotra, Vern Wagman, Vijay Sridharan, Vikram Kapoor, Vimala Jacob

This book was published using: oracle*tutor*

Table of Contents

Practices for Lesson 1: Introduction

Chapter 1

Practices for Lesson 1

Practices Overview

In this practice, you will be introduced to the laboratory environment used to support all the practices during this course.

Practice 1-1: Lab Environment Introduction

Overview

In this practice, you will learn how to use the laboratory environment that supports all the practices in this course.

The laboratory environment for this course is based on a Quarter Rack Database Machine. It consists of several virtual machines configured to provide an Oracle Database server and three Exadata Storage Servers. The second Oracle Database server, which is normally found on a Quarter Rack Database Machine, is initially not started due to physical resource constraints.

To access the virtual machines, you will first establish a graphical session which is connected to the VM server. Your instructor will provide specific details for each student's server. From there, you will create terminal sessions as required and connect to the virtualized Database Machine servers by using SSH as described in the tasks for this practice.

Tasks

1. Establish a terminal session connected to the `qr01db01` database server by using the `grid` operating system account.

   Note that you may see additional messages relating to server identities. Answer `yes` if you are prompted to acknowledge server authenticity.

```
$ ssh grid@qr01db01
grid@qr01db01's password: <oracle>
[grid@qr01db01 ~]$
```

2. Execute the following command and verify that all of the listed services are online in your laboratory environment. Your output should look like the example below. Alert your instructor if you do not have the same services online in your environment.

```
[grid@qr01db01 ~]$ crsctl status resource -w "TARGET = ONLINE" -t
--------------------------------------------------------------------------------
NAME           TARGET  STATE        SERVER                   STATE_DETAILS
--------------------------------------------------------------------------------
Local Resources
--------------------------------------------------------------------------------
ora.DATA_QR01.dg
               ONLINE  ONLINE       qr01db01
ora.DBFS_DG.dg
               ONLINE  ONLINE       qr01db01
ora.LISTENER.lsnr
               ONLINE  ONLINE       qr01db01
ora.RECO_QR01.dg
               ONLINE  ONLINE       qr01db01
ora.asm
               ONLINE  ONLINE       qr01db01                 Started
ora.net1.network
               ONLINE  ONLINE       qr01db01
ora.ons
               ONLINE  ONLINE       qr01db01
--------------------------------------------------------------------------------
Cluster Resources
```

```
--------------------------------------------------------------------------------
ora.LISTENER_SCAN1.lsnr
      1        ONLINE  ONLINE       qr01db01
ora.LISTENER_SCAN2.lsnr
      1        ONLINE  ONLINE       qr01db01
ora.LISTENER_SCAN3.lsnr
      1        ONLINE  ONLINE       qr01db01
ora.cvu
      1        ONLINE  ONLINE       qr01db01
ora.dbm.db
      1        ONLINE  ONLINE       qr01db01                 Open
      2        ONLINE  OFFLINE
ora.oc4j
      1        ONLINE  ONLINE       qr01db01
ora.qr01db01.vip
      1        ONLINE  ONLINE       qr01db01
ora.qr01db02.vip
      1        ONLINE  INTERMEDIATE qr01db01                 FAILED OVER
ora.scan1.vip
      1        ONLINE  ONLINE       qr01db01
ora.scan2.vip
      1        ONLINE  ONLINE       qr01db01
ora.scan3.vip
      1        ONLINE  ONLINE       qr01db01
[grid@qr01db01 ~]$
```

3. Using SQL*Plus, connect to the clustered ASM environment as an ASM administrator. Verify that you are connected to the ASM instance `+ASM1` and exit your SQL*Plus session.

```
[grid@qr01db01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 11.2.0.3.0 Production...

SQL> select instance_name from v$instance;

INSTANCE_NAME
----------------
+ASM1

SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - 64bit Production...
[oracle@qr01db01 ~]$
```

4. Establish a new shell as the `oracle` OS user.

```
[grid@qr01db01 ~]$ su - oracle
Password: <oracle>
[oracle@qr01db01 ~]$
```

5. Using SQL*Plus, connect to your database as the database administrator. Verify that you are connected to the `DBM` database and exit your SQL*Plus session.

```
[oracle@qr01db01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 11.2.0.3.0 Production...

SQL> select name from v$database;

NAME
---------
DBM

SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - 64bit Production...
[oracle@qr01db01 ~]$
```

6. Use the `srvctl` utility to verify the status of the `DBM` database.

```
[oracle@qr01db01 ~]$ srvctl status database -d dbm
Instance dbm1 is running on node qr01db01
Instance dbm2 is not running on node qr01db02
[oracle@qr01db01 ~]$
```

Note that the `DBM` database is an administrator-managed Oracle RAC database created on two database servers; `qr01db01` and `qr01db02`. As mentioned previously, you will use the database instance running on `qr01db01` to perform most the practices. The other database server (`qr01db02`) will not be started for most of the practices to avoid unnecessary resource consumption in the laboratory environment.

7. Many practices refer to SQL scripts, which are provided as an alternative to typing lengthy commands. These scripts are located in the `labs` directory under your student home directory. List the files in the `labs` directory. Notice also the `CSV`, `TTS` and `Exaconf` subdirectories. These subdirectories contain additional files for specific practices. You will be directed to these files in the associated practice instructions.

```
[oracle@qr01db01 ~]$ cd labs
[oracle@qr01db01 labs]$ ls
CSV              lab04-04-03.sql  lab06-03-07.sql  lab11-01-15.sql
Exaconf          lab04-04-04.sql  lab06-03-08.sql  lab12-01-22.sql
lab03-02-03.sql  lab06-02-03.sql  lab06-04-21.sql  lab15-01-33.sql
lab03-03-11.sql  lab06-02-04.sql  lab08-02-04.sql  lab15-01-35.sql
lab03-03-12.sql  lab06-02-06.sql  lab08-02-06.sql  lab15-01-36.sql
lab04-01-03.sql  lab06-02-09.sql  lab08-02-07.sql  lab20-02-10.sql
lab04-01-04.sql  lab06-02-18.sql  lab08-02-15.sql  lab20-02-11.sql
```

```
lab04-01-08.sql  lab06-02-28.sql  lab09-01-06.sql  lab20-02-16.sql
lab04-02-03.sql  lab06-02-38.sql  lab09-01-07.sql  lab20-03-13.sql
lab04-02-04.sql  lab06-03-03.sql  lab09-02-07.sql  TTS
lab04-03-05.sql  lab06-03-04.sql  lab09-02-08.sql
lab04-03-08.sql  lab06-03-05.sql  lab09-02-09.sql
lab04-03-15.sql  lab06-03-06.sql  lab11-01-05.sql
[oracle@qr01db01 labs]$
```

8. Establish a terminal session connected to your first Exadata cell using the `celladmin` user. Confirm that you are connected to the cell and then exit the session.

   Note that you may see additional messages relating to server identities. Answer `yes` if you are prompted to acknowledge server authenticity.

```
$ ssh celladmin@qr01cel01
celladmin@qr01cel01's password: <welcome>
[celladmin@qr01cel01 ~]$ cellcli -e list cell
         qr01cel01       online
[celladmin@qr01cel01 ~]$ exit
logout
Connection to qr01cel01 closed.
$
```

9. Establish a terminal session connected to your second Exadata cell using the `cellmonitor` user. Confirm you are connected to the cell and then exit the session.

   Note that you may see additional messages related to server identities. Answer `yes` if you are prompted to acknowledge server authenticity.

```
$ ssh cellmonitor@qr01cel02
cellmonitor@qr01cel02's password: <welcome>
[cellmonitor@qr01cel02 ~]$ cellcli -e list cell
         qr01cel02       online
[cellmonitor@qr01cel02 ~]$ exit
logout
Connection to qr01cel02 closed.
$
```

10. Exit your terminal sessions. We recommend that you start fresh terminal sessions at the beginning of each practice, and that you exit all of your terminal sessions at the conclusion of every practice. This eliminates the possibility that environment settings used in one practice could cause problems in following practices.

Practices for Lesson 2: Exadata Database Machine: Overview

Chapter 2

Practices for Lesson 2

Practices Overview

There is no practice for Lesson 2.

Practices for Lesson 3: Exadata Database Machine Architecture

Chapter 3

Practices for Lesson 3

Practices Overview

In these practices, you will be familiarized with the Exadata cell architecture. You will:

- Examine the Exadata processes
- Exercise Exadata high availability
- Examine the hierarchy of cell objects
- Examine Exadata Smart Flash Cache

Practice 3-1: Process Familiarization

Overview

In this practice, you will examine the Exadata cell software processes.

Tasks

1. Establish a terminal connection to `qr01cel01` as the `celladmin` user.
2. Restart Server (`RS`) is used to start up and shut down the Cell Server (`CELLSRV`) and Management Server (`MS`). It also monitors these services to check whether they need to be restarted. Locate the `RS` processes by using the following command:

```
[celladmin@qr01cel01 ~]$ ps -ef | grep cellrs
root      2025     1  0 18:29 ?        00:00:00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellrssrm -ms 1 -
cellsrv 1
root      2032  2025  0 18:29 ?        00:00:00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellrsmmt -ms 1 -
cellsrv 1
root      2033  2025  0 18:29 ?        00:00:00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellrsbmt -ms 1 -
cellsrv 1
root      2035  2033  0 18:29 ?        00:00:00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellrsbkm -rs_conf
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/config/cellinit.ora
-ms_conf
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/config/cellrsms.sta
te -cellsrv_conf
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/config/cellrsos.sta
te -debug 0
root      2036  2025  0 18:29 ?        00:00:01
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellrsomt -ms 1 -
cellsrv 1
root      2044  2035  0 18:29 ?        00:00:00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellrssmt -rs_conf
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/config/cellinit.ora
-ms_conf
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/config/cellrsms.sta
te -cellsrv_conf
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/config/cellrsos.sta
te -debug 0
1000      3609  3577  0 19:02 pts/0    00:00:00 grep cellrs
[celladmin@qr01cel01 ~]$
```

3. `MS` provides Exadata cell management and configuration. It works in cooperation with the Exadata cell command-line interface (CellCLI). In addition, `MS` is responsible for sending alerts and collects some statistics in addition to those collected by `CELLSRV`. Locate the `MS` process by using the following command:

```
[celladmin@qr01cel01 ~]$ ps -ef | grep ms.err
root      2034  2032  1 18:29 ?        00:00:30 /usr/java/jdk1.5.0_15/bin/java
-Xms256m -Xmx512m -
Djava.library.path=/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/lib -
Ddisable.checkForUpdate=true -jar
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/oc4j/ms/j2ee/home/oc4j.jar -out
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/log/ms.lst -err
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/log/ms.err
1000      3623  3577  0 19:02 pts/0    00:00:00 grep ms.err
[celladmin@qr01cel01 ~]$
```

4. Locate the MS parent process. Use the parent process number associated with MS in the output for step 3. Note that RS spawns (and, when required, re-spawns) MS.

```
[celladmin@qr01cel01 ~]$ ps -ef | grep 2032
root      2032  2025  0 18:29 ?        00:00:00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellrsmmt -ms 1 -
cellsrv 1
root      2034  2032  1 18:29 ?        00:00:31 /usr/java/jdk1.5.0_15/bin/java
-Xms256m -Xmx512m -
Djava.library.path=/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/lib -
Ddisable.checkForUpdate=true -jar
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/oc4j/ms/j2ee/home/oc4j.jar -out
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/log/ms.lst -err
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/deploy/log/ms.err
1000      3625  3577  0 19:03 pts/0    00:00:00 grep 2032
[celladmin@qr01cel01 ~]$
```

5. CELLSRV is the primary Exadata software component and provides the majority of Exadata storage services. CELLSRV is a multithreaded server. Primarily, CELLSRV communicates with Oracle Database to serve simple block requests, such as database buffer cache reads, and Smart Scan requests, such as table scans with projections and filters. CELLSRV also implements I/O Resource Management (IORM) and collects numerous statistics relating to its operations. Locate the CELLSRV process by using the following command:

```
[celladmin@qr01cel01 ~]$ ps -ef | grep "/cellsrv "
root      2037  2036 12 18:29 ?        00:04:28
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellsrv 25 1200 9 5042
1000      3652  3577  0 19:04 pts/0    00:00:00 grep /cellsrv
[celladmin@qr01cel01 ~]$
```

6. Locate the CELLSRV parent process. Use the parent process number associated with CELLSRV in the output for step 5. Note that RS spawns (and, when required, re-spawns) CELLSRV.

```
[celladmin@qr01cel01 ~]$ ps -ef | grep 2036
root      2036  2025  0 18:29 ?        00:00:01
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellrsomt -ms 1 -
cellsrv 1
root      2037  2036 13 18:29 ?        00:04:32
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellsrv 25 1200 9 5042
1000      3654  3577  0 19:04 pts/0    00:00:00 grep 2036
[celladmin@qr01cel01 ~]$
```

7. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

8. Execute the following CellCLI command to examine the attributes of the cell. Note that the output also confirms that `CELLSRV`, `MS`, and `RS` are currently up and running. Exit CellCLI after examining the cell attributes.

```
CellCLI> list cell detail
         name:                   qr01cel01
         bbuTempThreshold:       60
         bbuChargeThreshold:     800
         bmcType:                absent
         cellVersion:            OSS_11.2.3.2.1_LINUX.X64_130109
         cpuCount:               1
         diagHistoryDays:        7
         fanCount:               1/1
         fanStatus:              normal
         flashCacheMode:         WriteThrough
         id:                     8ab50138-a667-4793-a976-c540dc1930c5
         interconnectCount:      3
         interconnect1:          eth1
         iormBoost:              0.0
         ipaddress1:             192.168.1.103/24
         kernelVersion:          2.6.32-400.11.1.el5uek
         makeModel:              Fake hardware
         metricHistoryDays:      7
         offloadEfficiency:      663.5
         powerCount:             1/1
         powerStatus:            normal
         releaseVersion:         11.2.3.2.1
         releaseTrackingBug:     14522699
         status:                 online
         temperatureReading:     0.0
         temperatureStatus:      normal
         upTime:                 0 days, 0:36
         cellsrvStatus:          running
         msStatus:               running
         rsStatus:               running

CellCLI> exit
quitting

[celladmin@qr01cel01 ~]$
```

9. Exit all of your terminal sessions.

Practice 3-2: Exadata High Availability

Overview

In this practice, you will observe some of the high-availability features of Exadata Storage Server.

Tasks

1. Establish a terminal connection to `qr01db01` as the `oracle` user.
2. Connect to your database with SQL*Plus. Log in as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Execute the SQL script `/home/oracle/labs/lab03-02-03.sql`. The script contains a series of I/O intensive queries, which are used in this practice to demonstrate how Oracle Database is insulated from the different Exadata failure scenarios that are demonstrated in the practice. Check on the workload periodically throughout the practice. If the workload completes before you finish all the tasks, then simply re-execute the script to maintain an active workload throughout the practice.

```
SQL> @/home/oracle/labs/lab03-02-03
SQL> set timing on
SQL> select count(*) from sales;

...
```

4. Establish a separate terminal connection to the `qr01cel01` Exadata cell as the `root` user. The `root` password is `oracle`. Leave your SQL*Plus terminal session and workload running in the background.
5. Locate the process identification number for `CELLSRV` by using the following `ps` command.

```
[root@qr01cel01 ~]# ps -ef | grep "/cellsrv "
root      2037  2036 12 18:29 ?        00:04:28
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellsrv 25 1200 9 5042
root      3752  3677  0 19:06 pts/0    00:00:00 grep /cellsrv
[root@qr01cel01 ~]#
```

6. Terminate the `CELLSRV` process by using the `kill` command and the process identification number you observed in step 5.

```
[root@qr01cel01 ~]# kill -9 2037
[root@qr01cel01 ~]#
```

7. Re-execute the `ps` command from step 5. You should observe that `CELLSRV` is automatically restarted with a new process identification number. How was `CELLSRV` restarted?

```
[root@qr01cel01 ~]# ps -ef | grep "/cellsrv "
root      3801  3800  3 19:07 ?        00:00:00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellsrv 25 1200 9 5042
root      3852  3735  0 19:07 pts/0    00:00:00 grep /cellsrv
[root@qr01cel01 ~]#
```

8. Check on the progress of your workload from step 3. You should observe that the workload continues without error.

```
...
SQL> select count(*) from sales where amount_sold > 2;

  COUNT(*)
----------
  14693419

Elapsed: 00:00:25.72
SQL> select count(*) from sales where amount_sold > 3;

  COUNT(*)
----------
  14540531

Elapsed: 00:00:18.79
SQL> select count(*) from sales where amount_sold > 4;
```

9. On `qr01cel01`, launch the Exadata cell command-line interface (CellCLI).

```
[root@qr01cel01 ~]# cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

10. You have already seen how Exadata automatically recovers from an unexpected process failure. Now observe the effect of restarting all the Exadata services.

```
CellCLI> alter cell restart services all

Stopping the RS, CELLSRV, and MS services...
The SHUTDOWN of services was successful.
Starting the RS, CELLSRV, and MS services...
Getting the state of RS services...  running
Starting CELLSRV services...
The STARTUP of CELLSRV services was successful.
Starting MS services...
The STARTUP of MS services was successful.

CellCLI>
```

11. Check on the progress of your workload from step 3. You should again observe that the workload continues without error.

```
...
SQL> select count(*) from sales where amount_sold > 7;

  COUNT(*)
----------
  13929029

Elapsed: 00:00:24.37
SQL> select count(*) from sales where amount_sold > 8;

  COUNT(*)
----------
  13776154

Elapsed: 00:00:18.29
SQL> select count(*) from sales where amount_sold > 9;
```

12. Exit your CellCLI session and re-execute the `ps` command from step 5. You should observe that `CELLSRV` has a different process identification number because of the restart operation executed in step 10.

```
CellCLI> exit
quitting

[root@qr01cel01 ~]# ps -ef | grep "/cellsrv "
root      4522  4519  9 19:13 ?        00:00:19
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/cellsrv/bin/cellsrv 25 1200 9 5042
root      4996  3735  0 19:16 pts/0    00:00:00 grep /cellsrv
[root@qr01cel01 ~]#
```

13. If the workload started in step 3 is still executing, stop it by typing `<Control>-C` in your SQL*Plus session window. Wait until the workload stops.

```
...
SQL> select count(*) from sales where amount_sold > 10;

  COUNT(*)
----------
  13469722

Elapsed: 00:00:18.05
SQL> select count(*) from sales where amount_sold > 11;
^Cselect count(*) from sales where amount_sold > 11
                           *
ERROR at line 1:
ORA-01013: user requested cancel of current operation


Elapsed: 00:00:09.30

SQL>
```

14. On `qr01cel01`, launch the Exadata cell command-line interface (CellCLI) again.

```
[root@qr01cel01 ~]# cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

15. Execute the `LIST ALERTHISTORY` command. You should observe an alert indicating that `CELLSRV` terminated unexpectedly. This alert relates to the process failure brought about by killing `CELLSRV` in step 6. No alert appears for the controlled service restart executed in step 10. If you don't see the alert, re-execute `LIST ALERTHISTORY` periodically until the alert appears.

```
CellCLI> list alerthistory
         1_1     2013-07-17T18:31:57-04:00       warning
"Hugepage allocation failure in service cellsrv.  Number of
Hugepages allocated is 0, failed to allocate 110"
         2       2013-07-17T19:07:31-04:00       critical
"RS-7445 [Serv CELLSRV is absent] [It will be restarted] [] []
[] [] [] [] [] [] [] []"

CellCLI>
```

Note that the “Hugepage allocation failure” alert is peculiar to the laboratory environment. You shouldn’t see this alert in a production environment.

16. Exit all of your terminal sessions.

Practice 3-3: Storage Object Familiarization

Overview

In this practice, you are introduced to the hierarchy of Exadata storage objects.

Tasks

1. Establish a terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.
2. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

3. In Exadata, a LUN (Logical Unit) is a logical abstraction of a storage device. LUNs are based on hard disks and flash devices. LUNs are automatically created when Exadata is initially configured. Each Exadata cell contains 12 hard disk-based LUNs along with 16 flash-based LUNs. List the LUNs on your primary Exadata cell.

```
CellCLI> list lun
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK00   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK01
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK01   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK02
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK02   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK03
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK03   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK04
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK04   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK05
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK05   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK06
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK06   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK07
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK07   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK08
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK08   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK10
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK10   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK11
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK11   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH00  normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH01
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH01  normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH02
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH02  normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH03
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH03  normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH04
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH04  normal
```

```
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH05
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH05   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH06
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH06   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH07
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH07   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH08
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH08   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH09
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH09   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH10
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH10   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH11
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH11   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH12
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH12   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH13
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH13   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH14
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH14   normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH15
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/FLASH15   normal

CellCLI>
```

Note that the output from the virtualized Exadata cell shows LUNs with names and identifiers that are paths to virtualized disks and virtualized flash devices. On a real Exadata cell, the LUN names and identifiers are based on the PCI slot number and device number of the hard disk or flash device. For example, here is the expected output for the `LIST LUN` command on a real Exadata cell.

```
CellCLI> list lun
         0_0     0_0     normal
         0_1     0_1     normal
         0_2     0_2     normal
         0_3     0_3     normal
         0_4     0_4     normal
         0_5     0_5     normal
         0_6     0_6     normal
         0_7     0_7     normal
         0_8     0_8     normal
         0_9     0_9     normal
         0_10    0_10    normal
         0_11    0_11    normal
         1_0     1_0     normal
         1_1     1_1     normal
         1_2     1_2     normal
         1_3     1_3     normal
         2_0     2_0     normal
         2_1     2_1     normal
         2_2     2_2     normal
         2_3     2_3     normal
         4_0     4_0     normal
         4_1     4_1     normal
```

```
         4_2     4_2     normal
         4_3     4_3     normal
         5_0     5_0     normal
         5_1     5_1     normal
         5_2     5_2     normal
         5_3     5_3     normal
```

4. List only the hard disk-based LUNs.

```
CellCLI> list lun where disktype = harddisk
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK00
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK00    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK01
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK01    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK02
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK02    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK03
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK03    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK04
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK04    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK05
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK05    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK06
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK06    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK07
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK07    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK08
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK08    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK10
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK10    normal
         /opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK11
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK11    normal

CellCLI>
```

5. Examine the detailed attribute listing for the LUN whose name ends with `DISK09`. Note the attribute setting `isSystemLun=FALSE`. This indicates that the LUN is not located on a system disk. Notice also that the LUN is associated with one physical disk and one cell disk.

```
CellCLI> list lun where name like '.*DISK09' detail
         name:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         cellDisk:               CD_09_qr01cel01
         deviceName:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         diskType:               HardDisk
         id:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         isSystemLun:            FALSE
         lunAutoCreate:          FALSE
         lunSize:                11
         physicalDrives:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         raidLevel:              "RAID 0"
         status:                 normal

CellCLI>
```

6. Exadata maintains the physical attributes of each hard disk in a `physicaldisk` object. A `physicaldisk` object is automatically created for each hard disk. Examine the attributes for the hard disk associated with LUN you examined in the previous step.

```
CellCLI> list physicaldisk where luns like '.*DISK09' detail
         name:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         diskType:               HardDisk
         luns:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         physicalInsertTime:     2013-02-24T21:12:47-05:00
         physicalSize:           11
         status:                 normal

CellCLI>
```

7. A cell disk is a higher-level storage abstraction. Each cell disk is based on a LUN and contains additional attributes and metadata. Examine the attributes for the cell disk-based on the LUN you examined in step 5.

```
CellCLI> list celldisk CD_09_qr01cel01 detail
         name:                   CD_09_qr01cel01
         comment:
         creationTime:           2013-02-28T16:31:52-05:00
         deviceName:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         devicePartition:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         diskType:               HardDisk
         errorCount:             0
         freeSpace:              0
         id:                     ed8be9ac-5851-4e03-aacb-c5eec33156fa
         interleaving:           none
         lun:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         physicalDisk:
/opt/oracle/cell11.2.3.2.1_LINUX.X64_130109/disks/raw/DISK09
         raidLevel:              "RAID 0"
         size:                   1.5625G
         status:                 normal

CellCLI>
```

8. A grid disk defines an area of storage on a cell disk. Grid disks are consumed by ASM and are used as the storage for ASM disk groups. Each cell disk can contain a number of grid disks. Examine the grid disks associated with the cell disk you examined in the previous step. Note the names and sizes of the grid disks.

```
CellCLI> list griddisk where celldisk=CD_09_qr01cel01 detail
         name:                   DATA_QR01_CD_09_qr01cel01
         asmDiskgroupName:       DATA_QR01
         asmDiskName:            DATA_QR01_CD_09_QR01CEL01
         asmFailGroupName:       QR01CEL01
         availableTo:
         cachingPolicy:          default
         cellDisk:               CD_09_qr01cel01
         comment:
         creationTime:           2013-02-28T16:32:10-05:00
         diskType:               HardDisk
         errorCount:             0
         id:                     9a735142-8a0f-4798-a83e-814ceb7f9ab5
         offset:                 208M
         size:                   592M
         status:                 active

         name:                   DBFS_DG_CD_09_qr01cel01
         asmDiskgroupName:       DBFS_DG
         asmDiskName:            DBFS_DG_CD_09_QR01CEL01
         asmFailGroupName:       QR01CEL01
         availableTo:
         cachingPolicy:          default
         cellDisk:               CD_09_qr01cel01
         comment:
         creationTime:           2013-02-28T16:32:03-05:00
         diskType:               HardDisk
         errorCount:             0
         id:                     dfb7dcda-f9fa-404b-9239-0f03e3cea480
         offset:                 48M
         size:                   160M
         status:                 active

         name:                   RECO_QR01_CD_09_qr01cel01
         asmDiskgroupName:       RECO_QR01
         asmDiskName:            RECO_QR01_CD_09_QR01CEL01
         asmFailGroupName:       QR01CEL01
         availableTo:
         cachingPolicy:          default
         cellDisk:               CD_09_qr01cel01
         comment:
         creationTime:           2013-02-28T16:32:24-05:00
         diskType:               HardDisk
         errorCount:             0
         id:                     7e028f52-604d-4189-8c2c-dafcd01b8f17
```

```
         offset:                 800M
         size:                   800M
         status:                 active

CellCLI>
```

9. Establish a terminal connection to `qr01db01` as the `grid` user.
10. Using SQL*Plus, connect to ASM as `sysasm`.

```
[grid@qr01db01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

11. Locate the grid disks from step 8 inside ASM (use the SQL script `/home/oracle/labs/lab03-03-11.sql` if you prefer). Check that the sizes reported by ASM match the grid disk attributes reported in step 8. Note the capitalization of the value in the `like` string.

```
SQL> select name, path, state, total_mb from v$asm_disk
2>   where name like '%_CD_09_QR01CEL01';

NAME
------------------------------
PATH
------------------------------------------------------------
STATE    TOTAL_MB
-------- ----------
RECO_QR01_CD_09_QR01CEL01
o/192.168.1.103/RECO_QR01_CD_09_qr01cel01
NORMAL          800

DATA_QR01_CD_09_QR01CEL01
o/192.168.1.103/DATA_QR01_CD_09_qr01cel01
NORMAL          592

DBFS_DG_CD_09_QR01CEL01
o/192.168.1.103/DBFS_DG_CD_09_qr01cel01
NORMAL          160


SQL>
```

12. Determine which ASM disk group the grid disks from step 8 are assigned to (use the SQL script `/home/oracle/labs/lab03-03-12.sql` if you prefer). Note the capitalization of the value in the `like` string.

```
SQL> select d.name disk, dg.name diskgroup
2>   from v$asm_disk d, v$asm_diskgroup dg
3>   where dg.group_number = d.group_number
4>   and d.name like '%_CD_09_QR01CEL01';

DISK                           DISKGROUP
------------------------------ ------------------------------
RECO_QR01_CD_09_QR01CEL01      RECO_QR01
DATA_QR01_CD_09_QR01CEL01      DATA_QR01
DBFS_DG_CD_09_QR01CEL01        DBFS_DG

SQL>
```

13. Exit all your SQL*Plus and CellCLI sessions.

Practice 3-4: Exadata Smart Flash Cache Familiarization

Overview

In this practice, you are introduced to Exadata Smart Flash Cache.

Tasks

1. Establish a terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.
2. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~] $ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

3. List the cell disks associated with the flash disk modules in your Exadata cell. By default, there should be 16 cell disks having names that start with `FD`.

```
CellCLI> list celldisk where disktype=flashdisk
         FD_00_qr01cel01         normal
         FD_01_qr01cel01         normal
         FD_02_qr01cel01         normal
         FD_03_qr01cel01         normal
         FD_04_qr01cel01         normal
         FD_05_qr01cel01         normal
         FD_06_qr01cel01         normal
         FD_07_qr01cel01         normal
         FD_08_qr01cel01         normal
         FD_09_qr01cel01         normal
         FD_10_qr01cel01         normal
         FD_11_qr01cel01         normal
         FD_12_qr01cel01         normal
         FD_13_qr01cel01         normal
         FD_14_qr01cel01         normal
         FD_15_qr01cel01         normal

CellCLI>
```

4. By default, Exadata Smart Flash Cache is configured across all the flash-based cell disks. Use the `LIST FLASHCACHE DETAIL` command to confirm that Exadata Smart Flash Cache is configured on your flash-based cell disks. Note that the size of the Exadata Smart Flash Cache on your laboratory cells is much smaller than what you would observe on a real cell; however, all of the other attributes would be similar on a real cell.

```
CellCLI> list flashcache detail
         name:                   qr01cel01_FLASHCACHE
         cellDisk:
FD_01_qr01cel01,FD_13_qr01cel01,FD_07_qr01cel01,FD_06_qr01cel01,FD_09_qr01cel0
1,FD_11_qr01cel01,FD_02_qr01cel01,FD_03_qr01cel01,FD_05_qr01cel01,FD_04_qr01ce
l01,FD_15_qr01cel01,FD_14_qr01cel01,FD_08_qr01cel01,FD_00_qr01cel01,FD_10_qr01
cel01,FD_12_qr01cel01
         creationTime:           2013-03-08T02:18:12-05:00
         degradedCelldisks:
         effectiveCacheSize:     3G
         id:                     c0dca501-f09a-46f0-b504-ec26f23def79
         size:                   3G
         status:                 normal

CellCLI>
```

5. In addition to Exadata Smart Flash Cache, Exadata Smart Flash Log provides a mechanism for improving the latency of database redo log write operations. Exadata Smart Flash Log uses a small portion of high-performance flash memory as temporary storage to facilitate low latency redo log writes. By default, Exadata Smart Flash Log uses 32 MB on each flash-based cell disk, for a total of 512 MB on each Exadata Storage Server. Use the `LIST FLASHLOG DETAIL` command to examine the Exadata Smart Flash Log area on this cell.

```
CellCLI> list flashlog detail
         name:                   qr01cel01_FLASHLOG
         cellDisk:
FD_08_qr01cel01,FD_10_qr01cel01,FD_04_qr01cel01,FD_13_qr01cel01,FD_09_qr01cel0
1,FD_11_qr01cel01,FD_00_qr01cel01,FD_05_qr01cel01,FD_14_qr01cel01,FD_02_qr01ce
l01,FD_15_qr01cel01,FD_03_qr01cel01,FD_06_qr01cel01,FD_01_qr01cel01,FD_12_qr01
cel01,FD_07_qr01cel01
         creationTime:           2013-02-28T16:21:30-05:00
         degradedCelldisks:
         effectiveSize:          512M
         efficiency:             100.0
         id:                     1417f53b-49c3-4419-919c-933b812f0159
         size:                   512M
         status:                 normal

CellCLI>
```

6. Use the `LIST FLASHCACHECONTENT DETAIL` command to show information about the data inside Exadata Smart Flash Cache. You can see that each entry contains a series of attributes relating to a database object in the cache. For each object, you can see how much data is being cached along with the number of cache hits and misses. This information can help you to assess cache efficiency for specific database objects.

```
CellCLI> list flashcachecontent detail
...
         cachedKeepSize:         0
         cachedSize:             8192
         dbID:                   2080757153
         dbUniqueName:           DBM
         hitCount:               0
         missCount:              0
         objectNumber:           290
         tableSpaceNumber:       0

         cachedKeepSize:         0
         cachedSize:             73728
         dbID:                   2080757153
         dbUniqueName:           DBM
         hitCount:               0
         missCount:              0
         objectNumber:           457
         tableSpaceNumber:       0

         cachedKeepSize:         0
         cachedSize:             24576
         dbID:                   2080757153
         dbUniqueName:           DBM
         hitCount:               0
         missCount:              0
         objectNumber:           458
         tableSpaceNumber:       0

         cachedKeepSize:         0
         cachedSize:             8192
         dbID:                   2080757153
         dbUniqueName:           DBM
         hitCount:               0
         missCount:              0
         objectNumber:           461
         tableSpaceNumber:       0
```

```
         cachedKeepSize:          0
         cachedSize:              65536
         dbID:                    2080757153
         dbUniqueName:            DBM
         hitCount:                101
         missCount:               1
         objectNumber:            4294967294
         tableSpaceNumber:        0

CellCLI>
```

7. Exit your CellCLI session.

Practices for Lesson 4: Key Capabilities of Exadata Database Machine

Chapter 4

Practices for Lesson 4

Practices Overview

In these practices, you are introduced to four major capabilities of Exadata, namely:

- Smart Scan
- Exadata Hybrid Columnar Compression
- Exadata Smart Flash Cache
- Storage Index

Practice 4-1: Smart Scan

Overview

In this practice, you are introduced to the Smart Scan capability of Exadata. You will execute a query with and without Smart Scan enabled and you will examine statistics to measure the effect of Smart Scan.

Tasks

1. Establish a terminal connection to `qr01db01` as the `oracle` user.
2. Connect to your database with SQL*Plus. Log in as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Execute the following query (or execute the SQL script `/home/oracle/labs/lab04-01-03.sql`) and verify that the statistics are at or near zero values:

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                       .015625
physical write total bytes                                            0
cell physical IO interconnect bytes                             .015625
cell physical IO bytes saved during optimized file creation           0
cell physical IO bytes saved during optimized RMAN file restore       0
cell physical IO bytes eligible for predicate offload                 0
cell physical IO bytes saved by storage index                         0
cell physical IO bytes sent directly to DB node to balance CPU        0
cell physical IO interconnect bytes returned by smart scan            0
cell IO uncompressed bytes                                            0

10 rows selected.

SQL>
```

4. Execute the following query (or execute the SQL script `/home/oracle/labs/lab04-01-04.sql`). Note the optimizer hint that disables Smart Scan for the query.

```
SQL> select /*+ OPT_PARAM('cell_offload_processing' 'false') */
  2  count(*) from sales
  3  where time_id between '01-JAN-2003' and '31-DEC-2003'
  4  and amount_sold = 1;

  COUNT(*)
----------
     10088

SQL>
```

5. Repeat the statistics query from step 3 (or execute the SQL script `/home/oracle/labs/lab04-01-03.sql`). Note that all of the data processed by the query in step 4 (`physical read total bytes`) is returned to the database server over the storage network (`cell physical IO interconnect bytes`).

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                     559.054688
physical write total bytes                                             0
cell physical IO interconnect bytes                           559.054688
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload                  0
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan             0
cell IO uncompressed bytes                                             0

10 rows selected.

SQL>
```

6. Reconnect to your database in order to reset the session level statistics.

```
SQL> connect sales/sales
Connected.
SQL>
```

7. Repeat the statistics query from step 3 (or execute the SQL script `/home/oracle/labs/lab04-01-03.sql`) and verify that the statistics are again at or near zero values:

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                              0
physical write total bytes                                             0
cell physical IO interconnect bytes                                    0
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload                  0
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan             0
cell IO uncompressed bytes                                             0

10 rows selected.

SQL>
```

8. Execute the following query (or execute the SQL script `/home/oracle/labs/lab04-01-08.sql`). This is the same query as in step 4; however, this time there is no optimizer hint to disable Smart Scan.

```
SQL> select count(*) from sales
  2  where time_id between '01-JAN-2003' and '31-DEC-2003'
  3  and amount_sold = 1;

  COUNT(*)
----------
     10088

SQL>
```

9. Repeat the statistics query from step 3 (or execute the SQL script `/home/oracle/labs/lab04-01-03.sql`). Note that the query still performs approximately 559 MB of I/O (`physical read total bytes`). However, this time only about 228 KB is actually returned to the database server (`cell physical IO interconnect bytes`). This is Smart Scan in action.

   Also note that in this case, Smart Scan is acting on all of the I/O associated with the query. This is the case because `cell physical IO bytes eligible for predicate offload` equals `physical read total bytes`, and `cell physical IO interconnect bytes returned by smart scan` equals `cell physical IO interconnect bytes`.

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                     559.039063
physical write total bytes                                             0
cell physical IO interconnect bytes                           .222244263
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload         559.039063
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan    .222244263
cell IO uncompressed bytes                                    559.039063

10 rows selected.

SQL>
```

10. Exit your SQL*Plus session.

Practice 4-2: Exadata Hybrid Columnar Compression

Overview

In this practice, you are introduced to Exadata Hybrid Columnar Compression. You will create compressed copies of an existing database table and examine the level of compression you achieve.

Tasks

1. Establish a terminal connection to `qr01db01` as the `oracle` user.
2. Connect to your database with SQL*Plus. Log in as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Determine the size of the uncompressed `MYCUSTOMERS` table (use the SQL script `/home/oracle/labs/lab04-02-03.sql` if you prefer).

```
SQL> col segment_name format a30
SQL> select segment_name, sum(bytes)/1024/1024 MB
  2  from user_segments
  3  where segment_name like 'MYCUST%'
  4  group by segment_name;

SEGMENT_NAME                           MB
------------------------------ ----------
MYCUSTOMERS                           208

SQL>
```

4. Verify that the `CUSTOMERS` table is uncompressed (use the SQL script `/home/oracle/labs/lab04-02-04.sql` if you prefer).

```
SQL> select table_name, compression, compress_for
  2  from user_tables
  3  where table_name like 'MYCUST%';

TABLE_NAME                     COMPRESS COMPRESS_FOR
------------------------------ -------- ------------
MYCUSTOMERS                    DISABLED

SQL>
```

5. Exadata Hybrid Columnar Compression achieves its highest levels of compression with data that is direct-path inserted. Execute the following `ALTER SESSION` commands to ensure the use of direct-path inserts later in the practice.

```
SQL> alter session force parallel query;

Session altered.

SQL> alter session force parallel ddl;

Session altered.

SQL> alter session force parallel dml;

Session altered.

SQL>
```

6. Create a compressed copy of the `MYCUSTOMERS` table by using the `QUERY HIGH` warehouse compression mode.

```
SQL> create table mycust_query compress for query high
  2  parallel 4 nologging as select * from mycustomers;

Table created.

SQL>
```

7. Create a compressed copy of the `MYCUSTOMERS` table using the `ARCHIVE HIGH` archive compression mode. Note that it may take approximately one minute for the table to be created.

```
SQL> create table mycust_archive compress for archive high
  2  parallel 4 nologging as select * from mycustomers;

Table created.

SQL>
```

8. Verify the compression mode settings for the tables you just created (use the SQL script `/home/oracle/labs/lab04-02-04.sql` again if you prefer).

```
SQL> select table_name, compression, compress_for
  2  from user_tables
  3  where table_name like 'MYCUST%';

TABLE_NAME                     COMPRESS COMPRESS_FOR
------------------------------ -------- ------------
MYCUST_QUERY                   ENABLED  QUERY HIGH
MYCUST_ARCHIVE                 ENABLED  ARCHIVE HIGH
MYCUSTOMERS                    DISABLED

SQL>
```

9. Compare the size of the original uncompressed table with the two compressed copies you created (use the SQL script `/home/oracle/labs/lab04-02-03.sql` if you prefer). Calculate the compression ratios achieved using the formula:

   `Compression Ratio = Uncompressed Size / Compressed Size`

```
SQL> select segment_name,sum(bytes)/1024/1024 MB
  2  from user_segments
  3  where segment_name like 'MYCUST%'
  4  group by segment_name;

SEGMENT_NAME                        MB
------------------------------ ----------
MYCUSTOMERS                         208
MYCUST_ARCHIVE                       18
MYCUST_QUERY                         31

SQL>
```

10. Drop the compressed tables that you created in this practice.

```
SQL> drop table mycust_query;

Table dropped.

SQL> drop table mycust_archive;

Table dropped.

SQL>
```

11. Exit your SQL*Plus session.

Practice 4-3: Exadata Smart Flash Cache

Overview

In this practice, you will examine the use of Exadata Smart Flash Cache. You will execute a series of record lookups and use database statistics to verify the use of Exadata Smart Flash Cache. You will also compare the execution statistics with and without the use of Exadata Smart Flash Cache.

Tasks

1. Establish a terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.
2. Execute the following two commands to drop and then re-create Exadata Smart Flash Cache on all of your Exadata cells. You must perform this action so that Exadata Smart Flash Cache is empty at the beginning of this practice; thus ensuring consistent results later in the practice. To do this, you will use the distributed command line utility (`dcli`) that is provided with Exadata. Using `dcli` you can execute cell-level administrative commands simultaneously on multiple Exadata cells. A more detailed discussion of `dcli` features and options is provided later in the course. Be careful not to add any extra spaces in the server list following the `dcli -c` command-line option.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> drop flashcache
qr01cel01: Flash cache qr01cel01_FLASHCACHE successfully dropped
qr01cel02: Flash cache qr01cel02_FLASHCACHE successfully dropped
qr01cel03: Flash cache qr01cel03_FLASHCACHE successfully dropped
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> create flashcache all
qr01cel01: Flash cache qr01cel01_FLASHCACHE successfully created
qr01cel02: Flash cache qr01cel02_FLASHCACHE successfully created
qr01cel03: Flash cache qr01cel03_FLASHCACHE successfully created
[celladmin@qr01cel01 ~]$
```

3. Establish a separate terminal connection to `qr01db01` as the `oracle` user.
4. Connect to your database with SQL*Plus. Log in as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

5. Execute the following query (or execute the SQL script `/home/oracle/labs/lab04-03-05.sql`) and verify that the statistics are at or near zero values:

```
SQL> select a.name, b.value from v$sysstat a, v$mystat b
  2  where a.statistic# = b.statistic# and
  3  (a.name like '%flash cache read hits'
  4  or a.name like 'cell phy%'
  5  or a.name like 'physical read tot%'
  6  or a.name like 'physical read req%');

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
physical read total IO requests                                        0
physical read total multi block requests                               0
physical read requests optimized                                       0
physical read total bytes optimized                                    0
physical read total bytes                                              0
cell physical IO interconnect bytes                                    0
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload                  0
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan             0
cell flash cache read hits                                             0

13 rows selected.

SQL>
```

6. Flush the buffer cache to ensure that the queries in step 8 must retrieve the required data from the Exadata cells.

```
SQL> alter system flush buffer_cache;

System altered.

SQL>
```

7. Configure the session to display server output.

```
SQL> set serveroutput on
SQL>
```

8. The following PL/SQL block performs 500 record lookups spread across a reasonably large table. The workload is representative of the scattered record access normally associated with an OLTP application. Execute the PL/SQL block against your database (or execute the SQL script `/home/oracle/labs/lab04-03-08.sql`). Note that the workload may take a few minutes to complete.

```
SQL> declare
  2    a number;
  3    s number := 0;
  4  begin
  5    for n in 1 .. 500 loop
  6      select cust_credit_limit into a from customers
  7        where cust_id=n*2000;
  8      s := s+a;
  9    end loop;
 10    dbms_output.put_line('Transaction total = '||s);
 11  end;
 12  /
Transaction total = 3761500

PL/SQL procedure successfully completed.

SQL>
```

9. Repeat the statistics query from step 5 (or execute the SQL script `/home/oracle/labs/lab04-03-05.sql`). Note the high number of IO requests (`physical read total IO requests`) relative to the low number of optimized requests (`physical read requests optimized` and `cell flash cache read hits`). This indicates that the queries were mostly satisfied by using physical disk reads and is indicative of a recently emptied cache.

```
SQL> select a.name, b.value from v$sysstat a, v$mystat b
  2  where a.statistic# = b.statistic# and
  3  (a.name like '%flash cache read hits'
  4  or a.name like 'cell phy%'
  5  or a.name like 'physical read tot%'
  6  or a.name like 'physical read req%');

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
physical read total IO requests                                          959
physical read total multi block requests                                  46
physical read requests optimized                                         140
physical read total bytes optimized                                  1146880
physical read total bytes                                           17833984
cell physical IO interconnect bytes                                 17833984
cell physical IO bytes saved during optimized file creation                0
cell physical IO bytes saved during optimized RMAN file restore            0
cell physical IO bytes eligible for predicate offload                      0
cell physical IO bytes saved by storage index                              0
cell physical IO bytes sent directly to DB node to balance CPU             0
cell physical IO interconnect bytes returned by smart scan                 0
cell flash cache read hits                                               140

13 rows selected.

SQL>
```

10. Reconnect to your database in order to reset the session level statistics.

```
SQL> connect sales/sales
Connected.
SQL>
```

11. Repeat the statistics query from step 5 (or execute the SQL script `/home/oracle/labs/lab04-03-05.sql`) and verify that the statistics are again at or near zero values:

```
SQL> select a.name, b.value from v$sysstat a, v$mystat b
  2  where a.statistic# = b.statistic# and
  3  (a.name like '%flash cache read hits'
  4  or a.name like 'cell phy%'
  5  or a.name like 'physical read tot%'
  6  or a.name like 'physical read req%');

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
physical read total IO requests                                        1
physical read total multi block requests                               0
physical read requests optimized                                       0
physical read total bytes optimized                                    0
physical read total bytes                                           8192
cell physical IO interconnect bytes                                 8192
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload                  0
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan             0
cell flash cache read hits                                             0

13 rows selected.

SQL>
```

12. Again, flush the buffer cache to ensure that the queries in the next step must retrieve the required data from the Exadata cells. Also, configure your new session to display server output.

```
SQL> alter system flush buffer_cache;

System altered.

SQL> set serveroutput on
SQL>
```

13. Re-execute the PL/SQL block introduced in step 8 (or execute the SQL script `/home/oracle/labs/lab04-03-08.sql`). Confirm that the result is the same.

```
SQL> declare
  2    a number;
  3    s number := 0;
  4  begin
  5    for n in 1 .. 500 loop
  6      select cust_credit_limit into a from customers
  7        where cust_id=n*2000;
  8      s := s+a;
  9    end loop;
 10    dbms_output.put_line('Transaction total = '||s);
 11  end;
 12  /
Transaction total = 3761500

PL/SQL procedure successfully completed.

SQL>
```

14. Repeat the statistics query from step 5 (or execute the SQL script `/home/oracle/labs/lab04-03-05.sql`). Compare the values for `cell flash cache read hits` and `physical read total IO requests`. They should be much closer together, indicating that most of the I/Os were satisfied by Exadata Smart Flash Cache.

```
SQL> select a.name, b.value from v$sysstat a, v$mystat b
  2  where a.statistic# = b.statistic# and
  3  (a.name like '%flash cache read hits'
  4  or a.name like 'cell phy%'
  5  or a.name like 'physical read tot%'
  6  or a.name like 'physical read req%');

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
physical read total IO requests                                         1013
physical read total multi block requests                                   0
physical read requests optimized                                         905
physical read total bytes optimized                                  7413760
physical read total bytes                                            8298496
cell physical IO interconnect bytes                                  8298496
cell physical IO bytes saved during optimized file creation                0
cell physical IO bytes saved during optimized RMAN file restore            0
cell physical IO bytes eligible for predicate offload                      0
cell physical IO bytes saved by storage index                              0
cell physical IO bytes sent directly to DB node to balance CPU             0
cell physical IO interconnect bytes returned by smart scan                 0
cell flash cache read hits                                               905

13 rows selected.

SQL>
```

In an earlier practice, you saw how to obtain general information about Exadata Smart Flash Cache on an Exadata cell using the `LIST FLASHCACHECONTENT` CellCLI command. Over the remainder of this practice, you will learn how to isolate specific information in Exadata Smart Flash Cache.

15. Use the following query (or execute the SQL script `/home/oracle/labs/lab04-03-15.sql`) to determine the number of optimized physical reads (reads optimized by Exadata Smart Flash Cache or Exadata storage index) for the `SALES.CUSTOMERS` table. Note the tablespace number (`TS#`) and object number (`DATAOBJ#`) associated with the table.

```
SQL> select owner, object_name, tablespace_name, ts#, dataobj#,
  2  statistic_name, value
  3  from v$segment_statistics
  4  where owner='SALES' and object_name='CUSTOMERS'
  5  and statistic_name='optimized physical reads';

OWNER                          OBJECT_NAME
------------------------------ ------------------------------
TABLESPACE_NAME                       TS#   DATAOBJ#
------------------------------ ---------- ----------
STATISTIC_NAME                                                    VALUE
---------------------------------------------------------------- ----------
SALES                          CUSTOMERS
SALES                                   7      77111
optimized physical reads                                          473


SQL>
```

16. Back in the terminal session connected to `qr01cel01`, launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

17. Use the tablespace number (TS#) and object number (DATAOBJ#) you gathered in step 15 to query the Exadata Smart Flash Cache. The output relates specifically to the SALES.CUSTOMERS table.

```
CellCLI> list flashcachecontent where objectnumber=77111 -
> and tablespacenumber=7 and dbuniquename=DBM detail
         cachedKeepSize:         0
         cachedSize:             10362880
         dbID:                   2080757153
         dbUniqueName:           DBM
         hitCount:               150
         missCount:              146
         objectNumber:           77111
         tableSpaceNumber:       7

CellCLI>
```

Note that in step 15 the value for optimized physical reads is 473 while the hitCount observed in this step is 150. Why is this so? In the remaining time allocated for this practice, query the Exadata Smart Flash Cache hitCount values for the other cells (qr01cel02 and qr01cel03) and compare the hitCount total across all the cells with the optimized physical reads value observed in step 15. Explain your observations.

18. Exit your terminal sessions.

Practice 4-4: Storage Index

Overview

In this practice, you are introduced to the storage index capability of Exadata. You will execute a query multiple times and examine statistics to measure the effect of storage index.

Tasks

1. Establish a terminal connection to `qr01db01` as the `oracle` user.
2. Connect to your database with SQL*Plus. Log in as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Execute the following query (or execute the SQL script `/home/oracle/labs/lab04-04-03.sql`) and verify that the statistics are at or near zero values:

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                              0
physical write total bytes                                             0
cell physical IO interconnect bytes                                    0
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload                  0
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan             0
cell IO uncompressed bytes                                             0

10 rows selected.

SQL>
```

4. Execute the following query (or execute the SQL script `/home/oracle/labs/lab04-04-04.sql`).

```
SQL> select cust_gender,count(*) from mycustomers
  2  where cust_income_level = 'C: 50,000 - 69,999'
  3  group by cust_gender;

C   COUNT(*)
- ----------
M       47924
F       27300

SQL>
```

5. Repeat the statistics query from step 3 (or execute the SQL script `/home/oracle/labs/lab04-04-03.sql`).

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                     204.484375
physical write total bytes                                             0
cell physical IO interconnect bytes                           1.07327271
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload            204.4375
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan    1.02639771
cell IO uncompressed bytes                                       204.4375

10 rows selected.

SQL>
```

The statistics show that the query in step 4 was conducted using Smart Scan. Note, however, that `cell physical IO bytes saved by storage index` is zero. This is because storage indexes are memory structures which do not persist when the Exadata cells are restarted. They are dynamically built when tables are referenced for the first time after the cells restart. Now that the `mycustomers` table has been scanned as a result of the query in step 4, all subsequent queries on the `mycustomers` table can benefit from whatever storage indexes the Exadata cells automatically create.

6. Reconnect to your database in order to reset the session-level statistics.

```
SQL> connect sales/sales
Connected.
SQL>
```

7. Repeat the statistics query from step 3 (or execute the SQL script `/home/oracle/labs/lab04-04-03.sql`) and verify that the statistics are again at or near zero values:

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                              0
physical write total bytes                                             0
cell physical IO interconnect bytes                                    0
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload                  0
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan             0
cell IO uncompressed bytes                                             0

10 rows selected.

SQL>
```

8. Re-execute the query from step 4 (or execute the SQL script `/home/oracle/labs/lab04-04-04.sql`).

```
SQL> select cust_gender,count(*) from mycustomers
  2  where cust_income_level = 'C: 50,000 - 69,999'
  3  group by cust_gender;

C   COUNT(*)
- ----------
M      47924
F      27300

SQL>
```

9. Repeat the statistics query from step 3 (or execute the SQL script `/home/oracle/labs/lab04-04-03.sql`).

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                        204.4375
physical write total bytes                                              0
cell physical IO interconnect bytes                           1.00943756
cell physical IO bytes saved during optimized file creation             0
cell physical IO bytes saved during optimized RMAN file restore         0
cell physical IO bytes eligible for predicate offload            204.4375
cell physical IO bytes saved by storage index                 111.140625
cell physical IO bytes sent directly to DB node to balance CPU          0
cell physical IO interconnect bytes returned by smart scan    1.00943756
cell IO uncompressed bytes                                       93.296875

10 rows selected.

SQL>
```

This time you will see that the query in step 8 benefits from the storage index. Instead of conducting more than 204 MB of I/O inside the cells, storage indexes were used to bypass more than 111 MB of I/O. In other words, approximately 93 MB of I/O was conducted instead of 204 MB. Queries that benefit from storage indexes can execute more quickly using fewer resources which allows other workloads to benefit from the unused I/O resources.

10. Exit your SQL*Plus session.

Practices for Lesson 5: Exadata Database Machine Initial Configuration

Chapter 5

Practices for Lesson 5

Practices Overview

In this practice, you will be introduced to the Oracle Exadata Deployment Assistant

Practice 5-1: Using the Oracle Exadata Deployment Assistant

Overview

In this practice, you will be introduced to the Oracle Exadata Deployment Assistant. You will use the assistant to generate a set of configuration files for an example Database Machine implementation scenario.

Tasks

1. Establish a terminal session connected to `qr01db01` using the `oracle` OS user. Ensure that you specify the `-X` option for `ssh`.

```
$ ssh -X oracle@qr01db01
oracle@qr01db01 password: <oracle>
[oracle@qr01db01 ~]$
```

2. The Oracle Exadata Configuration Assistant is bundled in the patch containing the OneCommand configuration utility for Database Machine. In your laboratory environment, the Oracle Exadata Configuration Assistant is located under `/home/oracle/labs/Exaconf` on `qr01db01`. Change directory to the directory containing the Oracle Exadata Configuration Assistant.

```
[oracle@qr01db01 ~]$ cd labs/Exaconf
[oracle@qr01db01 Exaconf]$
```

3. Start the Oracle Exadata Configuration Assistant.

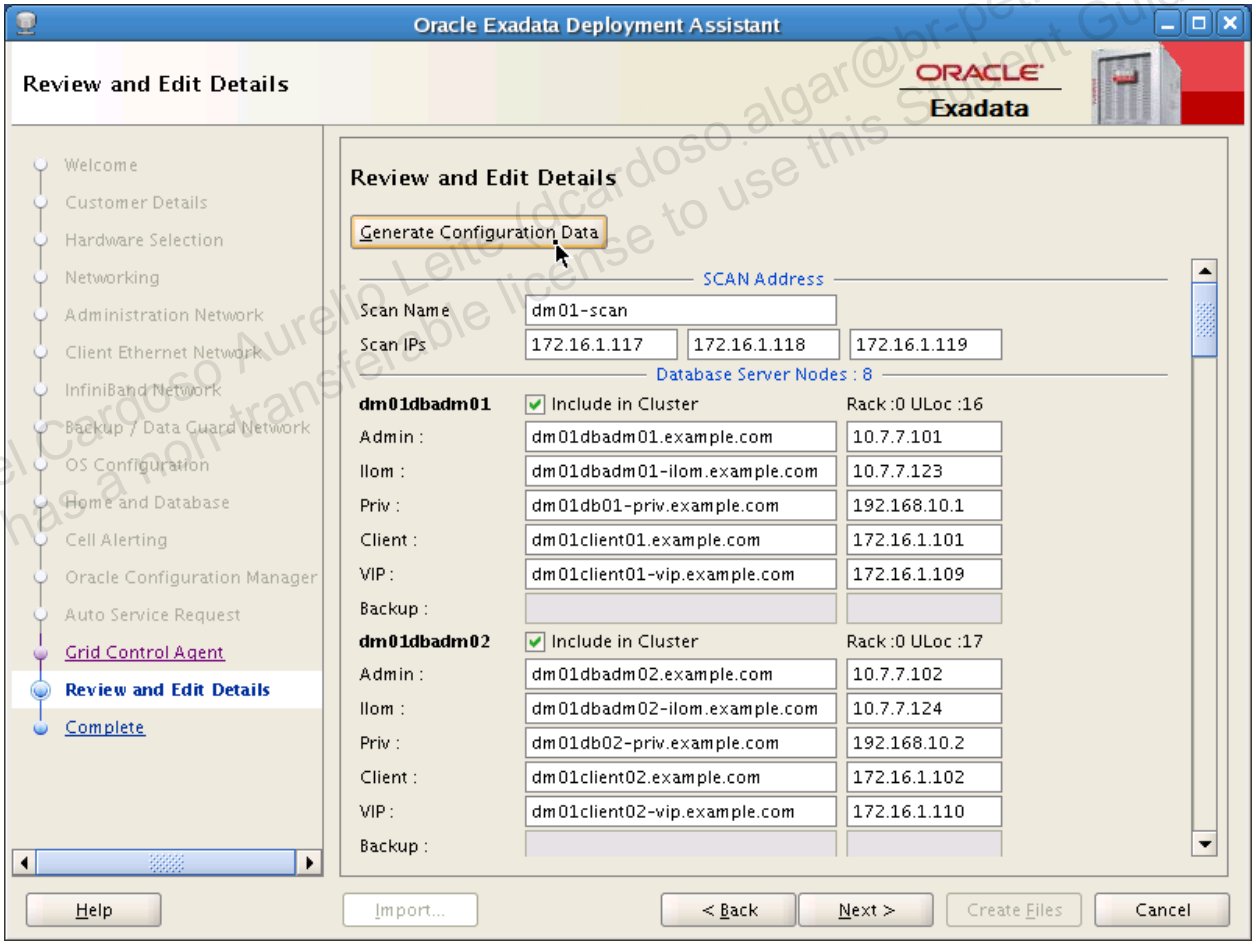
```
[oracle@qr01db01 Exaconf]$ ./exaconf.sh
```

4. By using the information in the following table, populate the Oracle Exadata Configuration Assistant pages. Leave the default values for fields that are not specified in the following table. Examine the options and additional information presented on each page. Proceed until you reach the Review and Edit Details page.

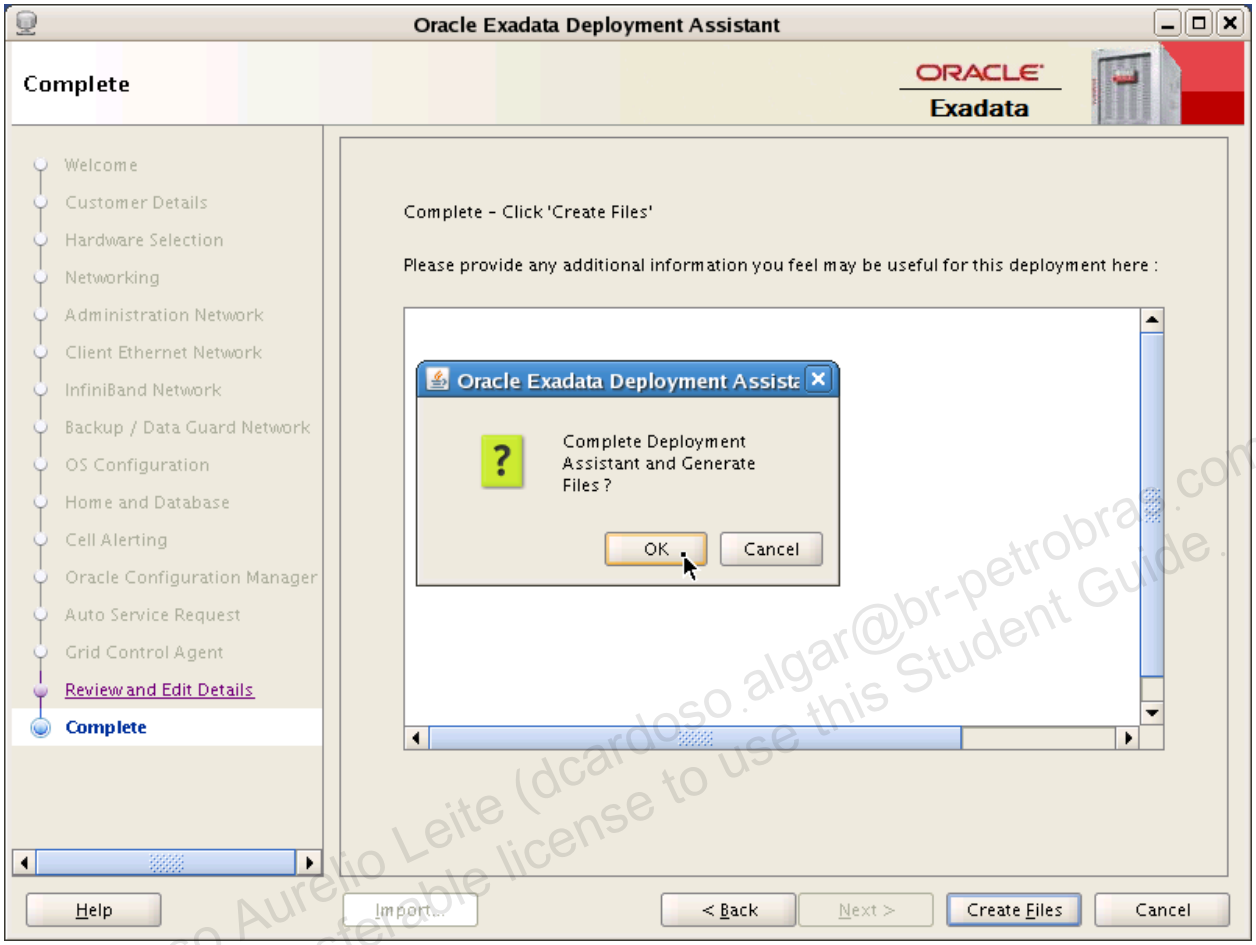
Step	Window/Page Description	Choices or Values
a.	Customer Details	Customer Name: `Example Industries` Application: `Example Full Rack`
b.	Hardware Selection	This is your deployment: `X3-2 Full Rack HP`
c.	Networking	Examine the IP address requirements summary.
d.	Administration Network	Starting IP Address for Pool: `10.7.7.101` Gateway: `10.7.7.1`
e.	Client Ethernet Network	Starting IP Address for Pool: `172.16.1.101` Gateway: `172.16.1.1`
f.	InfiniBand Network	Leave the default values.
g.	Backup / Data Guard Ethernet Network	Leave unconfigured.

Step	Window/Page Description	Choices or Values
h.	OS Configuration	Domain Name: `example.com` DNS Servers: `10.7.7.5` NTP Servers: `10.7.7.5` Check the option for “Separate Grid Infrastructure owner from the Database Owner”
i.	Home and Database	Leave the default values.
j.	Cell Alerting	Leave unconfigured.
k.	Oracle Configuration Manager	Leave unconfigured.
l.	Auto Service Request	Leave unconfigured.
m.	Grid Control Agent	Leave unconfigured.

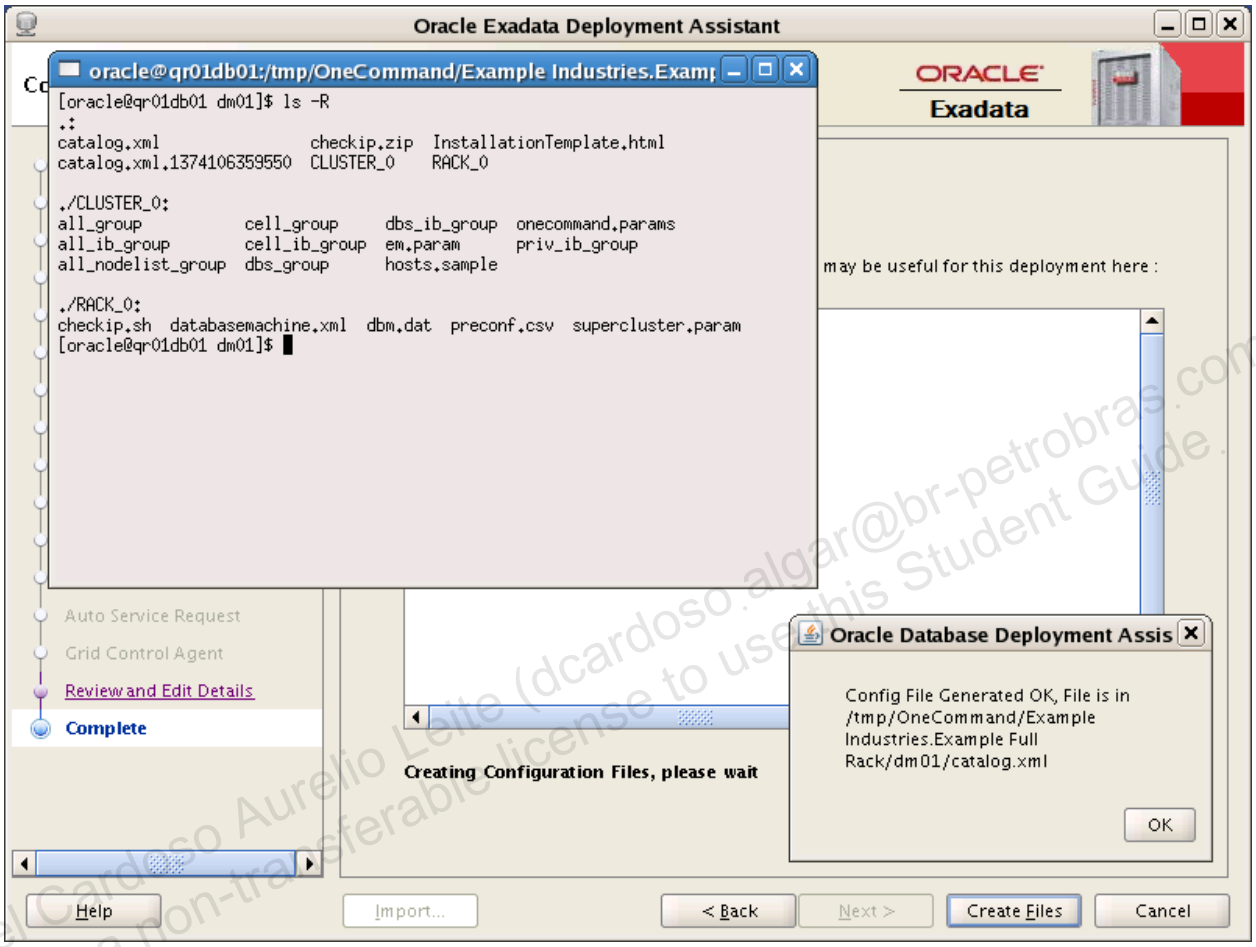
5. On the Review and Edit Details page, click Generate Configuration Data. Examine the configuration details on the page. When you have finished, click Next to proceed.



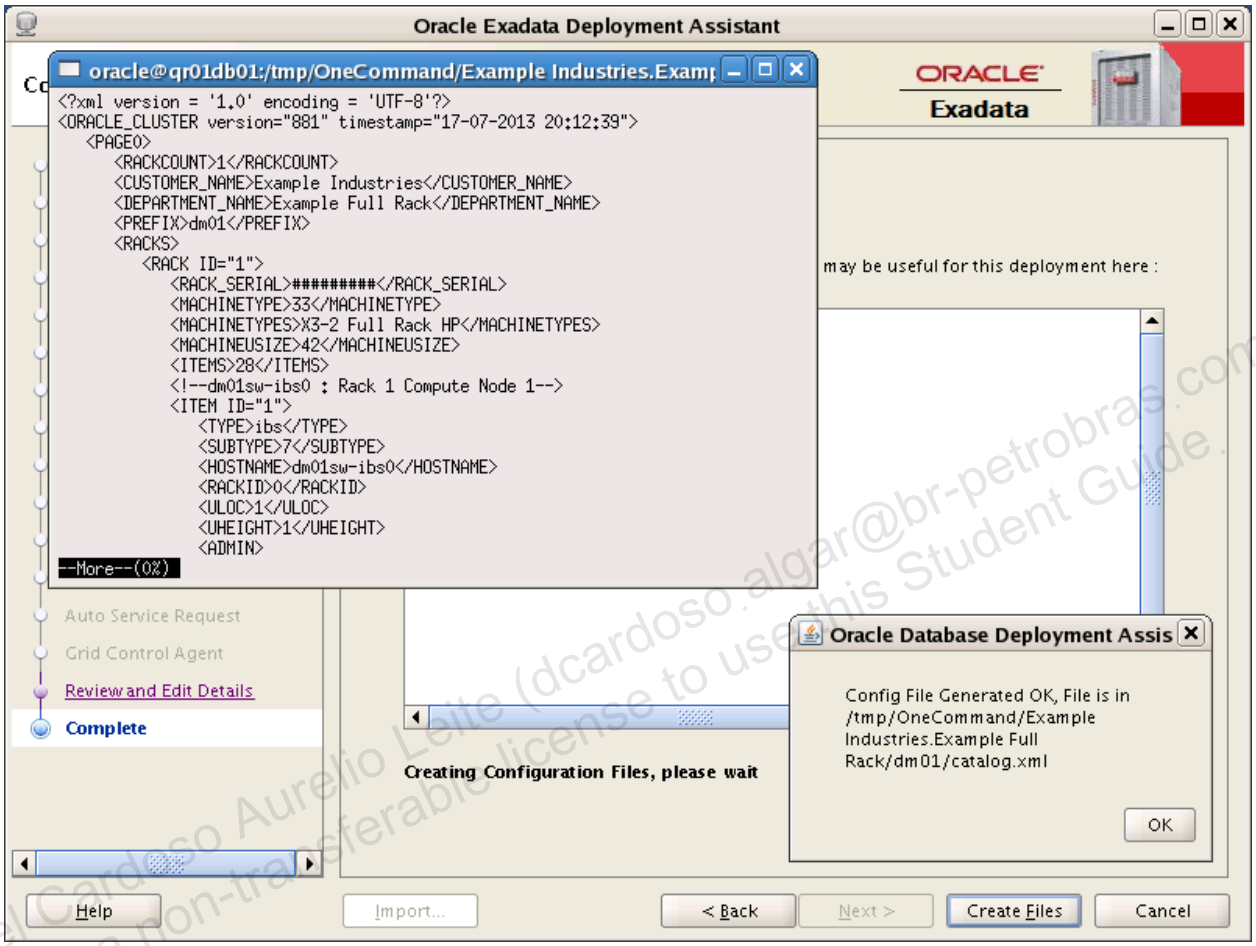
6. On the Complete page, click Create Files to create the configuration files for this deployment scenario. Click OK in the confirmation dialog.



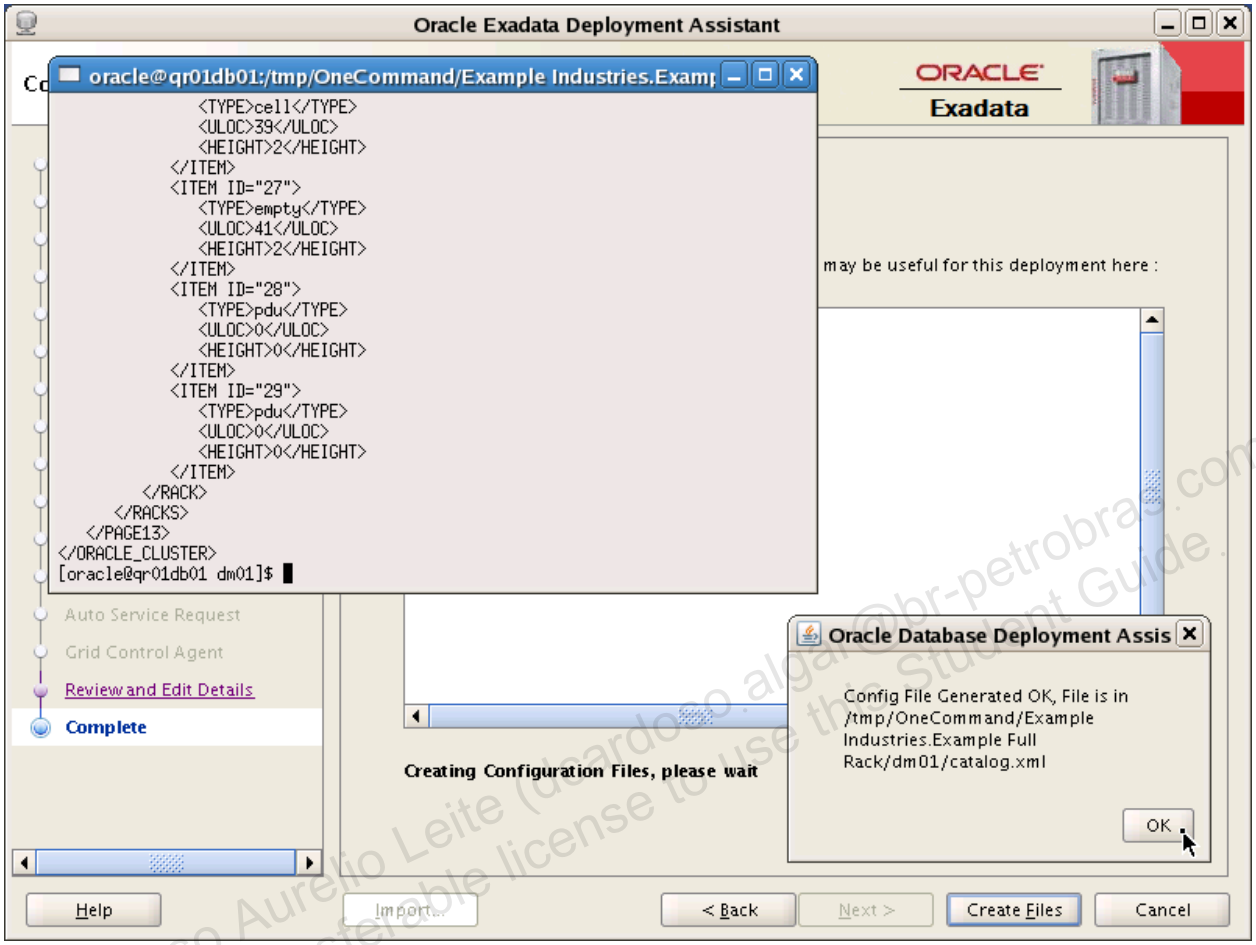
7. At this point, you should see a dialog indicating that the configuration files were successfully generated. The assistant will also open a new terminal window with the current directory set to the location of the generated configuration files. You may use this terminal window to examine the generated files. Run `ls -R` and confirm that you see a file listing similar to the screen shot below.



8. View the `catalog.xml` file (using the `more` command or `vi` if you prefer). This file, also known as the Database Machine schematic file, is one of the main configuration files that drives the Database Machine configuration process. Examine the file and confirm that the details within it match your inputs to the deployment assistant. Take a few minutes to examine the other configuration files as well.



9. After you are satisfied, click OK in the dialog window to complete the Oracle Exadata Deployment Assistant session.

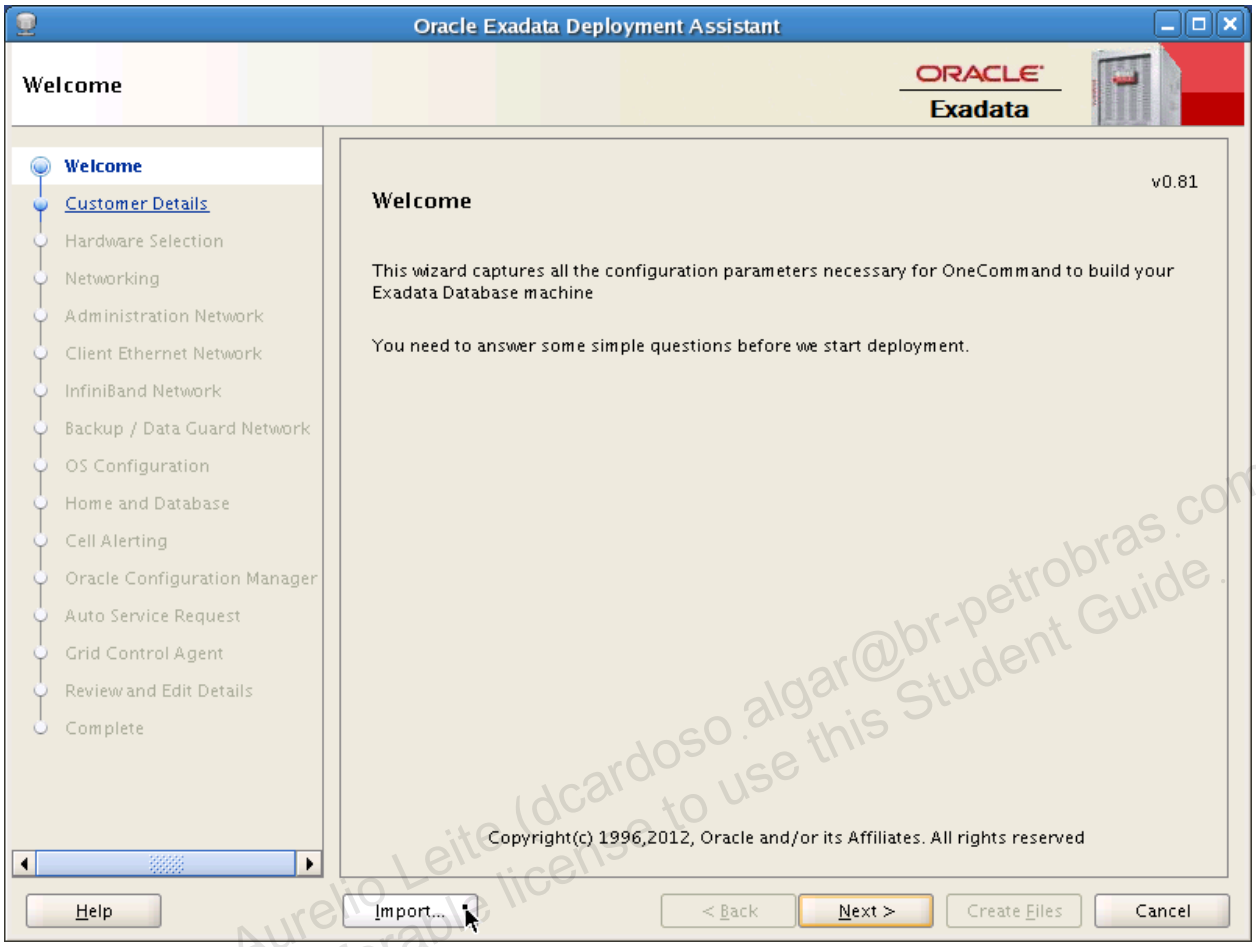


Congratulations, you have now used the Oracle Exadata Deployment Assistant to generate a set of configuration files for a database Machine deployment scenario. In the final part of this practice you will use the deployment assistant to import your configuration information and make some changes.

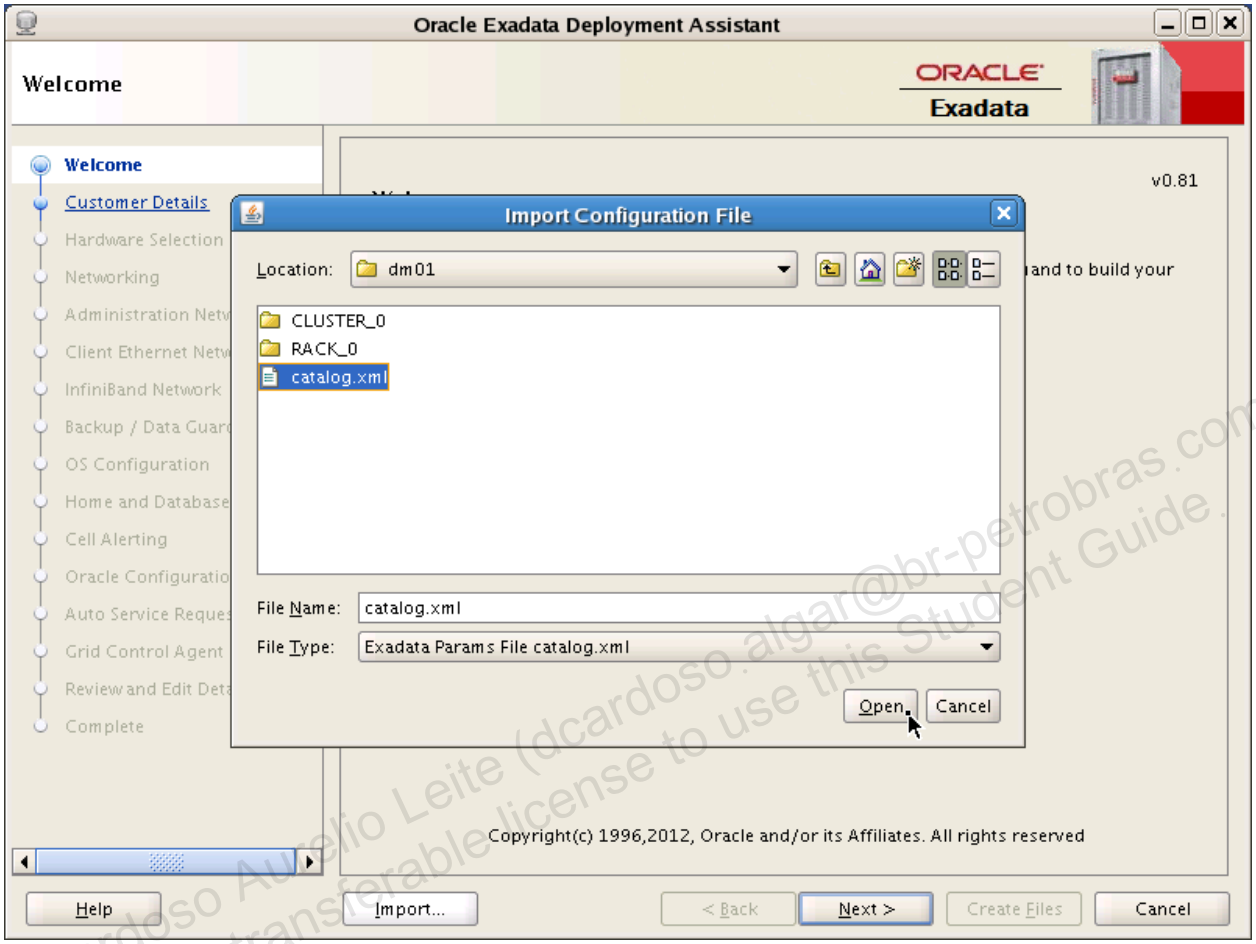
10. By using your original terminal window, start the Oracle Exadata Configuration Assistant.

```
[oracle@qr01db01 Exaconf]$ ./exaconf.sh
```

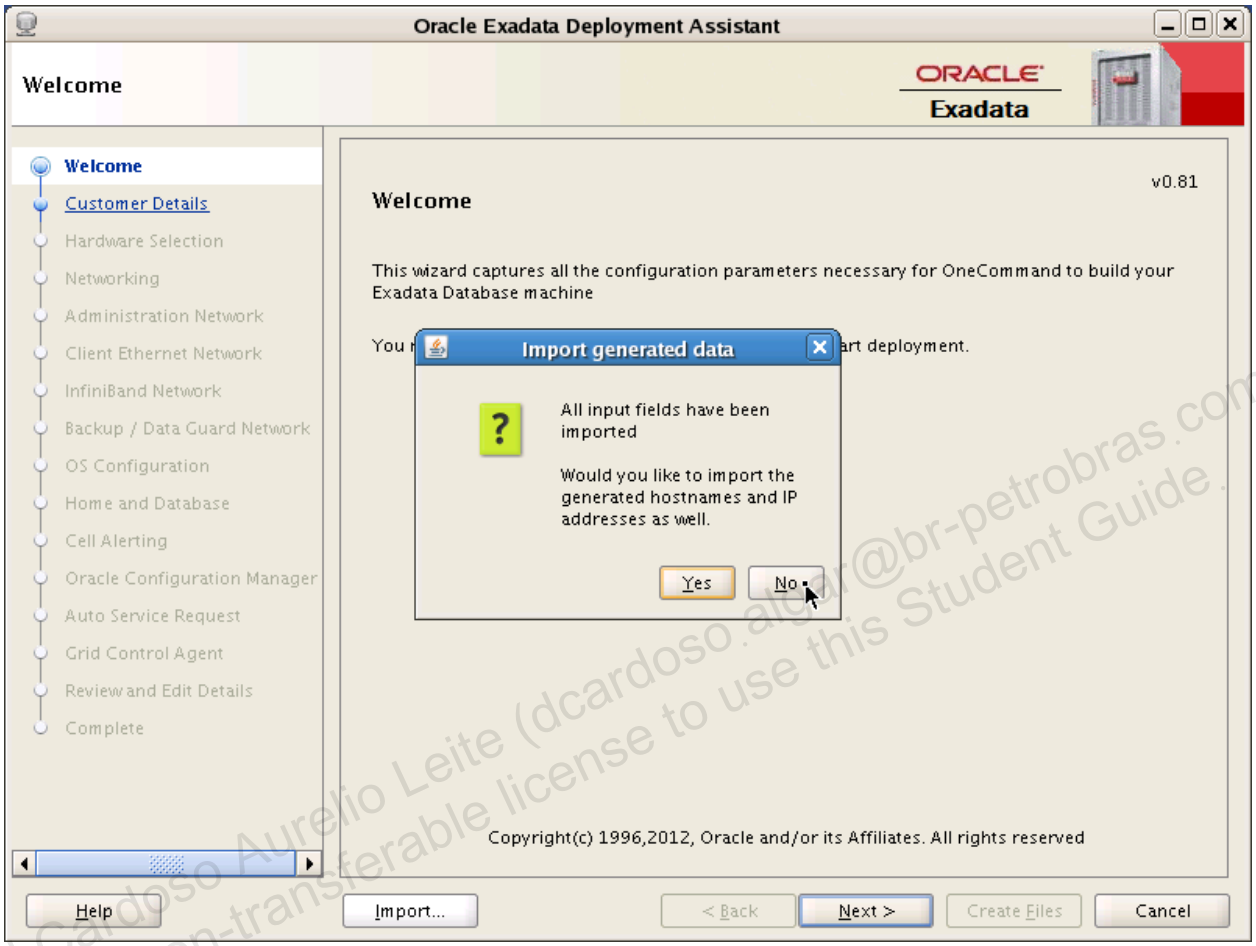
11. On the Welcome page, click Import.



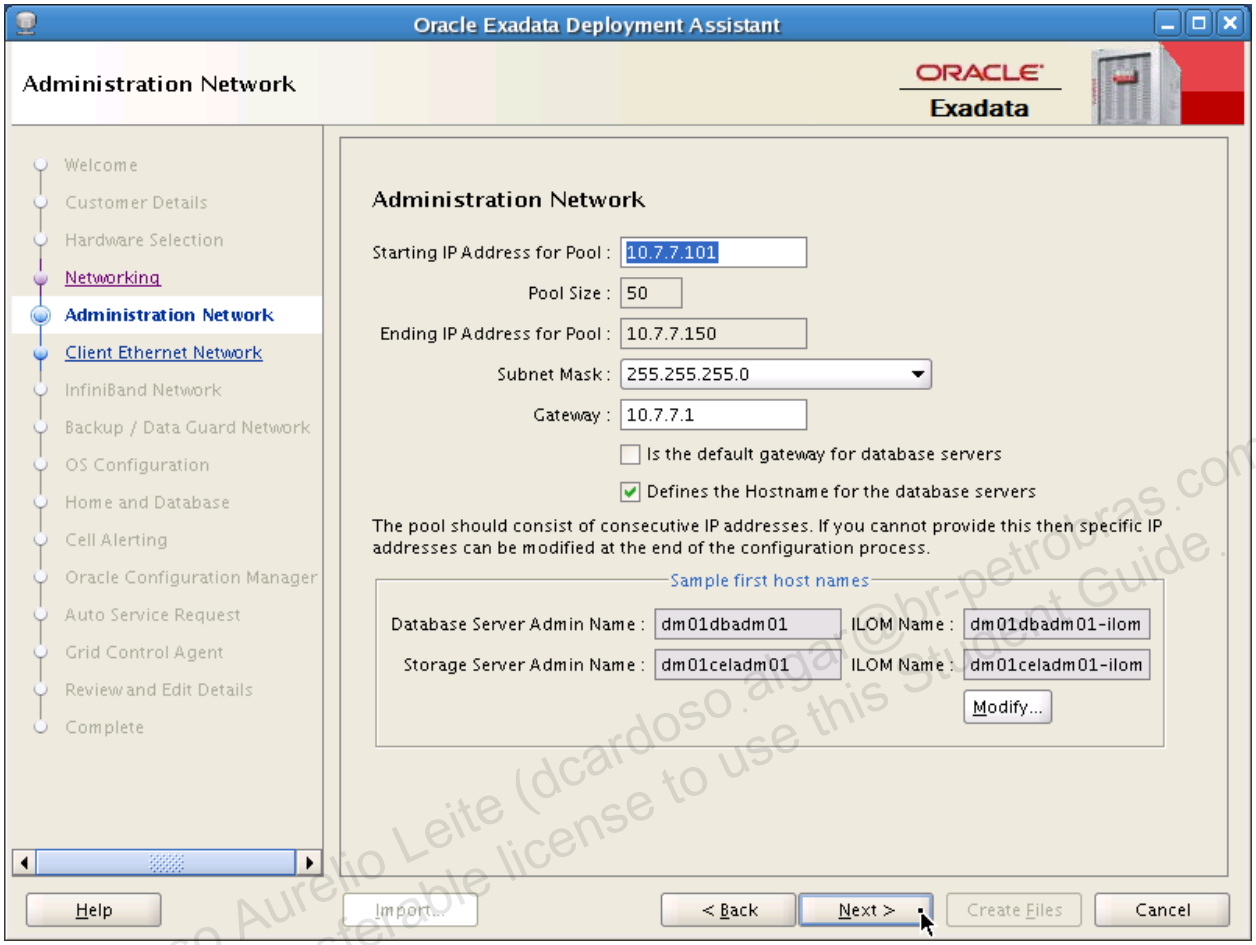
12. Use the Import Configuration File dialog to select the `catalog.xml` file under `/tmp/OneCommand/Example Industries.Example Full Rack/dm01`. After you select the file, click Open to proceed.



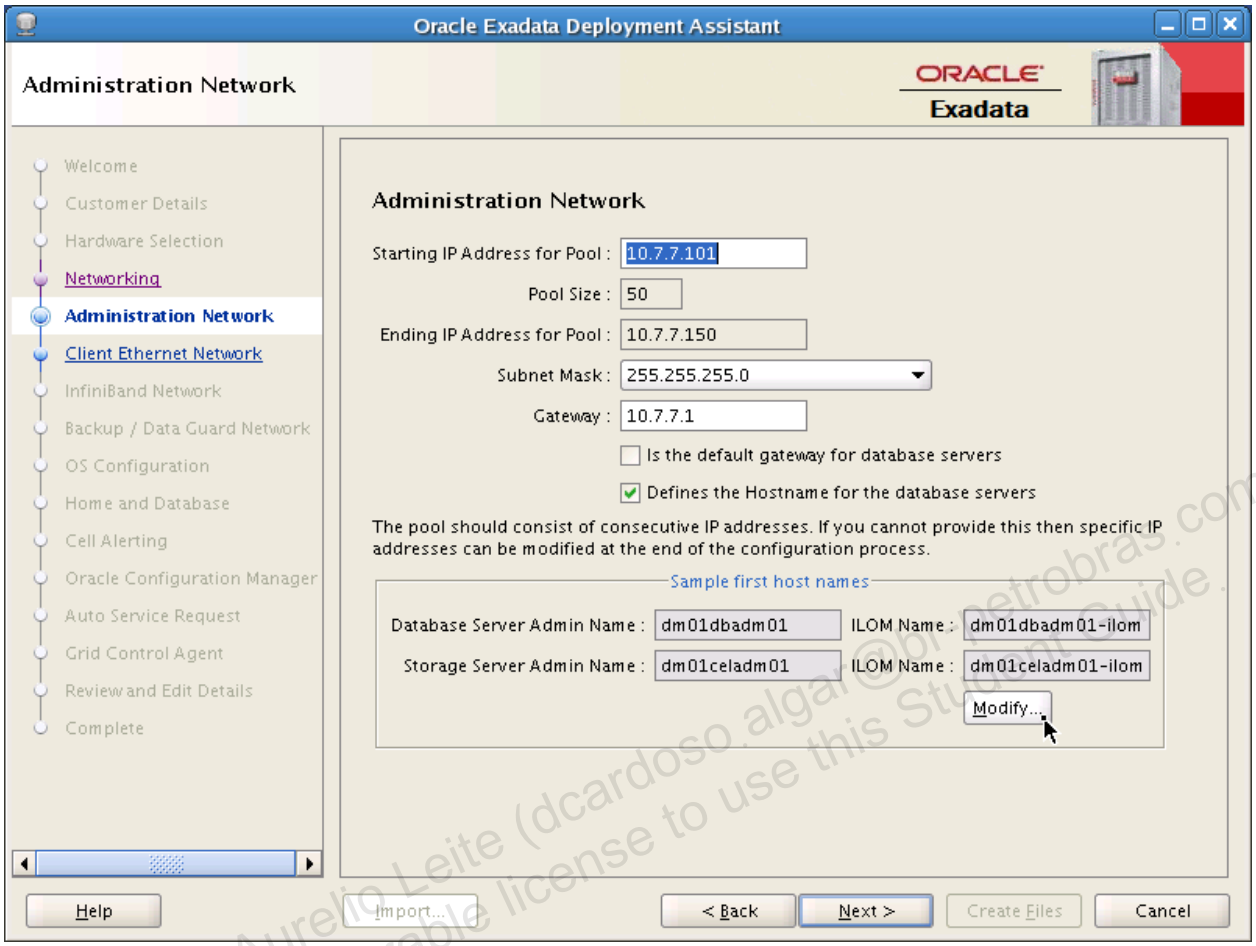
13. Click No in the Import generated data dialog box. This instructs the deployment assistant to only import the inputs gathered during the original deployment assistant session, and not all of the consequently generated settings. Finally, click OK in the dialog box indicating that the file import is complete.



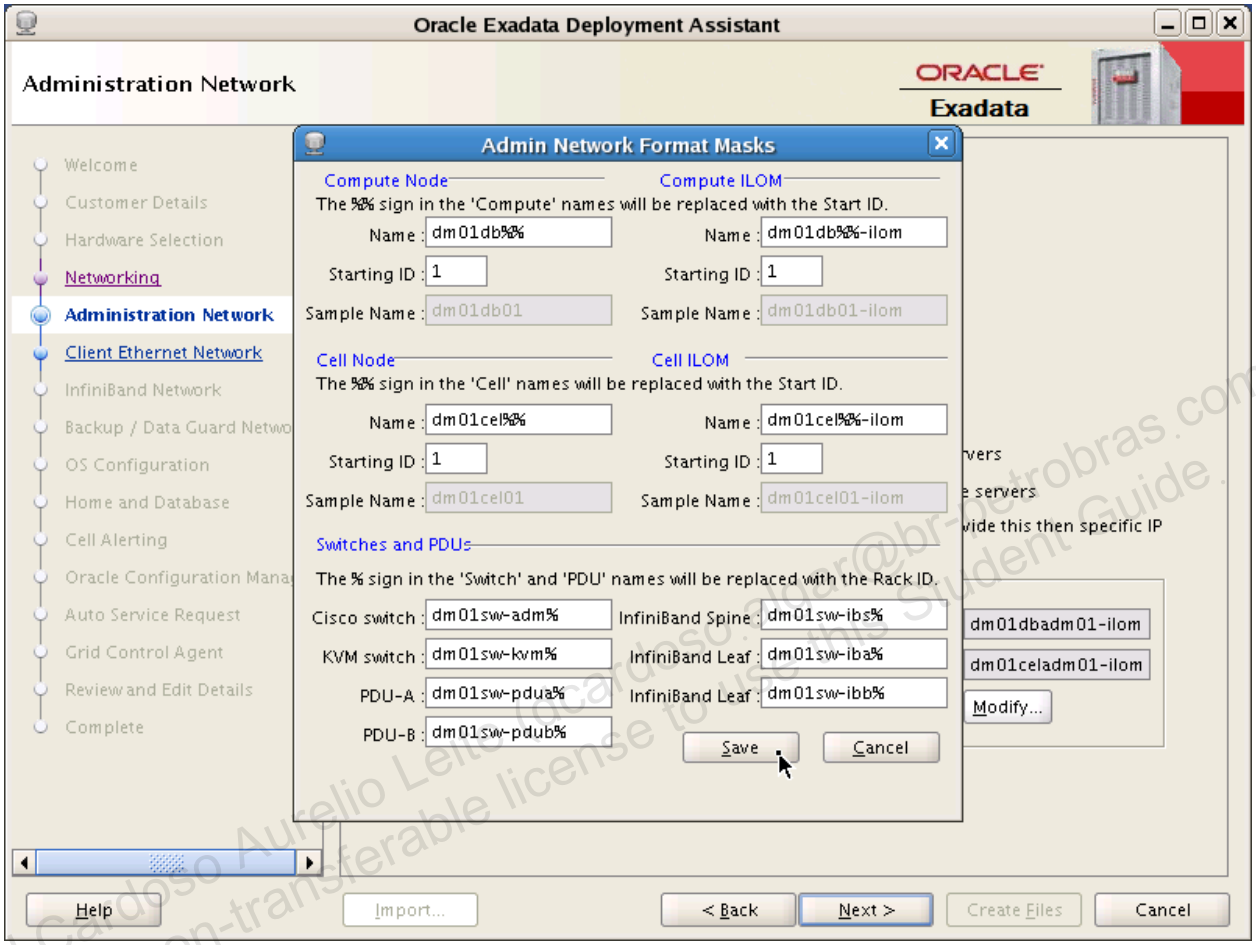
14. Navigate to the Administration Network page. Along the way, confirm that the deployment assistant pages contain details from your previously generated configuration.



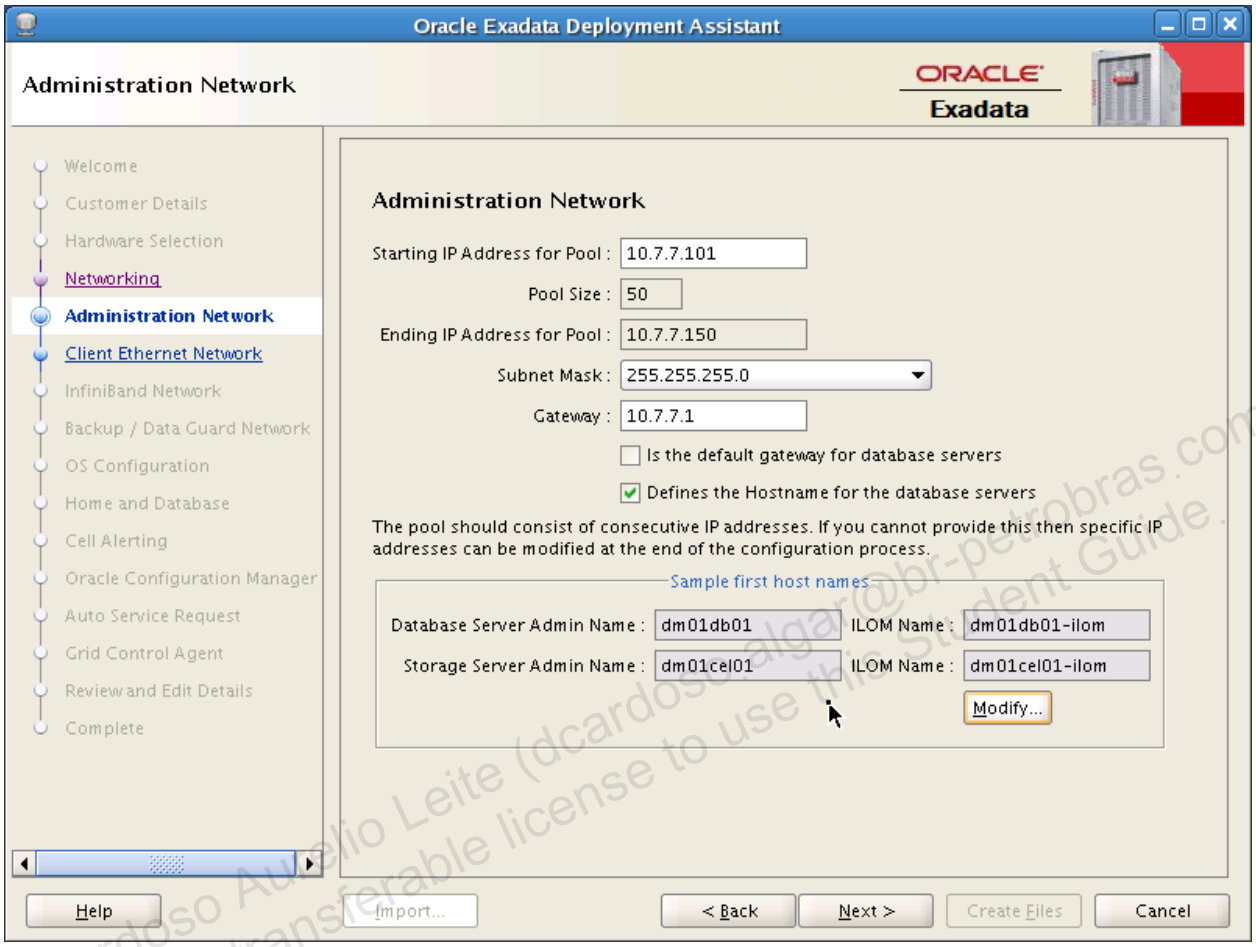
15. Imagine that you wish to implement a non-default host naming convention. Click Modify.



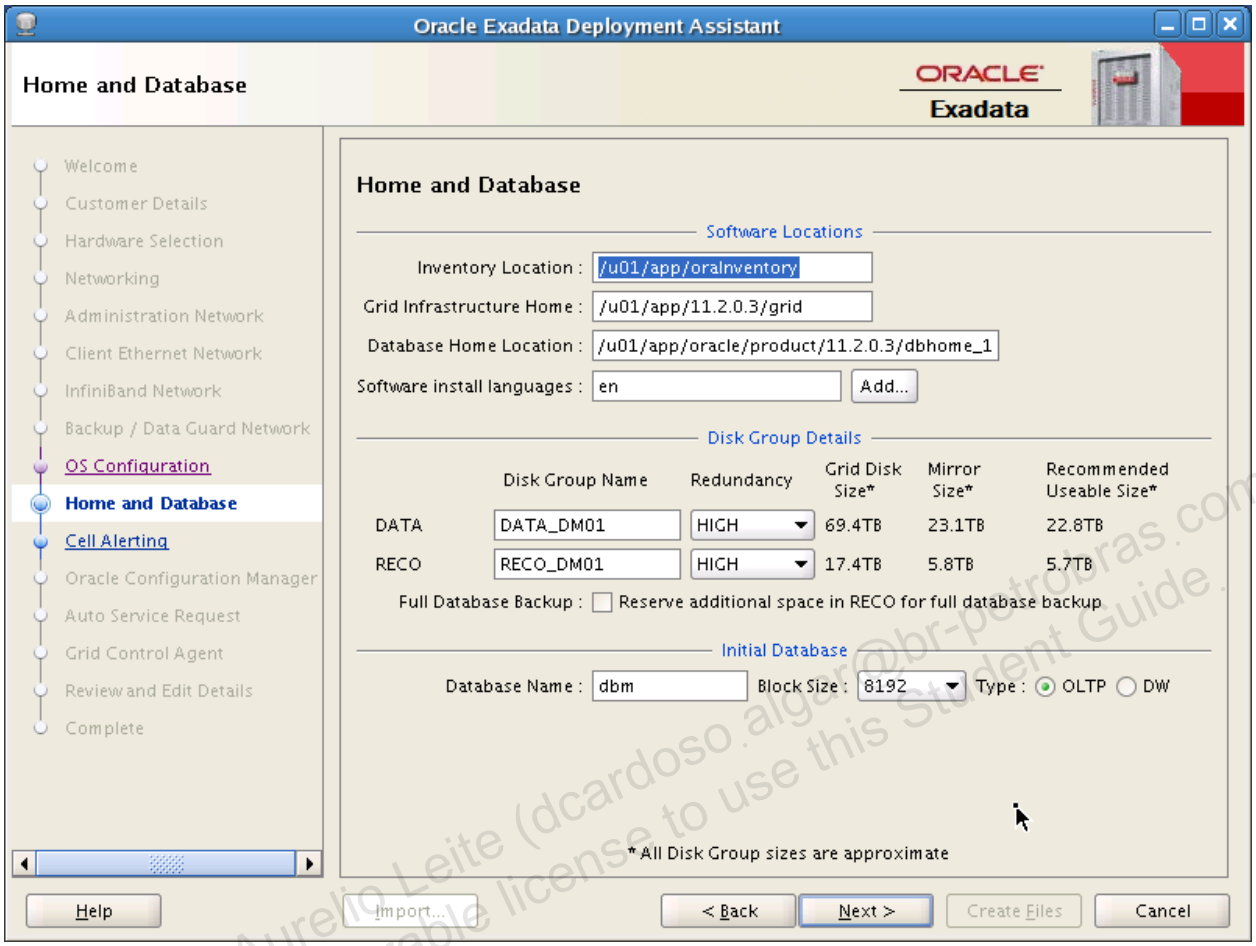
16. In the Admin Network Format Masks dialog, modify the Compute Node Name to `dm01db%%` and modify the Cell Node Name to `dm01cel%%`. Notice that as you make these changes, the associated ILOM hosts names also change. Finally, click Save to accept the modifications.



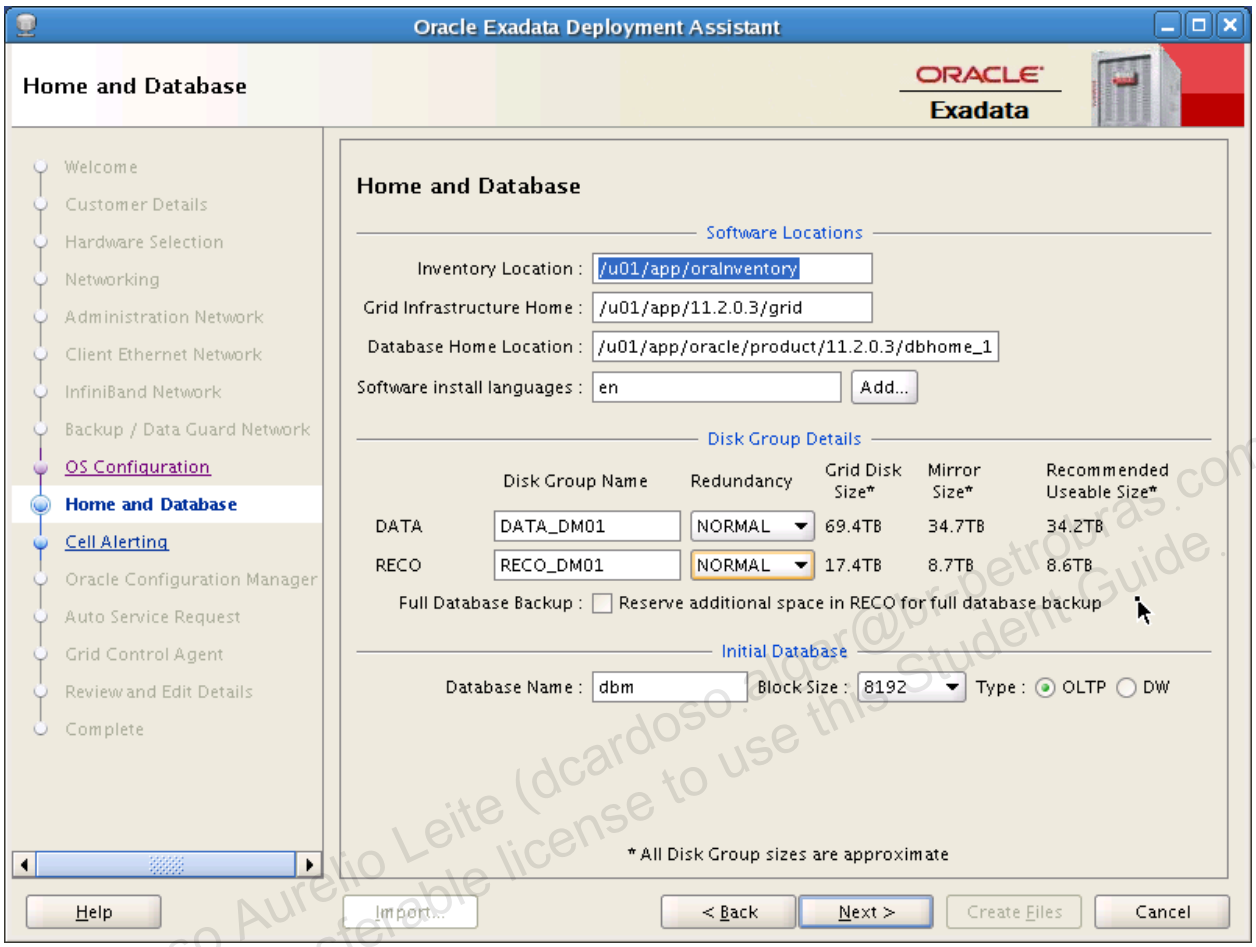
17. Notice how the change alters the sample first host names displayed in the Administration Network Page. Similar capabilities are also available on the Client Ethernet Network and InfiniBand Network pages.



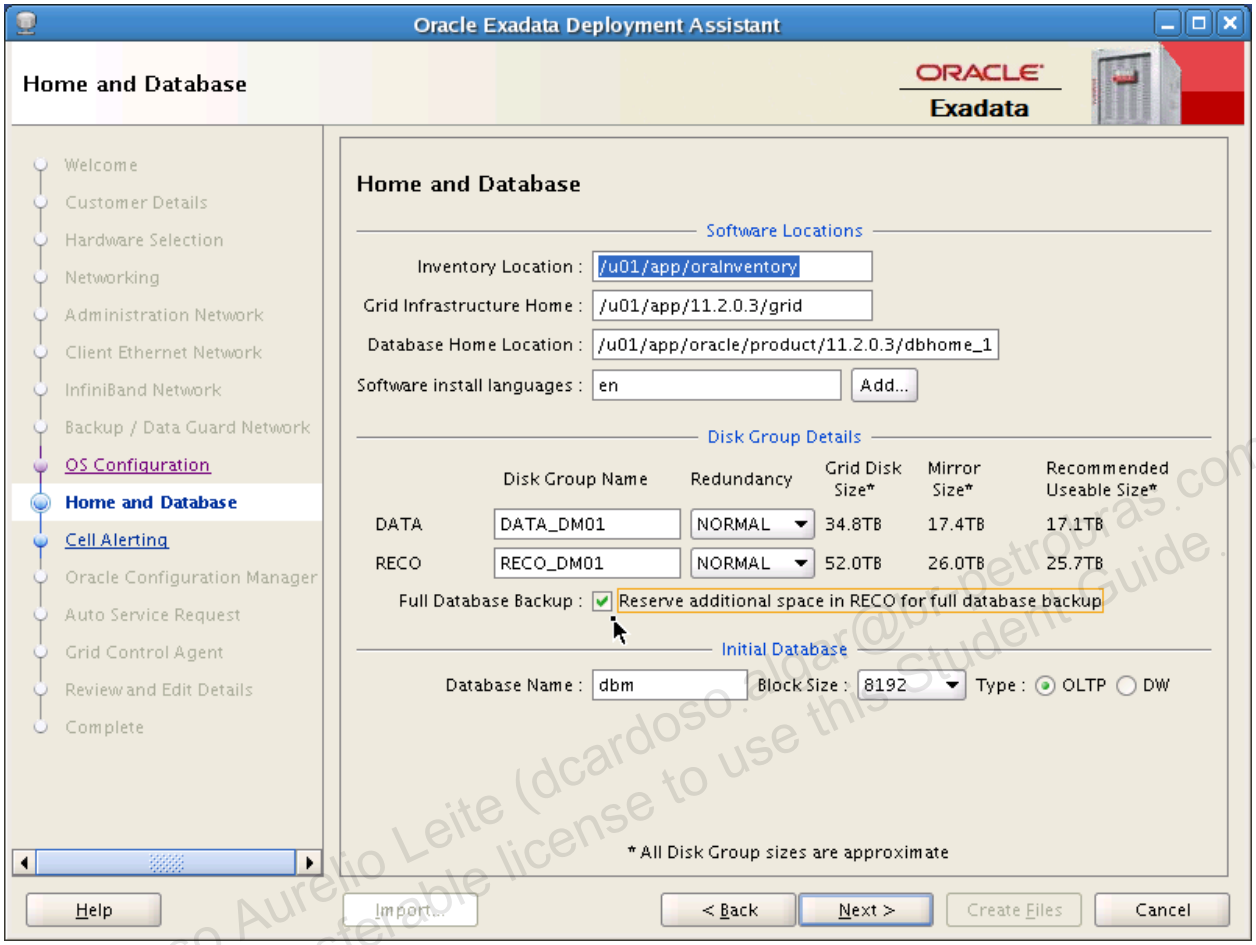
18. Navigate to the Home and Database page.



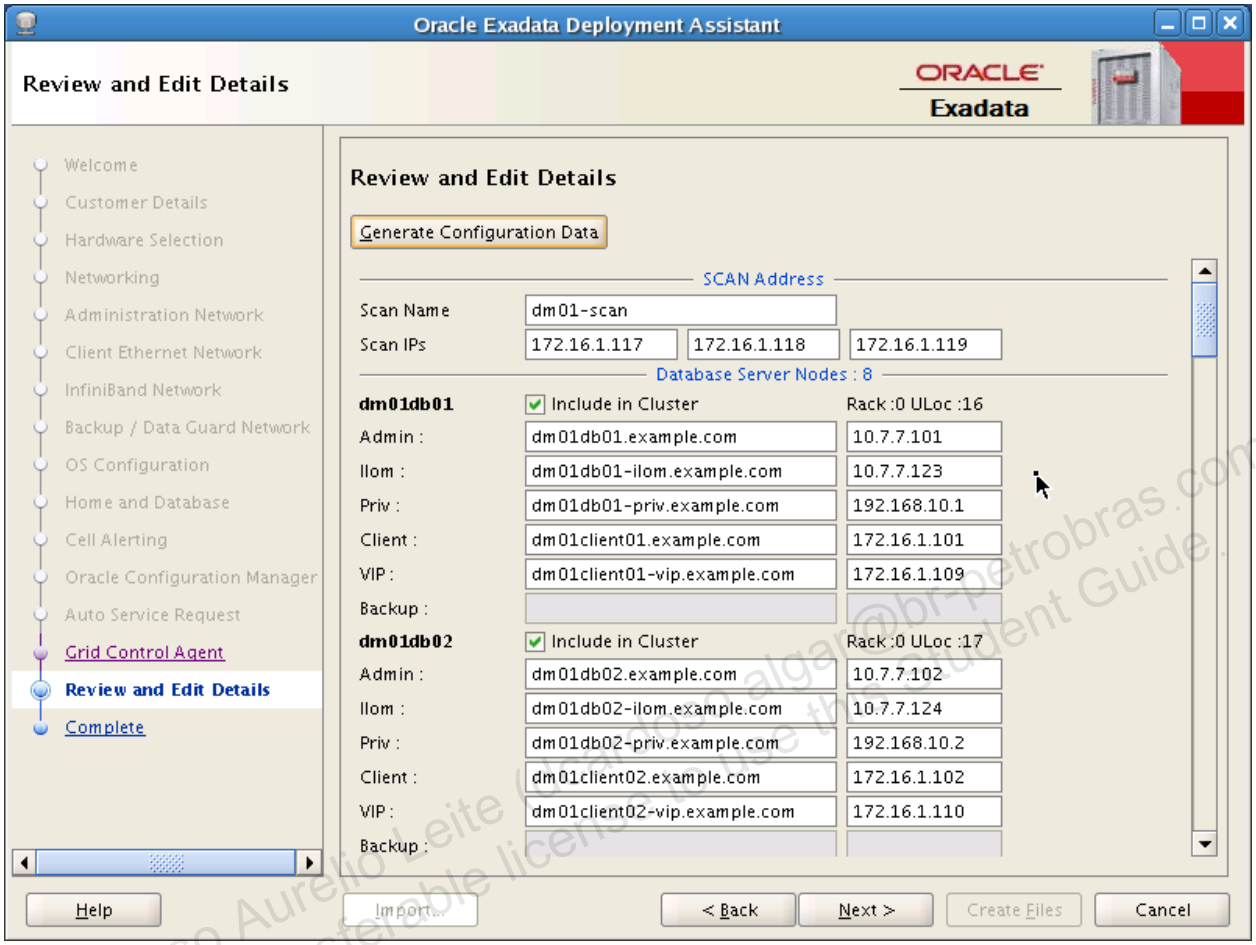
19. Modify the disk group redundancy setting to `NORMAL` for both disk groups. Notice the resulting change in disk group sizing information.



20. Check the option to reserve additional space in the RECO disk group for a full database backup. Notice the resulting change in disk group sizing information.

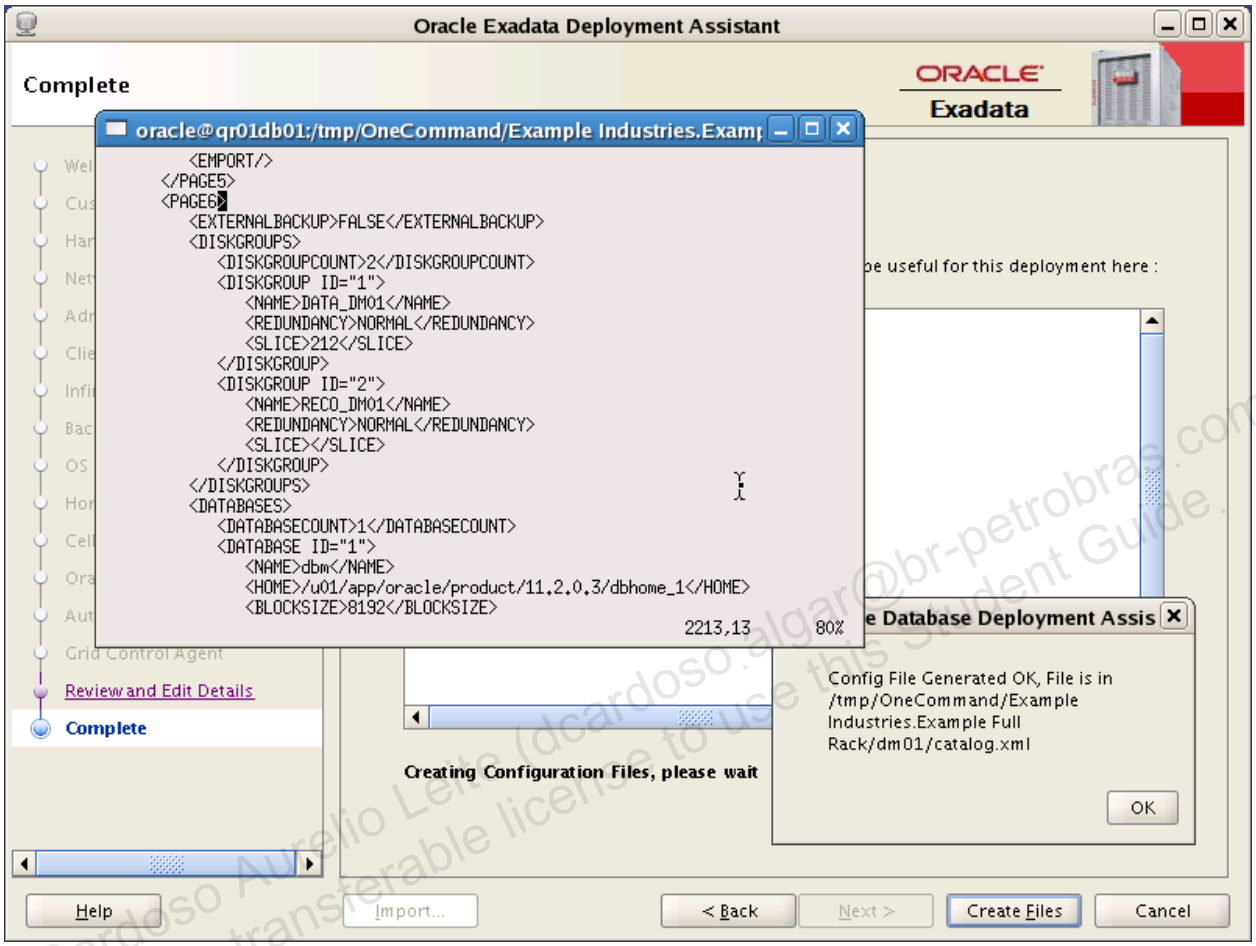


21. Navigate to the Review and Edit Details page and click Generate Configuration Data. Notice how the host names have changed to reflect the changes you made in step 16.

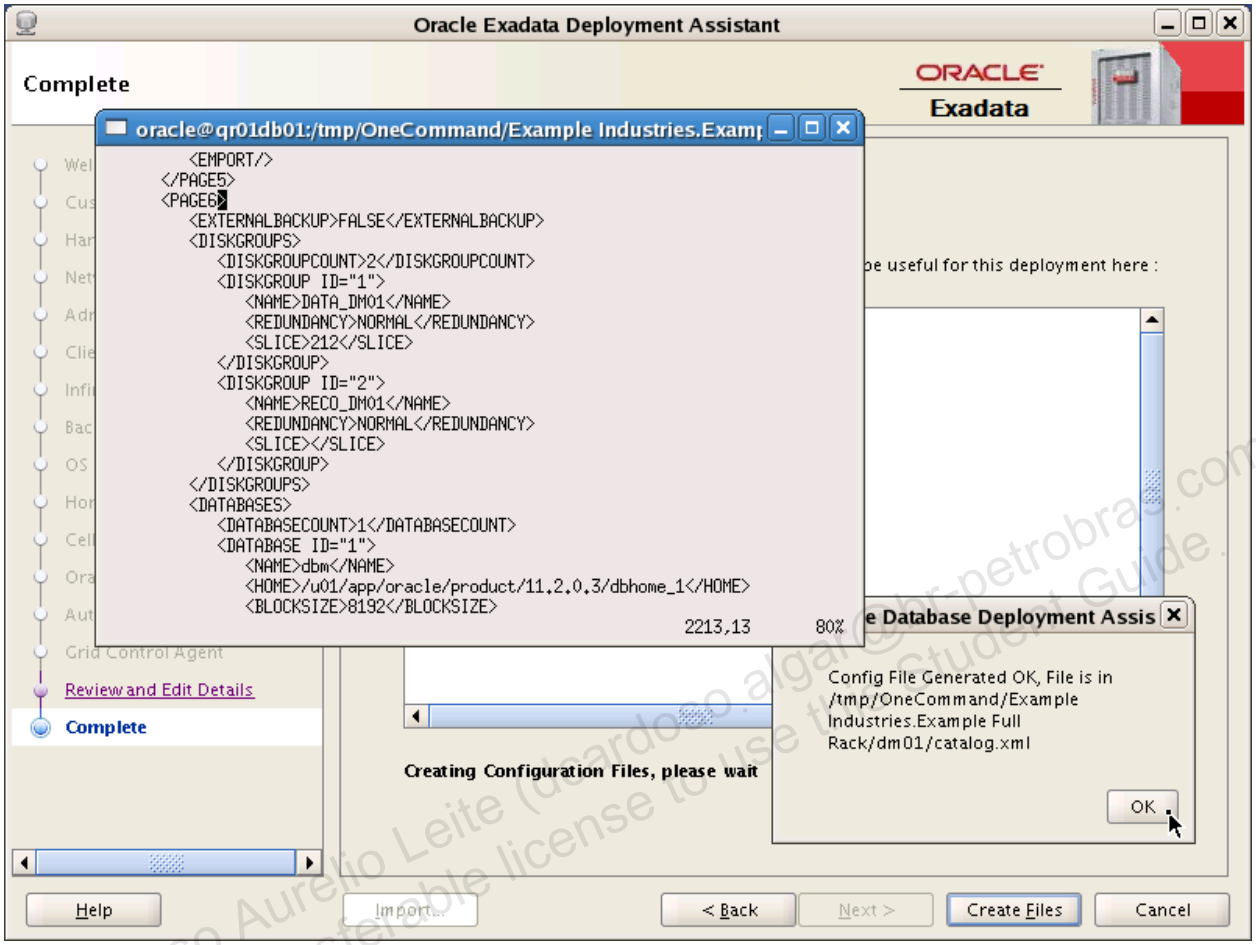


22. Generate an updated set of configuration files and examine the updated `catalog.xml` file to verify the modified disk group settings.

    Hint: See steps 6-8 earlier in this practice.



23. After you are satisfied, click OK in the dialog box to complete the Oracle Exadata Deployment Assistant session.



24. Exit your terminal sessions.

Practices for Lesson 6: Exadata Storage Server Configuration

Chapter 6

Practices for Lesson 6

Practices Overview

In these practices, you will perform a variety of Exadata configuration tasks, including cell configuration and storage reconfiguration. You will also consume Exadata storage using ASM, configure Exadata storage security, exercise the privileges associated with the different cell user accounts and use the distributed command line utility (`dcli`).

Practice 6-1: Cell Configuration

Overview

In this practice you examine, set, and validate some Exadata cell parameters.

Tasks

1. Establish a terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.
2. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

3. Execute the following CellCLI command to examine the attributes of the cell.

```
CellCLI> list cell detail
         name:                   qr01cel01
         bbuTempThreshold:       60
         bbuChargeThreshold:     800
         bmcType:                absent
         cellVersion:            OSS_11.2.3.2.1_LINUX.X64_130109
         cpuCount:               1
         diagHistoryDays:        7
         fanCount:               1/1
         fanStatus:              normal
         flashCacheMode:         WriteThrough
         id:                     8ab50138-a667-4793-a976-c540dc1930c5
         interconnectCount:      3
         interconnect1:          eth1
         iormBoost:              0.0
         ipaddress1:             192.168.1.103/24
         kernelVersion:          2.6.32-400.11.1.el5uek
         makeModel:              Fake hardware
         metricHistoryDays:      7
         offloadEfficiency:      596.0
         powerCount:             1/1
         powerStatus:            normal
         releaseVersion:         11.2.3.2.1
         releaseTrackingBug:     14522699
         status:                 online
         temperatureReading:     0.0
         temperatureStatus:      normal
         upTime:                 0 days, 1:55
         cellsrvStatus:          running
         msStatus:               running
         rsStatus:               running

CellCLI>
```

4. Configure the cell to send email alerts to a fictitious Exadata administrator.

```
CellCLI> alter cell smtpServer='my_mail.example.com',           -
>                  smtpFromAddr='john.doe@example.com',         -
>                  smtpFrom='John Doe',                         -
>                  smtpToAddr='jane.smith@example.com',         -
>                  notificationPolicy='critical,warning,clear', -
>                  notificationMethod='mail'
Cell qr01cel01 successfully altered

CellCLI>
```

5. Re-examine the cell configuration to verify the changes you made in step 4.

```
CellCLI> list cell detail
         name:                   qr01cel01
         bbuTempThreshold:       60
         bbuChargeThreshold:     800
         bmcType:                absent
         cellVersion:            OSS_11.2.3.2.1_LINUX.X64_130109
         cpuCount:               1
         diagHistoryDays:        7
         fanCount:               1/1
         fanStatus:              normal
         flashCacheMode:         WriteThrough
         id:                     8ab50138-a667-4793-a976-c540dc1930c5
         interconnectCount:      3
         interconnect1:          eth1
         iormBoost:              0.0
         ipaddress1:             192.168.1.103/24
         kernelVersion:          2.6.32-400.11.1.el5uek
         makeModel:              Fake hardware
         metricHistoryDays:      7
         notificationMethod:     mail
         notificationPolicy:     critical,warning,clear
         offloadEfficiency:      596.0
         powerCount:             1/1
         powerStatus:            normal
         releaseVersion:         11.2.3.2.1
         releaseTrackingBug:     14522699
         smtpFrom:               "John Doe"
         smtpFromAddr:           john.doe@example.com
         smtpServer:             my_mail.example.com
         smtpToAddr:             jane.smith@example.com
         status:                 online
         temperatureReading:     0.0
         temperatureStatus:      normal
         upTime:                 0 days, 1:56
         cellsrvStatus:          running
         msStatus:               running
         rsStatus:               running
CellCLI>
```

6. Execute the following CellCLI command to validate the email attributes configured for the cell.

   Note: Executing this command attempts to send a test email to each of the configured email addresses. The validation process only confirms the ability to successfully send a test email using the specified configuration. The validation process does not confirm the existence of the target email account, nor does it confirm successful receipt of the test email message. In this case an error message is observed because the target email server (`my_mail.example.com`) does not really exist.

```
CellCLI> alter cell validate mail

CELL-02578: An error was detected in the SMTP configuration:
CELL-05503: An error was detected during notification. The text
of the associated internal error is: Unknown SMTP host:
my_mail.example.com.
The notification recipient is jane.smith@example.com.

Please verify your SMTP configuration.

CellCLI>
```

7. Execute the following CellCLI command to perform a complete internal check of the cell configuration settings.

```
CellCLI> alter cell validate configuration
Cell qr01cel01 successfully altered

CellCLI>
```

   Note that the `ALTER CELL VALIDATE CONFIGURATION` command does not perform I/O tests against the cell's hard disks and flash modules. You must use the `CALIBRATE` command to perform such tests. The `CALIBRATE` command can only be executed in a CellCLI session initiated by the `root` user.

8. Exit your CellCLI session.

Practice 6-2: Storage Reconfiguration

Overview

In this practice, you alter the Database Machine storage configuration. The approach used in this practice allows the storage reconfiguration to occur while the system is running and databases remain available. The procedure is based on one of the methods described in the section entitled *Resizing Storage Grid Disks* in the *Oracle Exadata Database Machine Owner's Guide*.

In this practice you will reconfigure the `RECO_QR01` ASM disk group. The aim is to resize the disk group and underlying Exadata grid disks so that some of the available free space can be used to create another disk group in the following practice.

Often storage is reconfigured to alter the balance of space allocated to the default DATA and RECO disk groups. In such cases, both disk groups are reconfigured in parallel, with the space freed from one disk group immediately consumed by the other disk group. See the *Oracle Exadata Database Machine Owner's Guide* for more information.

Note: To complete this practice successfully, you must follow the instructions carefully and replicate the commands exactly. Failure to do so could result in unrecoverable damage to your lab environment. Please take care.

Tasks

1. Establish a terminal connection to `qr01db01` as the `grid` user.
2. Using SQL*Plus, connect to ASM as an ASM administrator.

```
[grid@qr01db01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Examine the ASM disk groups in your environment using the following query (or execute the SQL script `/home/oracle/labs/lab06-02-03.sql`). For the reconfiguration approach used in the practice, the amount of free space in the disk group must exceed the associated `REQUIRED_MIRROR_FREE_MB` value. Ensure this is the case for the `RECO_QR01` disk group.

```
SQL> select name, total_mb, free_mb, required_mirror_free_mb
  2  from v$asm_diskgroup;

NAME                            TOTAL_MB    FREE_MB REQUIRED_MIRROR_FREE_MB
------------------------------ ---------- ---------- -----------------------
DATA_QR01                          21312      10596                    7104
DBFS_DG                             5760       4424                    1920
RECO_QR01                          28800      27604                    9600

SQL>
```

4. The following query shows a summary of the space utilization for the disks in each disk group (use the SQL script `/home/oracle/labs/lab06-02-04.sql` if you prefer). For each disk group the query shows the number of associated disks, the size of each disk and the minimum and maximum amount of free space on the disks. Examine the output to ensure that the disk group being reconfigured is reasonably well balanced. In particular ensure that none of the disks are at or near capacity because this may cause problems with later rebalancing operations. If required rebalance the disk group using the `ALTER DISKGROUP ... REBALANCE` command prior to proceeding.

```
SQL> select dg.name, count(*), d.total_mb,
  2  min(d.free_mb) MIN_FREE_MB, max(d.free_mb) MAX_FREE_MB
  3  from v$asm_disk d, v$asm_diskgroup dg
  4  where dg.group_number=d.group_number and d.mount_status='CACHED'
  5  group by dg.name, d.total_mb;

NAME                           COUNT(*)   TOTAL_MB MIN_FREE_MB MAX_FREE_MB
------------------------------ ---------- ---------- ----------- -----------
DBFS_DG                                36        160          88         136
DATA_QR01                              36        592         268         344
RECO_QR01                              36        800         748         788

SQL>
```

5. Check that no ASM rebalance operations are currently active across the cluster.

```
SQL> select * from gv$asm_operation;

no rows selected

SQL>
```

6. Drop the disks associated with the cell `qr01cel01` (use the SQL script `/home/oracle/labs/lab06-02-06.sql` if you prefer). Note that a disk group rebalance is specified in order to maintain data redundancy.

```
SQL> alter diskgroup reco_qr01
  2  drop disks in failgroup qr01cel01
  3  rebalance power 11;

Diskgroup altered.

SQL>
```

7. Monitor the rebalance operation using the following query.

```
SQL> select * from gv$asm_operation;

   INST_ID GROUP_NUMBER OPERA STAT      POWER     ACTUAL      SOFAR   EST_WORK
---------- ------------ ----- ---- ---------- ---------- ---------- ----------
  EST_RATE EST_MINUTES ERROR_CODE
---------- ---------- ---------------------------------------------
         1            3 REBAL RUN          11         11         60        199
       225           0


SQL>
```

8. Periodically repeat the query to monitor the rebalance operation. When the query returns no results, the rebalance operation is completed. **Do not proceed to the next step until the rebalance operation completes.**

```
SQL> select * from gv$asm_operation;

no rows selected

SQL>
```

9. Use the following query (or execute the SQL script `/home/oracle/labs/lab06-02-09.sql`) to confirm that the disks are dropped (`HEADER_STATUS=FORMER` and `MOUNT_STATUS=CLOSED`).

```
SQL> select path, free_mb, header_status, mount_status
  2  from v$asm_disk
  3  where path like '%RECO_QR01%cel01';

PATH
--------------------------------------------------------------------------------
   FREE_MB HEADER_STATU MOUNT_S
---------- ------------ -------
o/192.168.1.103/RECO_QR01_CD_11_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_08_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_00_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_06_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_10_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_01_qr01cel01
```

```
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_04_qr01cel01

PATH
--------------------------------------------------------------------------------
   FREE_MB HEADER_STATU MOUNT_S
---------- ------------ -------
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_07_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_05_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_09_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_03_qr01cel01
         0 FORMER       CLOSED

o/192.168.1.103/RECO_QR01_CD_02_qr01cel01
         0 FORMER       CLOSED


12 rows selected.

SQL>
```

10. Establish a separate terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user. Maintain your ASM administrator SQL session as you will require this throughout the rest of the practice.
11. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

12. Examine the grid disks on `qr01cel01`. Notice the `ASMModeStatus` for the dropped disks.

```
CellCLI> list griddisk attributes name, size, ASMModeStatus
         DATA_QR01_CD_00_qr01cel01       592M    ONLINE
         DATA_QR01_CD_01_qr01cel01       592M    ONLINE
         DATA_QR01_CD_02_qr01cel01       592M    ONLINE
         DATA_QR01_CD_03_qr01cel01       592M    ONLINE
         DATA_QR01_CD_04_qr01cel01       592M    ONLINE
         DATA_QR01_CD_05_qr01cel01       592M    ONLINE
         DATA_QR01_CD_06_qr01cel01       592M    ONLINE
         DATA_QR01_CD_07_qr01cel01       592M    ONLINE
         DATA_QR01_CD_08_qr01cel01       592M    ONLINE
```

```
         DATA_QR01_CD_09_qr01cel01       592M    ONLINE
         DATA_QR01_CD_10_qr01cel01       592M    ONLINE
         DATA_QR01_CD_11_qr01cel01       592M    ONLINE
         DBFS_DG_CD_00_qr01cel01         160M    ONLINE
         DBFS_DG_CD_01_qr01cel01         160M    ONLINE
         DBFS_DG_CD_02_qr01cel01         160M    ONLINE
         DBFS_DG_CD_03_qr01cel01         160M    ONLINE
         DBFS_DG_CD_04_qr01cel01         160M    ONLINE
         DBFS_DG_CD_05_qr01cel01         160M    ONLINE
         DBFS_DG_CD_06_qr01cel01         160M    ONLINE
         DBFS_DG_CD_07_qr01cel01         160M    ONLINE
         DBFS_DG_CD_08_qr01cel01         160M    ONLINE
         DBFS_DG_CD_09_qr01cel01         160M    ONLINE
         DBFS_DG_CD_10_qr01cel01         160M    ONLINE
         DBFS_DG_CD_11_qr01cel01         160M    ONLINE
         RECO_QR01_CD_00_qr01cel01       800M    UNUSED
         RECO_QR01_CD_01_qr01cel01       800M    UNUSED
         RECO_QR01_CD_02_qr01cel01       800M    UNUSED
         RECO_QR01_CD_03_qr01cel01       800M    UNUSED
         RECO_QR01_CD_04_qr01cel01       800M    UNUSED
         RECO_QR01_CD_05_qr01cel01       800M    UNUSED
         RECO_QR01_CD_06_qr01cel01       800M    UNUSED
         RECO_QR01_CD_07_qr01cel01       800M    UNUSED
         RECO_QR01_CD_08_qr01cel01       800M    UNUSED
         RECO_QR01_CD_09_qr01cel01       800M    UNUSED
         RECO_QR01_CD_10_qr01cel01       800M    UNUSED
         RECO_QR01_CD_11_qr01cel01       800M    UNUSED

CellCLI>
```

13. Drop the grid disks on `qr01cel01` previously associated with the `RECO_QR01` disk group.

```
CellCLI> drop griddisk all prefix=reco_qr01
GridDisk RECO_QR01_CD_00_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_01_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_02_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_03_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_04_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_05_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_06_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_07_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_08_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_09_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_10_qr01cel01 successfully dropped
GridDisk RECO_QR01_CD_11_qr01cel01 successfully dropped

CellCLI>
```

14. Create a new set of grid disks. Use the same grid disk prefix as before, but specify a size of 480 MB for each grid disk.

```
CellCLI> create griddisk all harddisk prefix=RECO_QR01, size=480M
GridDisk RECO_QR01_CD_00_qr01cel01 successfully created
GridDisk RECO_QR01_CD_01_qr01cel01 successfully created
GridDisk RECO_QR01_CD_02_qr01cel01 successfully created
GridDisk RECO_QR01_CD_03_qr01cel01 successfully created
GridDisk RECO_QR01_CD_04_qr01cel01 successfully created
GridDisk RECO_QR01_CD_05_qr01cel01 successfully created
GridDisk RECO_QR01_CD_06_qr01cel01 successfully created
GridDisk RECO_QR01_CD_07_qr01cel01 successfully created
GridDisk RECO_QR01_CD_08_qr01cel01 successfully created
GridDisk RECO_QR01_CD_09_qr01cel01 successfully created
GridDisk RECO_QR01_CD_10_qr01cel01 successfully created
GridDisk RECO_QR01_CD_11_qr01cel01 successfully created

CellCLI>
```

15. Re-examine the grid disk on `qr01cel01`. Notice the reconfigured grid disks with the `RECO_QR01` prefix.

```
CellCLI> list griddisk attributes name, size, ASMModeStatus
         DATA_QR01_CD_00_qr01cel01       592M    ONLINE
         DATA_QR01_CD_01_qr01cel01       592M    ONLINE
         DATA_QR01_CD_02_qr01cel01       592M    ONLINE
         DATA_QR01_CD_03_qr01cel01       592M    ONLINE
         DATA_QR01_CD_04_qr01cel01       592M    ONLINE
         DATA_QR01_CD_05_qr01cel01       592M    ONLINE
         DATA_QR01_CD_06_qr01cel01       592M    ONLINE
         DATA_QR01_CD_07_qr01cel01       592M    ONLINE
         DATA_QR01_CD_08_qr01cel01       592M    ONLINE
         DATA_QR01_CD_09_qr01cel01       592M    ONLINE
         DATA_QR01_CD_10_qr01cel01       592M    ONLINE
         DATA_QR01_CD_11_qr01cel01       592M    ONLINE
         DBFS_DG_CD_00_qr01cel01         160M    ONLINE
         DBFS_DG_CD_01_qr01cel01         160M    ONLINE
         DBFS_DG_CD_02_qr01cel01         160M    ONLINE
         DBFS_DG_CD_03_qr01cel01         160M    ONLINE
         DBFS_DG_CD_04_qr01cel01         160M    ONLINE
         DBFS_DG_CD_05_qr01cel01         160M    ONLINE
         DBFS_DG_CD_06_qr01cel01         160M    ONLINE
         DBFS_DG_CD_07_qr01cel01         160M    ONLINE
         DBFS_DG_CD_08_qr01cel01         160M    ONLINE
         DBFS_DG_CD_09_qr01cel01         160M    ONLINE
         DBFS_DG_CD_10_qr01cel01         160M    ONLINE
         DBFS_DG_CD_11_qr01cel01         160M    ONLINE
         RECO_QR01_CD_00_qr01cel01       480M    UNUSED
         RECO_QR01_CD_01_qr01cel01       480M    UNUSED
         RECO_QR01_CD_02_qr01cel01       480M    UNUSED
         RECO_QR01_CD_03_qr01cel01       480M    UNUSED
         RECO_QR01_CD_04_qr01cel01       480M    UNUSED
```

```
         RECO_QR01_CD_05_qr01cel01       480M    UNUSED
         RECO_QR01_CD_06_qr01cel01       480M    UNUSED
         RECO_QR01_CD_07_qr01cel01       480M    UNUSED
         RECO_QR01_CD_08_qr01cel01       480M    UNUSED
         RECO_QR01_CD_09_qr01cel01       480M    UNUSED
         RECO_QR01_CD_10_qr01cel01       480M    UNUSED
         RECO_QR01_CD_11_qr01cel01       480M    UNUSED

CellCLI>
```

16. Exit your CellCLI session but keep your terminal session open. You will require a terminal session connected to `qr01cel01` as the `celladmin` user later in the practice.

```
CellCLI> exit
quitting

[celladmin@qr01cel01 ~]$
```

17. Back in your ASM administrator SQL session, re-execute the following query (or execute the SQL script `/home/oracle/labs/lab06-02-09.sql`). Notice that the reconfigured grid disks are listed with `HEADER_STATUS=CANDIDATE`.

```
SQL> select path, free_mb, header_status, mount_status
  2  from v$asm_disk
  3  where path like '%RECO_QR01%cel01';

PATH
--------------------------------------------------------------------------------
   FREE_MB HEADER_STATU MOUNT_S
---------- ------------ -------
o/192.168.1.103/RECO_QR01_CD_04_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_00_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_02_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_09_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_03_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_05_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_08_qr01cel01

PATH
--------------------------------------------------------------------------------
```

```
   FREE_MB HEADER_STATU MOUNT_S
---------- ------------ -------
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_11_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_07_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_06_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_10_qr01cel01
         0 CANDIDATE    CLOSED

o/192.168.1.103/RECO_QR01_CD_01_qr01cel01
         0 CANDIDATE    CLOSED


12 rows selected.

SQL>
```

18. Add the reconfigured grid disks back into the `RECO_QR01` disk group, and at the same time drop the disks associated with the cell `qr01cel02` (use the SQL script `/home/oracle/labs/lab06-02-18.sql` if you prefer).

```
SQL> alter diskgroup reco_qr01 add disk
  2  'o/192.168.1.103/RECO_QR01_CD_00_qr01cel01',
  3  'o/192.168.1.103/RECO_QR01_CD_01_qr01cel01',
  4  'o/192.168.1.103/RECO_QR01_CD_02_qr01cel01',
  5  'o/192.168.1.103/RECO_QR01_CD_03_qr01cel01',
  6  'o/192.168.1.103/RECO_QR01_CD_04_qr01cel01',
  7  'o/192.168.1.103/RECO_QR01_CD_05_qr01cel01',
  8  'o/192.168.1.103/RECO_QR01_CD_06_qr01cel01',
  9  'o/192.168.1.103/RECO_QR01_CD_07_qr01cel01',
 10  'o/192.168.1.103/RECO_QR01_CD_08_qr01cel01',
 11  'o/192.168.1.103/RECO_QR01_CD_09_qr01cel01',
 12  'o/192.168.1.103/RECO_QR01_CD_10_qr01cel01',
 13  'o/192.168.1.103/RECO_QR01_CD_11_qr01cel01'
 14  drop disks in failgroup qr01cel02
 15  rebalance power 11;

Diskgroup altered.

SQL>
```

19. Monitor the rebalance operation by using the following query.

```
SQL> select * from gv$asm_operation;

   INST_ID GROUP_NUMBER OPERA STAT      POWER     ACTUAL      SOFAR   EST_WORK
---------- ------------ ----- ---- ---------- ---------- ---------- ----------
  EST_RATE EST_MINUTES ERROR_CODE
---------- ----------- -----------------------------------------------
         1           3 REBAL RUN          11         11         78        145
       162           0


SQL>
```

20. Periodically repeat the query to monitor the rebalance operation. When the query returns no results the rebalance operation is completed. **Do not proceed to the next step until the rebalance operation completes.**

```
SQL> select * from gv$asm_operation;

no rows selected

SQL>
```

21. Re-execute the query from step 4 (use the SQL script `/home/oracle/labs/lab06-02-04.sql` if you prefer). Now you can see the `RECO_QR01` disk group in a partially reconfigured state. At this point the storage associated with the `RECO_QR01` disk group has been reconfigured on `qr01cel01` (12 disks) and the disks on `qr01cel02` have been dropped.

```
SQL> select dg.name, count(*), d.total_mb,
  2  min(d.free_mb) MIN_FREE_MB, max(d.free_mb) MAX_FREE_MB
  3  from v$asm_disk d, v$asm_diskgroup dg
  4  where dg.group_number=d.group_number and d.mount_status='CACHED'
  5  group by dg.name, d.total_mb;

NAME                           COUNT(*)   TOTAL_MB MIN_FREE_MB MAX_FREE_MB
------------------------------ ---------- ---------- ---------- ----------
DBFS_DG                                36        160         80        136
DATA_QR01                              36        592        268        344
RECO_QR01                              12        800        756        768
RECO_QR01                              12        480        436        444

SQL>
```

22. Establish a separate terminal connection to the `qr01cel02` Exadata cell as the `celladmin` user.

23. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel02 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

24. Examine the grid disks on `qr01cel02`. Notice again the `ASMModeStatus` for the dropped disks.

    Note: If the ASMModeStatus for the `RECO_QR01` prefixed grid disks is `ONLINE` then you have either dropped the wrong disks in step 18 or you are connected to the wrong cell (you should now be connected to `qr01cel02`). In either case, **do not proceed to the next step until you have resolved the problem.**

```
CellCLI> list griddisk attributes name, size, ASMModeStatus
         DATA_QR01_CD_00_qr01cel02       592M    ONLINE
         DATA_QR01_CD_01_qr01cel02       592M    ONLINE
         DATA_QR01_CD_02_qr01cel02       592M    ONLINE
         DATA_QR01_CD_03_qr01cel02       592M    ONLINE
         DATA_QR01_CD_04_qr01cel02       592M    ONLINE
         DATA_QR01_CD_05_qr01cel02       592M    ONLINE
         DATA_QR01_CD_06_qr01cel02       592M    ONLINE
         DATA_QR01_CD_07_qr01cel02       592M    ONLINE
         DATA_QR01_CD_08_qr01cel02       592M    ONLINE
         DATA_QR01_CD_09_qr01cel02       592M    ONLINE
         DATA_QR01_CD_10_qr01cel02       592M    ONLINE
         DATA_QR01_CD_11_qr01cel02       592M    ONLINE
         DBFS_DG_CD_00_qr01cel02         160M    ONLINE
         DBFS_DG_CD_01_qr01cel02         160M    ONLINE
         DBFS_DG_CD_02_qr01cel02         160M    ONLINE
         DBFS_DG_CD_03_qr01cel02         160M    ONLINE
         DBFS_DG_CD_04_qr01cel02         160M    ONLINE
         DBFS_DG_CD_05_qr01cel02         160M    ONLINE
         DBFS_DG_CD_06_qr01cel02         160M    ONLINE
         DBFS_DG_CD_07_qr01cel02         160M    ONLINE
         DBFS_DG_CD_08_qr01cel02         160M    ONLINE
         DBFS_DG_CD_09_qr01cel02         160M    ONLINE
         DBFS_DG_CD_10_qr01cel02         160M    ONLINE
         DBFS_DG_CD_11_qr01cel02         160M    ONLINE
         RECO_QR01_CD_00_qr01cel02       800M    UNUSED
         RECO_QR01_CD_01_qr01cel02       800M    UNUSED
         RECO_QR01_CD_02_qr01cel02       800M    UNUSED
         RECO_QR01_CD_03_qr01cel02       800M    UNUSED
         RECO_QR01_CD_04_qr01cel02       800M    UNUSED
         RECO_QR01_CD_05_qr01cel02       800M    UNUSED
         RECO_QR01_CD_06_qr01cel02       800M    UNUSED
         RECO_QR01_CD_07_qr01cel02       800M    UNUSED
         RECO_QR01_CD_08_qr01cel02       800M    UNUSED
         RECO_QR01_CD_09_qr01cel02       800M    UNUSED
         RECO_QR01_CD_10_qr01cel02       800M    UNUSED
         RECO_QR01_CD_11_qr01cel02       800M    UNUSED

CellCLI>
```

25. Drop the grid disks on `qr01cel02` previously associated with the `RECO_QR01` disk group.

```
CellCLI> drop griddisk all prefix=reco_qr01
GridDisk RECO_QR01_CD_00_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_01_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_02_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_03_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_04_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_05_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_06_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_07_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_08_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_09_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_10_qr01cel02 successfully dropped
GridDisk RECO_QR01_CD_11_qr01cel02 successfully dropped

CellCLI>
```

26. Create a new set of grid disks. Use the same grid disk prefix as before, but specify a size of 480 MB for each grid disk.

```
CellCLI> create griddisk all harddisk prefix=RECO_QR01, size=480M
GridDisk RECO_QR01_CD_00_qr01cel02 successfully created
GridDisk RECO_QR01_CD_01_qr01cel02 successfully created
GridDisk RECO_QR01_CD_02_qr01cel02 successfully created
GridDisk RECO_QR01_CD_03_qr01cel02 successfully created
GridDisk RECO_QR01_CD_04_qr01cel02 successfully created
GridDisk RECO_QR01_CD_05_qr01cel02 successfully created
GridDisk RECO_QR01_CD_06_qr01cel02 successfully created
GridDisk RECO_QR01_CD_07_qr01cel02 successfully created
GridDisk RECO_QR01_CD_08_qr01cel02 successfully created
GridDisk RECO_QR01_CD_09_qr01cel02 successfully created
GridDisk RECO_QR01_CD_10_qr01cel02 successfully created
GridDisk RECO_QR01_CD_11_qr01cel02 successfully created

CellCLI>
```

27. Exit your CellCLI session.

```
CellCLI> exit
quitting

[celladmin@qr01cel02 ~]$
```

28. Back in your ASM administrator SQL session, add the reconfigured grid disks on `qr01cel02` back into the `RECO_QR01` disk group, and at the same time drop the disks associated with the cell `qr01cel03` (use the SQL script `/home/oracle/labs/lab06-02-28.sql` if you prefer).

```
SQL> alter diskgroup reco_qr01 add disk
  2  'o/192.168.1.104/RECO_QR01_CD_00_qr01cel02',
  3  'o/192.168.1.104/RECO_QR01_CD_01_qr01cel02',
  4  'o/192.168.1.104/RECO_QR01_CD_02_qr01cel02',
  5  'o/192.168.1.104/RECO_QR01_CD_03_qr01cel02',
  6  'o/192.168.1.104/RECO_QR01_CD_04_qr01cel02',
  7  'o/192.168.1.104/RECO_QR01_CD_05_qr01cel02',
  8  'o/192.168.1.104/RECO_QR01_CD_06_qr01cel02',
  9  'o/192.168.1.104/RECO_QR01_CD_07_qr01cel02',
 10  'o/192.168.1.104/RECO_QR01_CD_08_qr01cel02',
 11  'o/192.168.1.104/RECO_QR01_CD_09_qr01cel02',
 12  'o/192.168.1.104/RECO_QR01_CD_10_qr01cel02',
 13  'o/192.168.1.104/RECO_QR01_CD_11_qr01cel02'
 14  drop disks in failgroup qr01cel03
 15  rebalance power 11;

Diskgroup altered.

SQL>
```

29. Monitor the rebalance operation as before.

```
SQL> select * from gv$asm_operation;

   INST_ID GROUP_NUMBER OPERA STAT      POWER     ACTUAL      SOFAR   EST_WORK
---------- ------------ ----- ---- ---------- ---------- ---------- ----------
  EST_RATE EST_MINUTES ERROR_CODE
---------- ----------- -------------------------------------------
         1           3 REBAL RUN          11         11         39        125
       131           0


SQL>
```

30. Periodically repeat the query to monitor the rebalance operation. When the query returns no results, the rebalance operation is completed. **Do not proceed to the next step until the rebalance operation completes.**

```
SQL> select * from gv$asm_operation;

no rows selected

SQL>
```

31. Re-execute the query from step 4 (use the SQL script `/home/oracle/labs/lab06-02-04.sql` if you prefer). Now the storage associated with the `RECO_QR01` disk group has been reconfigured on two cells (24 disks on `qr01cel01` and `qr01cel02`) and the disks on the remaining cell (`qr01cel03`) have been dropped.

```
SQL> select dg.name, count(*), d.total_mb,
  2  min(d.free_mb) MIN_FREE_MB, max(d.free_mb) MAX_FREE_MB
  3  from v$asm_disk d, v$asm_diskgroup dg
  4  where dg.group_number=d.group_number and d.mount_status='CACHED'
  5  group by dg.name, d.total_mb;

NAME                           COUNT(*)   TOTAL_MB MIN_FREE_MB MAX_FREE_MB
------------------------------ ---------- ---------- ----------- -----------
DBFS_DG                                36        160          80         136
DATA_QR01                              36        592         268         344
RECO_QR01                              24        480         432         452

SQL>
```

32. Establish a separate terminal connection to the `qr01cel03` Exadata cell as the `celladmin` user.
33. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel03 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

34. Examine the grid disks on `qr01cel03`. Notice again the `ASMModeStatus` for the dropped disks.

    Note: If the ASMModeStatus for the `RECO_QR01` prefixed grid disks is `ONLINE` then you have either dropped the wrong disks in step 28 or you are connected to the wrong cell (you should now be connected to `qr01cel03`). In either case, **do not proceed to the next step until you have resolved the problem.**

```
CellCLI> list griddisk attributes name, size, ASMModeStatus
         DATA_QR01_CD_00_qr01cel03       592M    ONLINE
         DATA_QR01_CD_01_qr01cel03       592M    ONLINE
         DATA_QR01_CD_02_qr01cel03       592M    ONLINE
         DATA_QR01_CD_03_qr01cel03       592M    ONLINE
         DATA_QR01_CD_04_qr01cel03       592M    ONLINE
         DATA_QR01_CD_05_qr01cel03       592M    ONLINE
         DATA_QR01_CD_06_qr01cel03       592M    ONLINE
         DATA_QR01_CD_07_qr01cel03       592M    ONLINE
         DATA_QR01_CD_08_qr01cel03       592M    ONLINE
         DATA_QR01_CD_09_qr01cel03       592M    ONLINE
         DATA_QR01_CD_10_qr01cel03       592M    ONLINE
         DATA_QR01_CD_11_qr01cel03       592M    ONLINE
         DBFS_DG_CD_00_qr01cel03         160M    ONLINE
         DBFS_DG_CD_01_qr01cel03         160M    ONLINE
         DBFS_DG_CD_02_qr01cel03         160M    ONLINE
         DBFS_DG_CD_03_qr01cel03         160M    ONLINE
```

```
         DBFS_DG_CD_04_qr01cel03         160M    ONLINE
         DBFS_DG_CD_05_qr01cel03         160M    ONLINE
         DBFS_DG_CD_06_qr01cel03         160M    ONLINE
         DBFS_DG_CD_07_qr01cel03         160M    ONLINE
         DBFS_DG_CD_08_qr01cel03         160M    ONLINE
         DBFS_DG_CD_09_qr01cel03         160M    ONLINE
         DBFS_DG_CD_10_qr01cel03         160M    ONLINE
         DBFS_DG_CD_11_qr01cel03         160M    ONLINE
         RECO_QR01_CD_00_qr01cel03       800M    UNUSED
         RECO_QR01_CD_01_qr01cel03       800M    UNUSED
         RECO_QR01_CD_02_qr01cel03       800M    UNUSED
         RECO_QR01_CD_03_qr01cel03       800M    UNUSED
         RECO_QR01_CD_04_qr01cel03       800M    UNUSED
         RECO_QR01_CD_05_qr01cel03       800M    UNUSED
         RECO_QR01_CD_06_qr01cel03       800M    UNUSED
         RECO_QR01_CD_07_qr01cel03       800M    UNUSED
         RECO_QR01_CD_08_qr01cel03       800M    UNUSED
         RECO_QR01_CD_09_qr01cel03       800M    UNUSED
         RECO_QR01_CD_10_qr01cel03       800M    UNUSED
         RECO_QR01_CD_11_qr01cel03       800M    UNUSED

CellCLI>
```

35. Drop the grid disks on `qr01cel03` previously associated with the `RECO_QR01` disk group.

```
CellCLI> drop griddisk all prefix=reco_qr01
GridDisk RECO_QR01_CD_00_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_01_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_02_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_03_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_04_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_05_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_06_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_07_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_08_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_09_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_10_qr01cel03 successfully dropped
GridDisk RECO_QR01_CD_11_qr01cel03 successfully dropped

CellCLI>
```

36. Create a new set of grid disks. Use the same grid disk prefix as before, but specify a size of 480 MB for each grid disk.

```
CellCLI> create griddisk all harddisk prefix=RECO_QR01, size=480M
GridDisk RECO_QR01_CD_00_qr01cel03 successfully created
GridDisk RECO_QR01_CD_01_qr01cel03 successfully created
GridDisk RECO_QR01_CD_02_qr01cel03 successfully created
GridDisk RECO_QR01_CD_03_qr01cel03 successfully created
GridDisk RECO_QR01_CD_04_qr01cel03 successfully created
GridDisk RECO_QR01_CD_05_qr01cel03 successfully created
GridDisk RECO_QR01_CD_06_qr01cel03 successfully created
GridDisk RECO_QR01_CD_07_qr01cel03 successfully created
GridDisk RECO_QR01_CD_08_qr01cel03 successfully created
GridDisk RECO_QR01_CD_09_qr01cel03 successfully created
GridDisk RECO_QR01_CD_10_qr01cel03 successfully created
GridDisk RECO_QR01_CD_11_qr01cel03 successfully created

CellCLI>
```

37. Exit your CellCLI session.

```
CellCLI> exit
quitting

[celladmin@qr01cel03 ~]$
```

38. Back in your ASM administrator SQL session, add the reconfigured grid disks on `qr01cel03` back into the `RECO_QR01` disk group (use the SQL script `/home/oracle/labs/lab06-02-38.sql` if you prefer).

```
SQL> alter diskgroup reco_qr01 add disk
  2  'o/192.168.1.105/RECO_QR01_CD_00_qr01cel03',
  3  'o/192.168.1.105/RECO_QR01_CD_01_qr01cel03',
  4  'o/192.168.1.105/RECO_QR01_CD_02_qr01cel03',
  5  'o/192.168.1.105/RECO_QR01_CD_03_qr01cel03',
  6  'o/192.168.1.105/RECO_QR01_CD_04_qr01cel03',
  7  'o/192.168.1.105/RECO_QR01_CD_05_qr01cel03',
  8  'o/192.168.1.105/RECO_QR01_CD_06_qr01cel03',
  9  'o/192.168.1.105/RECO_QR01_CD_07_qr01cel03',
 10  'o/192.168.1.105/RECO_QR01_CD_08_qr01cel03',
 11  'o/192.168.1.105/RECO_QR01_CD_09_qr01cel03',
 12  'o/192.168.1.105/RECO_QR01_CD_10_qr01cel03',
 13  'o/192.168.1.105/RECO_QR01_CD_11_qr01cel03'
 14  rebalance power 11;

Diskgroup altered.

SQL>
```

39. Monitor the rebalance operation as before.

```
SQL> select * from gv$asm_operation;

   INST_ID GROUP_NUMBER OPERA STAT      POWER     ACTUAL      SOFAR   EST_WORK
---------- ------------ ----- ---- ---------- ---------- ---------- ----------
  EST_RATE EST_MINUTES ERROR_CODE
---------- ---------- -------------------------------------------------
         1            3 REBAL RUN          11         11         70        176
       152           0


SQL>
```

40. Periodically repeat the query to monitor the rebalance operation. When the query returns no results the rebalance operation is completed. **Do not proceed to the next step until the rebalance operation completes.**

```
SQL> select * from gv$asm_operation;

no rows selected

SQL>
```

41. Re-execute the query from step 4 (use the SQL script `/home/oracle/labs/lab06-02-04.sql` if you prefer). Now the storage associated with the `RECO_QR01` disk group has been reconfigured on all three cells.

```
SQL> select dg.name, count(*), d.total_mb,
  2  min(d.free_mb) MIN_FREE_MB, max(d.free_mb) MAX_FREE_MB
  3  from v$asm_disk d, v$asm_diskgroup dg
  4  where dg.group_number=d.group_number and d.mount_status='CACHED'
  5  group by dg.name, d.total_mb;

NAME                           COUNT(*)   TOTAL_MB MIN_FREE_MB MAX_FREE_MB
------------------------------ ---------- ---------- ----------- -----------
DBFS_DG                                36        160          80         136
DATA_QR01                              36        592         268         344
RECO_QR01                              36        480         432         460

SQL>
```

In the final part of this practice, the free space created by reconfiguring the `RECO_QR01` disk group will be provisioned into another set of grid disks.

42. Launch the Exadata cell command-line interface (CellCLI) on `qr01cel01`.

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

43. Use the following CellCLI command to show the free space on each cell disk.

```
CellCLI> list celldisk attributes name, freeSpace where freeSpace != 0
         CD_00_qr01cel01         320M
         CD_01_qr01cel01         320M
         CD_02_qr01cel01         320M
         CD_03_qr01cel01         320M
         CD_04_qr01cel01         320M
         CD_05_qr01cel01         320M
         CD_06_qr01cel01         320M
         CD_07_qr01cel01         320M
         CD_08_qr01cel01         320M
         CD_09_qr01cel01         320M
         CD_10_qr01cel01         320M
         CD_11_qr01cel01         320M

CellCLI>
```

44. Create a set of grid disks which consume all of the available free space. Specify `prefix=DATA2_QR01`.

```
CellCLI> create griddisk all harddisk prefix=DATA2_QR01
GridDisk DATA2_QR01_CD_00_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_01_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_02_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_03_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_04_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_05_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_06_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_07_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_08_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_09_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_10_qr01cel01 successfully created
GridDisk DATA2_QR01_CD_11_qr01cel01 successfully created

CellCLI>
```

45. Examine the newly created grid disks. Note that they are now ready to be consumed in an ASM disk group.

```
CellCLI> list griddisk attributes name, size, ASMModeStatus -
> where name like 'DATA2.*'
         DATA2_QR01_CD_00_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_01_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_02_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_03_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_04_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_05_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_06_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_07_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_08_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_09_qr01cel01      320M    UNUSED
         DATA2_QR01_CD_10_qr01cel01      320M    UNUSED
```

```
         DATA2_QR01_CD_11_qr01cel01      320M    UNUSED

CellCLI>
```

46. Exit your CellCLI session but keep your terminal session open.

```
CellCLI> exit
quitting

[celladmin@qr01cel01 ~]$
```

47. In step 44, a set of grid disks was created on `qr01cel01`. Use the following command to create similar grid disks on `qr01cel02` and `qr01cel03`.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel02,qr01cel03 cellcli -e \
> create griddisk all harddisk prefix=DATA2_QR01
qr01cel02: GridDisk DATA2_QR01_CD_00_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_01_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_02_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_03_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_04_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_05_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_06_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_07_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_08_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_09_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_10_qr01cel02 successfully created
qr01cel02: GridDisk DATA2_QR01_CD_11_qr01cel02 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_00_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_01_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_02_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_03_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_04_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_05_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_06_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_07_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_08_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_09_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_10_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_11_qr01cel03 successfully created
[celladmin@qr01cel01 ~]$
```

48. Use the following command to verify the existence of the newly created grid disks on all three cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> list griddisk attributes name, size, ASMModeStatus \
> where name like \'DATA2.*\'"
qr01cel01: DATA2_QR01_CD_00_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_01_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_02_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_03_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_04_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_05_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_06_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_07_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_08_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_09_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_10_qr01cel01    320M    UNUSED
qr01cel01: DATA2_QR01_CD_11_qr01cel01    320M    UNUSED
qr01cel02: DATA2_QR01_CD_00_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_01_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_02_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_03_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_04_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_05_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_06_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_07_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_08_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_09_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_10_qr01cel02    320M    UNUSED
qr01cel02: DATA2_QR01_CD_11_qr01cel02    320M    UNUSED
qr01cel03: DATA2_QR01_CD_00_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_01_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_02_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_03_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_04_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_05_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_06_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_07_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_08_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_09_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_10_qr01cel03    320M    UNUSED
qr01cel03: DATA2_QR01_CD_11_qr01cel03    320M    UNUSED
[celladmin@qr01cel01 ~]$
```

49. Exit your CellCLI and SQL*Plus sessions.

Practice 6-3: Consuming Grid Disks by Using ASM

Overview

In this practice, you consume some newly created Exadata grid disks using ASM.

Assumptions

Before beginning this practice you must complete Practice 6-2. Your ability to complete this practice depends on the existence of the grid disks that are created in practice 6-2.

Tasks

1. Establish a terminal connection to `qr01db01` as the `grid` user.
2. By using SQL*Plus, connect to ASM as an ASM administrator.

```
[grid@qr01db01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

The ASM environment you are using is clustered across the entire Database Machine. All the grid disks defined across all the Exadata cells are visible inside ASM. In the previous practice you configured some grid disks. Now you will consume those grid disks in different ways using ASM.

3. Execute the following query (or execute the SQL script `/home/oracle/labs/lab06-03-03.sql`). Note the capitalization inside the `like` string. The output shows a series of disks having `HEADER_STATUS=CANDIDATE`. This indicates that the disks do not belong to an ASM disk group and are ready to be consumed by ASM. Examine the `PATH` output. Can you determine the Exadata cell that contains each grid disk?

```
SQL> select name, header_status, path from v$asm_disk
  2  where path like 'o/%/DATA2%'
  3  and header_status='CANDIDATE';

NAME                           HEADER_STATU
------------------------------ ------------
PATH
--------------------------------------------------------------------------------
                               CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_07_qr01cel03

                               CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_07_qr01cel01

                               CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_02_qr01cel03

                               CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_00_qr01cel03

                               CANDIDATE
```

```
o/192.168.1.105/DATA2_QR01_CD_08_qr01cel03

                                CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_06_qr01cel03

                                CANDIDATE

NAME                            HEADER_STATU
------------------------------ ------------
PATH
--------------------------------------------------------------------------------
o/192.168.1.105/DATA2_QR01_CD_10_qr01cel03

                                CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_09_qr01cel03

                                CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_03_qr01cel03

                                CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_05_qr01cel03

                                CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_01_qr01cel03

                                CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_04_qr01cel03

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_09_qr01cel02

NAME                            HEADER_STATU
------------------------------ ------------
PATH
--------------------------------------------------------------------------------

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_06_qr01cel02

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_07_qr01cel02

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_11_qr01cel02

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_08_qr01cel02

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_01_qr01cel02
```

```
                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_10_qr01cel02


NAME                            HEADER_STATU
------------------------------ ------------
PATH
--------------------------------------------------------------------------------
                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_03_qr01cel02

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_00_qr01cel02

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_04_qr01cel02

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_02_qr01cel02

                                CANDIDATE
o/192.168.1.104/DATA2_QR01_CD_05_qr01cel02

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_03_qr01cel01

                                CANDIDATE

NAME                            HEADER_STATU
------------------------------ ------------
PATH
--------------------------------------------------------------------------------
o/192.168.1.103/DATA2_QR01_CD_09_qr01cel01

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_08_qr01cel01

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_05_qr01cel01

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_06_qr01cel01

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_11_qr01cel01

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_04_qr01cel01
```

```
                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_02_qr01cel01

NAME                            HEADER_STATU
------------------------------ ------------
PATH
--------------------------------------------------------------------------------

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_10_qr01cel01

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_00_qr01cel01

                                CANDIDATE
o/192.168.1.103/DATA2_QR01_CD_01_qr01cel01

                                CANDIDATE
o/192.168.1.105/DATA2_QR01_CD_11_qr01cel03


36 rows selected.

SQL>
```

Exadata grid disks are consumed inside ASM in two ways. You can add disks to an existing ASM disk group or you can create a new ASM disk group based on Exadata grid disks.

4. Select one of the disks listed in the output to step 3 and add it to the existing `RECO` disk group (use the SQL script `/home/oracle/labs/lab06-03-04.sql` if you prefer). Use the `PATH` of your selected disk to identify it in the `ADD DISK` clause.

```
SQL> alter diskgroup reco_qr01
  2  add disk 'o/192.168.1.103/DATA2_QR01_CD_00_qr01cel01'
  3  rebalance power 0;

Diskgroup altered.

SQL>
```

5. Verify that the disk is added to your disk group using the following query (or execute the SQL script `/home/oracle/labs/lab06-03-05.sql`).

```
SQL> select dg.name GNAME, d.name DNAME, d.header_status, d.path
  2  from v$asm_disk d left outer join
  3  (select * from v$asm_diskgroup where group_number != 0) dg
  4  on d.group_number = dg.group_number
  5  where dg.name is not null and d.path like 'o/%/DATA2%';

GNAME                          DNAME                          HEADER_STATU
------------------------------ ------------------------------ ------------
PATH
--------------------------------------------------------------------------------
RECO_QR01                      DATA2_QR01_CD_00_QR01CEL01     MEMBER
o/192.168.1.103/DATA2_QR01_CD_00_qr01cel01


SQL>
```

6. Remove the recently added disk from your disk group using the following SQL command (or execute the SQL script `/home/oracle/labs/lab06-03-06.sql`). Note the use of `WAIT` to ensure that the statement does not return until the disk is completely removed from the disk group and the associated rebalance operation completes. It may take a few minutes for the command to complete.

```
SQL> alter diskgroup reco_qr01
  2  drop disk DATA2_QR01_CD_00_QR01CEL01
  3  rebalance power 11 wait;

Diskgroup altered.

SQL>
```

7. Create a new ASM disk group consuming all the grid disks created in the previous practice (use the SQL script `/home/oracle/labs/lab06-03-07.sql` if you prefer).

```
SQL> create diskgroup data2_qr01 normal redundancy
  2  disk 'o/*/DATA2_QR01*'
  3  attribute 'compatible.rdbms' = '11.2.0.0.0',
  4  'compatible.asm' = '11.2.0.0.0',
  5  'cell.smart_scan_capable' = 'TRUE',
  6  'au_size' = '4M';

Diskgroup created.

SQL>
```

The newly created disk group can be used to house Oracle data files in the same way as an ASM disk group based on any other storage. To complement the recommended `AU_SIZE` setting of 4 MB, you should set the initial extent size to 8 MB for large segments. The recommended approaches are discussed in the lesson entitled *Optimizing Database Performance with Exadata*.

8. Examine your newly created disk group using the following query (or execute the SQL script `/home/oracle/labs/lab06-03-08.sql`). Note how the grid disks from each different Exadata cell are automatically grouped into separate failure groups.

```
SQL> select d.path, dg.name GNAME, d.failgroup, d.state
  2  from v$asm_disk d, v$asm_diskgroup dg
  3  where d.group_number = dg.group_number
  4  and dg.name = 'DATA2_QR01';

PATH
--------------------------------------------------------------------------------
GNAME                          FAILGROUP                      STATE
------------------------------ ------------------------------ --------
o/192.168.1.105/DATA2_QR01_CD_07_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_11_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_02_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_00_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_08_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_06_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_10_qr01cel03

PATH
--------------------------------------------------------------------------------
GNAME                          FAILGROUP                      STATE
------------------------------ ------------------------------ --------
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_09_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_03_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL
```

```
o/192.168.1.103/DATA2_QR01_CD_09_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_08_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_05_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_06_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

PATH
--------------------------------------------------------------------------------
GNAME                          FAILGROUP                      STATE
------------------------------ ------------------------------ --------

o/192.168.1.103/DATA2_QR01_CD_11_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_04_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_02_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_10_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_00_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.103/DATA2_QR01_CD_01_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL


PATH
--------------------------------------------------------------------------------
GNAME                          FAILGROUP                      STATE
------------------------------ ------------------------------ --------
o/192.168.1.103/DATA2_QR01_CD_07_qr01cel01
DATA2_QR01                     QR01CEL01                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_03_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_05_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_01_qr01cel03
```

```
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.105/DATA2_QR01_CD_04_qr01cel03
DATA2_QR01                     QR01CEL03                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_09_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_06_qr01cel02

PATH
--------------------------------------------------------------------------------
GNAME                          FAILGROUP                      STATE
------------------------------ ------------------------------ --------
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_07_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_11_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_08_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_01_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_10_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_03_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

PATH
--------------------------------------------------------------------------------
GNAME                          FAILGROUP                      STATE
------------------------------ ------------------------------ --------

o/192.168.1.104/DATA2_QR01_CD_00_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_04_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_02_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL

o/192.168.1.104/DATA2_QR01_CD_05_qr01cel02
DATA2_QR01                     QR01CEL02                      NORMAL
```

```
36 rows selected.

SQL>
```

9. Drop the disk group you created in step 7.

```
SQL> drop diskgroup data2_qr01;

Diskgroup dropped.

SQL>
```

10. Exit your SQL*Plus session.

Practice 6-4: Configuring Exadata Storage Security

Overview

In this practice, you configure Exadata storage security.

Assumptions

Before beginning this practice you must complete Practice 6-2. Your ability to complete this practice depends on the existence of the grid disks that are created in practice 6-2.

Tasks

Exadata storage security has two modes; ASM-scoped security and database-scoped security. ASM-scoped security must be implemented before database-scoped security can be configured. In the first part of this practice, you will configure ASM-scoped security across your lab environment.

1. Establish a terminal connection to `qr01db01` as the `grid` user.
2. Using SQL*Plus, connect to ASM as an ASM administrator.

```
[grid@qr01db01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Note the `DB_UNIQUE_NAME` setting for the ASM environment and then exit SQL*Plus.

```
SQL> show parameter unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------
db_unique_name                       string      +ASM
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - 64bit Production...
[oracle@qr01db01 ~]$
```

4. Use the `su` command to assume the privileges of the `root` user. Enter `oracle` when prompted for the password.

```
[grid@qr01db01 ~]$ su
Password: <oracle>
[root@qr01db01 grid]#
```

5. Shut down the Oracle Database, ASM, and Grid Infrastructure. Note that on a Database Machine, you would need to perform this step on every server in the ASM cluster.

```
[root@qr01db01 grid]# crsctl stop crs
CRS-2791: Starting shutdown of Oracle High Availability Services-managed
resources on 'qr01db01'
CRS-2673: Attempting to stop 'ora.crsd' on 'qr01db01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed resources on
'qr01db01'
CRS-2673: Attempting to stop 'ora.oc4j' on 'qr01db01'
```

```
CRS-2673: Attempting to stop 'ora.qr01db02.vip' on 'qr01db01'
...
CRS-2673: Attempting to stop 'ora.net1.network' on 'qr01db01'
CRS-2677: Stop of 'ora.net1.network' on 'qr01db01' succeeded
CRS-2792: Shutdown of Cluster Ready Services-managed resources on 'qr01db01'
has completed
CRS-2677: Stop of 'ora.crsd' on 'qr01db01' succeeded
CRS-2673: Attempting to stop 'ora.drivers.acfs' on 'qr01db01'
...
CRS-2677: Stop of 'ora.diskmon' on 'qr01db01' succeeded
CRS-2677: Stop of 'ora.gpnpd' on 'qr01db01' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed resources on
'qr01db01' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[root@qr01db01 grid]#
```

6. Exit the `root` user session.

```
[root@qr01db01 grid]# exit
exit
[grid@qr01db01 ~]$
```

7. Leave the current terminal session active and establish a separate terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.
8. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

9. Use the `CREATE KEY` command to generate a random hexadecimal key string. Then exit CellCLI.

```
CellCLI> create key
        aecacf517c96683eb33eaff589a59818
CellCLI> exit
quitting

[celladmin@qr01cel01 ~]$
```

10. Use the `ASSIGN KEY` command to assign the security key generated in step 9 to the Oracle ASM cluster on all the cells that you want the Oracle ASM cluster to access. Use the `DB_UNIQUE_NAME` observed earlier. Note that this is case-sensitive.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> assign key for +ASM=\'aecacf517c96683eb33eaff589a59818\'"
qr01cel01: Key for +ASM successfully created
qr01cel02: Key for +ASM successfully created
qr01cel03: Key for +ASM successfully created
[celladmin@qr01cel01 ~]$
```

11. Use the `CREATE GRIDDISK` or `ALTER GRIDDISK` command to configure security on the grid disks you want the Oracle ASM cluster to access. Set the Oracle ASM `DB_UNIQUE_NAME` in the `availableTo` attribute of each grid disk.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> alter griddisk all availableTo=\'+ASM\'"
qr01cel01: GridDisk DATA2_QR01_CD_00_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_01_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_02_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_03_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_04_qr01cel01 successfully altered
...
qr01cel03: GridDisk RECO_QR01_CD_07_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_08_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_09_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_10_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_11_qr01cel03 successfully altered
[celladmin@qr01cel01 ~]$
```

12. Return to your `grid` user session on `qr01db01` and change directory to `/etc/oracle/cell/network-config`.

```
[grid@qr01db01 ~]$ cd /etc/oracle/cell/network-config
[grid@qr01db01 network-config]$
```

13. Create a `cellkey.ora` file containing the key value from step 9 and the `DB_UNIQUE_NAME` for the ASM cluster.

```
[grid@qr01db01 network-config]$ cat << END > cellkey.ora
> key=aecacf517c96683eb33eaff589a59818
> asm=+ASM
> END
[grid@qr01db01 network-config]$
```

14. Confirm the contents of the `cellkey.ora` file.

```
[grid@qr01db01 network-config]$ cat cellkey.ora
key=aecacf517c96683eb33eaff589a59818
asm=+ASM
[grid@qr01db01 network-config]$
```

15. Set the file permissions and verify the settings. Note that on a Database Machine you would need to configure the `cellkey.ora` file on every server in the ASM cluster.

```
[grid@qr01db01 network-config]# chown grid:asmadmin cellkey.ora
[grid@qr01db01 network-config]$ chmod 640 cellkey.ora
[grid@qr01db01 network-config]$ ls -l cellkey.ora
-rw-r----- 1 grid asmadmin 46 Jul 17 20:50 cellkey.ora
[grid@qr01db01 network-config]$
```

16. Use the `su` command to assume the privileges of the `root` user. Enter `oracle` when prompted for the password.

```
[grid@qr01db01 network-config]$ su
Password: <oracle>
[root@qr01db01 network-config]#
```

17. Restart the Oracle Database, ASM and Grid Infrastructure. Note that on a Database Machine you would need to perform this step on every server in the ASM cluster.

```
[root@qr01db01 network-config]# crsctl start crs
CRS-4123: Oracle High Availability Services has been started.
[root@qr01db01 network-config]#
```

18. Verify that all the Oracle cluster resources restart using the following command. Note that you may receive an error message indicating a failure to communicate with a cluster service if you execute the command while the cluster is restarting. You can safely ignore the error message and re-execute the command until all the resources start. **Do not proceed to the next step before all the cluster resources restart.**

```
[root@qr01db01 network-config]# crsctl stat res -w "TARGET = ONLINE" -t
--------------------------------------------------------------------------------
NAME           TARGET  STATE        SERVER                   STATE_DETAILS
--------------------------------------------------------------------------------
Local Resources
--------------------------------------------------------------------------------
ora.DATA_QR01.dg
               ONLINE  ONLINE       qr01db01
ora.DBFS_DG.dg
               ONLINE  ONLINE       qr01db01
ora.LISTENER.lsnr
               ONLINE  ONLINE       qr01db01
ora.RECO_QR01.dg
               ONLINE  ONLINE       qr01db01
ora.asm
               ONLINE  ONLINE       qr01db01                 Started
ora.net1.network
               ONLINE  ONLINE       qr01db01
ora.ons
               ONLINE  ONLINE       qr01db01
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
ora.LISTENER_SCAN1.lsnr
      1        ONLINE  ONLINE       qr01db01
ora.LISTENER_SCAN2.lsnr
      1        ONLINE  ONLINE       qr01db01
ora.LISTENER_SCAN3.lsnr
      1        ONLINE  ONLINE       qr01db01
ora.cvu
      1        ONLINE  ONLINE       qr01db01
ora.dbm.db
      1        ONLINE  ONLINE       qr01db01                 Open
```

```
      2        ONLINE  OFFLINE
ora.oc4j
      1        ONLINE  ONLINE       qr01db01
ora.qr01db01.vip
      1        ONLINE  ONLINE       qr01db01
ora.qr01db02.vip
      1        ONLINE  INTERMEDIATE qr01db01                FAILED OVER
ora.scan1.vip
      1        ONLINE  ONLINE       qr01db01
ora.scan2.vip
      1        ONLINE  ONLINE       qr01db01
ora.scan3.vip
      1        ONLINE  ONLINE       qr01db01

[root@qr01db01 network-config]#
```

19. Exit the `root` user session and change back to the `grid` user home directory.

```
[root@qr01db01 network-config]# exit
exit
[grid@qr01db01 network-config]$ cd
[grid@qr01db01 ~]$
```

ASM-scoped security is now configured. The fact that ASM and Oracle Database restarted shows that the ASM environment can access the grid disks configured on the Exadata storage. To further prove this is the case, you will now create a disk group on some of the grid disks.

20. Using SQL*Plus, connect to ASM as an ASM administrator.

```
[grid@qr01db01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

21. Create a new ASM disk group consuming the grid disks created in a previous practice (use the SQL script `/home/oracle/labs/lab06-04-21.sql` if you prefer).

```
SQL> create diskgroup data2_qr01_asm_sec normal redundancy
  2  disk 'o/*/DATA2_QR01*'
  3  attribute 'compatible.rdbms' = '11.2.0.0.0',
  4  'compatible.asm' = '11.2.0.0.0',
  5  'cell.smart_scan_capable' = 'TRUE',
  6  'au_size' = '4M';

Diskgroup created.

SQL>
```

22. Drop the newly created disk group and exit SQL*Plus.

```
SQL> drop diskgroup data2_qr01_asm_sec;

Diskgroup dropped.

SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - 64bit Production...
[grid@qr01db01 ~]$
```

As expected, the ASM cluster is able to access the Exadata storage using ASM-scoped security. Now imagine that another ASM cluster is configured and some of the grid disks are assigned to it. In the next part of this practice you will reconfigure the Exadata storage and see the effect.

23. Leave the current terminal session active and establish a separate terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.
24. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

25. Use the `CREATE KEY` command to generate another random hexadecimal key string. Then exit CellCLI.

```
CellCLI> create key
        4b03b5b2b54c871de54784b8064dabdd
CellCLI> exit
quitting

[celladmin@qr01cel01 ~]$
```

26. Use the `ASSIGN KEY` command to assign the security key generated in step 25 to another Oracle ASM cluster called `+ASM2`.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> assign key for +ASM2=\'4b03b5b2b54c871de54784b8064dabdd\'"
qr01cel01: Key for +ASM2 successfully created
qr01cel02: Key for +ASM2 successfully created
qr01cel03: Key for +ASM2 successfully created
[celladmin@qr01cel01 ~]$
```

27. Confirm the key assignment. Note that each cell now has two key assignments for different ASM clusters.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> list key
qr01cel01: +ASM          aecacf517c96683eb33eaff589a59818
qr01cel01: +ASM2         4b03b5b2b54c871de54784b8064dabdd
qr01cel02: +ASM          aecacf517c96683eb33eaff589a59818
qr01cel02: +ASM2         4b03b5b2b54c871de54784b8064dabdd
```

```
qr01cel03: +ASM          aecacf517c96683eb33eaff589a59818
qr01cel03: +ASM2         4b03b5b2b54c871de54784b8064dabdd
[celladmin@qr01cel01 ~]$
```

28. Drop the grid disks having `prefix=DATA2_QR01`.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> drop griddisk all prefix=DATA2_QR01
qr01cel01: GridDisk DATA2_QR01_CD_00_qr01cel01 successfully dropped
qr01cel01: GridDisk DATA2_QR01_CD_01_qr01cel01 successfully dropped
qr01cel01: GridDisk DATA2_QR01_CD_02_qr01cel01 successfully dropped
qr01cel01: GridDisk DATA2_QR01_CD_03_qr01cel01 successfully dropped
qr01cel01: GridDisk DATA2_QR01_CD_04_qr01cel01 successfully dropped
...
qr01cel03: GridDisk DATA2_QR01_CD_07_qr01cel03 successfully dropped
qr01cel03: GridDisk DATA2_QR01_CD_08_qr01cel03 successfully dropped
qr01cel03: GridDisk DATA2_QR01_CD_09_qr01cel03 successfully dropped
qr01cel03: GridDisk DATA2_QR01_CD_10_qr01cel03 successfully dropped
qr01cel03: GridDisk DATA2_QR01_CD_11_qr01cel03 successfully dropped
[celladmin@qr01cel01 ~]$
```

29. Create a new set of grid disks with the `availableTo` attribute set to `+ASM2`.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> create griddisk all harddisk prefix=DATA2_QR01, availableTo=\'+ASM2\'"
qr01cel01: GridDisk DATA2_QR01_CD_00_qr01cel01 successfully created
qr01cel01: GridDisk DATA2_QR01_CD_01_qr01cel01 successfully created
qr01cel01: GridDisk DATA2_QR01_CD_02_qr01cel01 successfully created
qr01cel01: GridDisk DATA2_QR01_CD_03_qr01cel01 successfully created
qr01cel01: GridDisk DATA2_QR01_CD_04_qr01cel01 successfully created
...
qr01cel03: GridDisk DATA2_QR01_CD_07_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_08_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_09_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_10_qr01cel03 successfully created
qr01cel03: GridDisk DATA2_QR01_CD_11_qr01cel03 successfully created
[celladmin@qr01cel01 ~]$
```

Note that the `ALTER GRIDDISK` command could have been used instead of dropping and recreating the grid disks. However, the `ALTER GRIDDISK` command can only be run against all the grid disks (regardless of prefix) or it can be used to modify an individual grid disk. Hence, using the `ALTER GRIDDISK` command in this case would require 36 separate commands (or equivalent scripting).

30. Confirm the `availableTo` attribute setting for all the grid disks.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> list griddisk attributes name,availableTo
qr01cel01: DATA2_QR01_CD_00_qr01cel01    +ASM2
qr01cel01: DATA2_QR01_CD_01_qr01cel01    +ASM2
qr01cel01: DATA2_QR01_CD_02_qr01cel01    +ASM2
qr01cel01: DATA2_QR01_CD_03_qr01cel01    +ASM2
qr01cel01: DATA2_QR01_CD_04_qr01cel01    +ASM2
...
qr01cel03: RECO_QR01_CD_07_qr01cel03     +ASM
```

```
qr01cel03: RECO_QR01_CD_08_qr01cel03     +ASM
qr01cel03: RECO_QR01_CD_09_qr01cel03     +ASM
qr01cel03: RECO_QR01_CD_10_qr01cel03     +ASM
qr01cel03: RECO_QR01_CD_11_qr01cel03     +ASM
[celladmin@qr01cel01 ~]$
```

ASM-scoped security is now reconfigured. Now see what happens when +ASM attempts to use the grid disks assigned to +ASM2.

31. Return to your qr01db01 terminal session. Using SQL*Plus, connect to ASM as an ASM administrator.

```
[grid@qr01db01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

32. Create an ASM disk group referencing the grid disks assigned to +ASM2 (use the SQL script /home/oracle/labs/lab06-04-21.sql if you prefer). Note that the command fails with an error message indicating that the disks do not exist.

```
SQL> create diskgroup data2_qr01_asm_sec normal redundancy
  2  disk 'o/*/DATA2_QR01*'
  3  attribute 'compatible.rdbms' = '11.2.0.0.0',
  4  'compatible.asm' = '11.2.0.0.0',
  5  'cell.smart_scan_capable' = 'TRUE',
  6  'au_size' = '4M';
create diskgroup data2_qr01_asm_sec normal redundancy
*
ERROR at line 1:
ORA-15018: diskgroup cannot be created
ORA-15031: disk specification 'o/*/DATA2_QR01*' matches no disks



SQL>
```

33. Execute the following query to confirm that the disks are not visible to ASM. Exadata storage security limits the visibility of grid disks to the environments which are allows to access them.

```
SQL> select * from v$asm_disk where path like '%DATA2_QR01%';

no rows selected

SQL>
```

34. Exit your SQL*Plus session.

```
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - 64bit Production...
[grid@qr01db01 ~]$
```

So far you have examined ASM-scoped security. Database-scoped security can only be implemented after ASM-scoped security is already in place. The configuration process for database-scoped security very similar to the process you have already used for ASM-scoped security. In the final part of this practice you will configure database-scoped security for one Oracle Database. In a Database Machine environment with multiple databases you would be required to repeat the process for each database.

35. Establish a terminal connection to `qr01db01` as the `oracle` user.
36. Using SQL*Plus, connect as a database administrator.

```
[oracle@qr01db01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

37. Note the `DB_UNIQUE_NAME` setting for the database and then exit SQL*Plus.

```
SQL> show parameter unique

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------
db_unique_name                       string      dbm
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - 64bit Production...
[oracle@qr01db01 ~]$
```

38. Use the `su` command to assume the privileges of the `root` user. Enter `oracle` when prompted for the password.

```
[oracle@qr01db01 ~]$ su
Password: <oracle>
[root@qr01db01 oracle]#
```

39. Shut down the Oracle Database, ASM and Grid Infrastructure. Note that on a Database Machine you would need to perform this step on every server in the ASM cluster.

```
[root@qr01db01 oracle]# /u01/app/11.2.0/grid/bin/crsctl stop crs
CRS-2791: Starting shutdown of Oracle High Availability Services-managed
resources on 'qr01db01'
CRS-2673: Attempting to stop 'ora.crsd' on 'qr01db01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed resources on
'qr01db01'
CRS-2673: Attempting to stop 'ora.LISTENER.lsnr' on 'qr01db01'
CRS-2673: Attempting to stop 'ora.registry.acfs' on 'qr01db01'
...
```

```
CRS-2673: Attempting to stop 'ora.net1.network' on 'qr01db01'
CRS-2677: Stop of 'ora.net1.network' on 'qr01db01' succeeded
CRS-2792: Shutdown of Cluster Ready Services-managed resources on 'qr01db01'
has completed
CRS-2677: Stop of 'ora.crsd' on 'qr01db01' succeeded
CRS-2673: Attempting to stop 'ora.ctssd' on 'qr01db01'
...
CRS-2673: Attempting to stop 'ora.gpnpd' on 'qr01db01'
CRS-2677: Stop of 'ora.gpnpd' on 'qr01db01' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed resources on
'qr01db01' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[root@qr01db01 oracle]#
```

40. Exit the `root` user session.

```
[root@qr01db01 oracle]# exit
exit
[oracle@qr01db01 ~]$
```

41. Leave the current terminal session active and establish a separate terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.
42. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

43. Use the `CREATE KEY` command to generate a random hexadecimal key string. Then exit CellCLI.

```
CellCLI> create key
        2877208d48fa273d86ee6492cd6fe331
CellCLI> exit
quitting

[celladmin@qr01cel01 ~]$
```

44. Use the `ASSIGN KEY` command to assign the security key generated in step 43 to your Oracle Database on all the cells that you want the database to access. Use the `DB_UNIQUE_NAME` observed in step 37. Note that this is case-sensitive.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> assign key for dbm=\'2877208d48fa273d86ee6492cd6fe331\'"
qr01cel01: Key for dbm successfully created
qr01cel02: Key for dbm successfully created
qr01cel03: Key for dbm successfully created
[celladmin@qr01cel01 ~]$
```

45. Use the `CREATE GRIDDISK` or `ALTER GRIDDISK` command to configure security on the grid disks. For database-scoped security you must set the `availableTo` attribute of each grid disk to include both the ASM environment and the database which are allowed to access the grid disk. Use the following command to make all the grid disks accessible to the `dbm` database. Note that in other environments you would typically assign groups of grid disks to different databases.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> alter griddisk all availableTo=\'+ASM,dbm\'"
qr01cel01: GridDisk DATA2_QR01_CD_00_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_01_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_02_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_03_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_04_qr01cel01 successfully altered
...
qr01cel03: GridDisk RECO_QR01_CD_07_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_08_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_09_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_10_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_11_qr01cel03 successfully altered
[celladmin@qr01cel01 ~]$
```

46. Return to your `oracle` user session on `qr01db01` and create a directory at `$ORACLE_HOME/admin/<DB_UNIQUE_NAME>/pfile` where `<DB_UNIQUE_NAME>` represents the `DB_UNIQUE_NAME` setting for the database. Change into the newly created directory.

```
[oracle@qr01db01 ~]$ mkdir -p $ORACLE_HOME/admin/dbm/pfile
[oracle@qr01db01 ~]$ cd $ORACLE_HOME/admin/dbm/pfile
[oracle@qr01db01 pfile]$
```

47. Create a `cellkey.ora` file containing the database key value from step 43 and the `DB_UNIQUE_NAME` for the ASM cluster associated with the database.

```
[oracle@qr01db01 pfile]$ cat << END > cellkey.ora
> key=2877208d48fa273d86ee6492cd6fe331
> asm=+ASM
> END
[oracle@qr01db01 pfile]$
```

48. Confirm the contents of the `cellkey.ora` file.

```
[oracle@qr01db01 pfile]$ cat cellkey.ora
key=2877208d48fa273d86ee6492cd6fe331
asm=+ASM
[oracle@qr01db01 pfile]$
```

49. Set the file permissions and verify the settings. Note that on a Database Machine you would need to configure the `cellkey.ora` file on every database server associated with the database.

```
[oracle@qr01db01 pfile]$ chmod 600 cellkey.ora
[oracle@qr01db01 pfile]$ ls -l cellkey.ora
-rw------- 1 oracle oinstall 46 Jul 17 21:08 cellkey.ora
[oracle@qr01db01 pfile]$
```

50. Use the `su` command to assume the privileges of the `root` user. Enter `oracle` when prompted for the password.

```
[oracle@qr01db01 pfile]$ su
Password: <oracle>
[root@qr01db01 pfile]#
```

51. Restart the Oracle Database, ASM and Grid Infrastructure. Note that on a Database Machine you would need to perform this step on every server in the ASM cluster.

```
[root@qr01db01 pfile]# /u01/app/11.2.0/grid/bin/crsctl start crs
CRS-4123: Oracle High Availability Services has been started.
[root@qr01db01 pfile]#
```

52. Verify that all the Oracle cluster resources restart using the following command. Note that you may receive an error message indicating a failure to communicate with a cluster service if you execute the command while the cluster is restarting. You can safely ignore the error message and re-execute the command until all the resources start. **Do not proceed to the next step before the cluster restarts.**

```
[root@qr01db01 pfile]# /u01/app/11.2.0/grid/bin/crsctl stat res \
> -w "TARGET = ONLINE" -t
--------------------------------------------------------------------------------
NAME           TARGET  STATE        SERVER                   STATE_DETAILS
--------------------------------------------------------------------------------
Local Resources
--------------------------------------------------------------------------------
ora.DATA_QR01.dg
               ONLINE  ONLINE       qr01db01
ora.DBFS_DG.dg
               ONLINE  ONLINE       qr01db01
ora.LISTENER.lsnr
               ONLINE  ONLINE       qr01db01
ora.RECO_QR01.dg
               ONLINE  ONLINE       qr01db01
ora.asm
               ONLINE  ONLINE       qr01db01                 Started
ora.net1.network
               ONLINE  ONLINE       qr01db01
ora.ons
               ONLINE  ONLINE       qr01db01
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
ora.LISTENER_SCAN1.lsnr
```

```
      1        ONLINE  ONLINE       qr01db01
ora.LISTENER_SCAN2.lsnr
      1        ONLINE  ONLINE       qr01db01
ora.LISTENER_SCAN3.lsnr
      1        ONLINE  ONLINE       qr01db01
ora.cvu
      1        ONLINE  ONLINE       qr01db01
ora.dbm.db
      1        ONLINE  ONLINE       qr01db01                 Open
      2        ONLINE  OFFLINE
ora.oc4j
      1        ONLINE  ONLINE       qr01db01
ora.qr01db01.vip
      1        ONLINE  ONLINE       qr01db01
ora.qr01db02.vip
      1        ONLINE  INTERMEDIATE qr01db01                 FAILED OVER
ora.scan1.vip
      1        ONLINE  ONLINE       qr01db01
ora.scan2.vip
      1        ONLINE  ONLINE       qr01db01
ora.scan3.vip
      1        ONLINE  ONLINE       qr01db01

[root@qr01db01 pfile]#
```

53. Exit the `root` user session and change back to the `oracle` user home directory.

```
[root@qr01db01 pfile]# exit
exit
[oracle@qr01db01 pfile]$ cd
[oracle@qr01db01 ~]$
```

Database-scoped security is now configured. The fact that ASM and Oracle Database restarted shows that the both environments can access the grid disks configured on the Exadata storage.

54. Confirm that the database can access its storage by connecting to the database and executing a query.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL> select count(*) from customers;

  COUNT(*)
----------
   1500000

SQL> exit
```

```
Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - 64bit Production...
[oracle@qr01db01 ~]$
```

In the final part of the practice you will remove the Exadata storage security that you have configured during this practice.

55. Use the `su` command to assume the privileges of the `root` user.

```
[oracle@qr01db01 ~]$ su
Password: <oracle>
[root@qr01db01 oracle]#
```

56. Shut down the Oracle Database, ASM and Grid Infrastructure.

```
[root@qr01db01 oracle]# /u01/app/11.2.0/grid/bin/crsctl stop crs
CRS-2791: Starting shutdown of Oracle High Availability Services-managed
resources on 'qr01db01'
CRS-2673: Attempting to stop 'ora.crsd' on 'qr01db01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed resources on
'qr01db01'
CRS-2673: Attempting to stop 'ora.LISTENER.lsnr' on 'qr01db01'
CRS-2673: Attempting to stop 'ora.registry.acfs' on 'qr01db01'
...
CRS-2673: Attempting to stop 'ora.net1.network' on 'qr01db01'
CRS-2677: Stop of 'ora.net1.network' on 'qr01db01' succeeded
CRS-2792: Shutdown of Cluster Ready Services-managed resources on 'qr01db01'
has completed
CRS-2677: Stop of 'ora.crsd' on 'qr01db01' succeeded
CRS-2673: Attempting to stop 'ora.ctssd' on 'qr01db01'
...
CRS-2673: Attempting to stop 'ora.gpnpd' on 'qr01db01'
CRS-2677: Stop of 'ora.gpnpd' on 'qr01db01' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed resources on
'qr01db01' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[root@qr01db01 oracle]#
```

57. Back in your `celladmin` session on `qr01cel01`, use the following command to clear the `availableTo` grid disk attribute.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> alter griddisk all availableTo=\'\'"
qr01cel01: GridDisk DATA2_QR01_CD_00_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_01_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_02_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_03_qr01cel01 successfully altered
qr01cel01: GridDisk DATA2_QR01_CD_04_qr01cel01 successfully altered
...
qr01cel03: GridDisk RECO_QR01_CD_07_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_08_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_09_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_10_qr01cel03 successfully altered
qr01cel03: GridDisk RECO_QR01_CD_11_qr01cel03 successfully altered
[celladmin@qr01cel01 ~]$
```

58. Clear the key assignment for the `dbm` database.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> assign key for dbm=\'\'"
qr01cel01: Key for dbm successfully dropped
qr01cel02: Key for dbm successfully dropped
qr01cel03: Key for dbm successfully dropped
[celladmin@qr01cel01 ~]$
```

59. Clear the key assignments for `+ASM` and `+ASM2`.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> assign key for +ASM=\'\'"
qr01cel01: Key for +ASM successfully dropped
qr01cel02: Key for +ASM successfully dropped
qr01cel03: Key for +ASM successfully dropped
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 "cellcli -e \
> assign key for +ASM2=\'\'"
qr01cel01: Key for +ASM2 successfully dropped
qr01cel02: Key for +ASM2 successfully dropped
qr01cel03: Key for +ASM2 successfully dropped
[celladmin@qr01cel01 ~]$
```

60. Return to your `root` session and remove the `cellkey.ora` files that you created earlier in the practice.

```
[root@qr01db01 oracle]# rm $ORACLE_HOME/admin/dbm/pfile/cellkey.ora
rm: remove regular file
`/u01/app/oracle/product/11.2.0/dbhome_1/admin/dbm/pfile/cellkey.ora'? y
[root@qr01db01 oracle]# rm /etc/oracle/cell/network-config/cellkey.ora
rm: remove regular file `/etc/oracle/cell/network-config/cellkey.ora'? y
[root@qr01db01 oracle]#
```

61. Restart the Oracle Database, ASM and Grid Infrastructure.

```
[root@qr01db01 pfile]# /u01/app/11.2.0/grid/bin/crsctl start crs
CRS-4123: Oracle High Availability Services has been started.
[root@qr01db01 pfile]#
```

62. Exit all of your terminal sessions.

Practice 6-5: Cell User Accounts

Overview

In this practice, you exercise the privileges available to the `celladmin` and `cellmonitor` Exadata administration accounts.

Tasks

1. Establish a terminal connection to the `qr01cel01` Exadata cell as the `cellmonitor` user. Enter `welcome` as the password when prompted.
2. Launch the Exadata cell command-line interface (CellCLI).

```
[cellmonitor@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

The `cellmonitor` user can only view Exadata cell objects using the CellCLI `LIST` command.

3. Confirm that `cellmonitor` can view the Exadata cell attributes.

```
CellCLI> list cell detail
         name:                   qr01cel01
         bbuTempThreshold:       60
         bbuChargeThreshold:     800
         bmcType:                absent
         cellVersion:            OSS_11.2.3.2.1_LINUX.X64_130109
         cpuCount:               1
         diagHistoryDays:        7
         fanCount:               1/1
         fanStatus:              normal
         flashCacheMode:         WriteThrough
         id:                     8ab50138-a667-4793-a976-c540dc1930c5
         interconnectCount:      3
         interconnect1:          eth1
         iormBoost:              0.0
         ipaddress1:             192.168.1.103/24
         kernelVersion:          2.6.32-400.11.1.el5uek
         makeModel:              Fake hardware
         metricHistoryDays:      7
         notificationMethod:     mail
         notificationPolicy:     critical,warning,clear
         offloadEfficiency:      596.0
         powerCount:             1/1
         powerStatus:            normal
         releaseVersion:         11.2.3.2.1
         releaseTrackingBug:     14522699
         smtpFrom:               "John Doe"
         smtpFromAddr:           john.doe@example.com
         smtpServer:             my_mail.example.com
         smtpToAddr:             jane.smith@example.com
         status:                 online
```

```
         temperatureReading:      0.0
         temperatureStatus:       normal
         upTime:                  0 days, 2:45
         cellsrvStatus:           running
         msStatus:                running
         rsStatus:                running


CellCLI>
```

4. Confirm that `cellmonitor` cannot modify the Exadata cell attributes.

```
CellCLI> alter cell smtpToAddr='admin@example.com'
CELL-01520: This command is not permitted in monitor mode.

CellCLI>
```

5. Confirm that `cellmonitor` cannot create or modify the Exadata cell objects.

```
CellCLI> create celldisk all harddisk
CELL-01520: This command is not permitted in monitor mode.

CellCLI> alter griddisk all comment="Here is a comment"
CELL-01520: This command is not permitted in monitor mode.

CellCLI>
```

6. Establish a terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.
7. Launch the Exadata cell command-line interface (CellCLI).

```
[celladmin@qr01cel01 ~]$ cellcli
CellCLI: Release 11.2.3.2.1 - Production...

CellCLI>
```

In previous practices you have already seen how the `celladmin` user can create, modify and drop Exadata cell objects. In fact, the `celladmin` user can execute any CellCLI command except for the `CALIBRATE` command. The `CALIBRATE` command can only be executed by the `root` user.

8. Confirm that `celladmin` cannot run the `CALIBRATE` command.

```
CellCLI> calibrate
CELL-01522: CALIBRATE must be run as the root user id.

CellCLI>
```

9. Exit your CellCLI sessions.

Practice 6-6: Using the Distributed Command-Line Utility (dcli)

Overview

The distributed command-line utility (dcli) is a utility program that is provided with Database Machine. Its purpose is to provide a means to simultaneously execute monitoring and administration commands across multiple servers.

In earlier practices you used dcli to execute CellCLI commands across multiple Exadata cells. In this practice you will extend your use of dcli by performing the initial configuration required to enable dcli to issue commands to all of your cells from your database server (qr01db01). You will also exercise some additional dcli functions.

Tasks

1. Establish a terminal connection to qr01db01 as the oracle user.
2. Create a file named mycells that contains the names of your Exadata cells on separate lines.

```
[oracle@qr01db01 ~]$ cat << END > mycells
> qr01cel01
> qr01cel02
> qr01cel03
> END
[oracle@qr01db01 ~]$
```

3. Generate a private/public key pair for use with SSH using the ssh-keygen command as shown below. For the sake of simplicity, accept the default key file and just press Enter when you are prompted for a passphrase.

```
[oracle@qr01db01 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/oracle/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oracle/.ssh/id_rsa.
Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.
The key fingerprint is:
26:ee:bd:76:6c:33:20:3f:8b:9f:23:49:77:6a:15:97 oracle@qr01db01
[oracle@qr01db01 ~]$
```

4. Execute the following command to configure SSH user-equivalence between your database server OS account (`oracle`) and the `celladmin` user on the cells specified in the `mycells` file.

   Answer `yes` if you are prompted to acknowledge server authenticity.

```
[oracle@qr01db01 ~]$ dcli -g mycells -k
The authenticity of host 'qr01cel01 (192.0.2.103)' can't be established.
RSA key fingerprint is fb:f9:ec:56:6b:8c:5b:a0:90:82:20:36:51:b8:59:af.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'qr01cel01,192.0.2.103' (RSA) to the list of known
hosts.
celladmin@qr01cel01's password: <welcome>
The authenticity of host 'qr01cel03 (192.0.2.105)' can't be established.
RSA key fingerprint is fb:f9:ec:56:6b:8c:5b:a0:90:82:20:36:51:b8:59:af.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'qr01cel03,192.0.2.105' (RSA) to the list of known
hosts.
celladmin@qr01cel03's password: <welcome>
The authenticity of host 'qr01cel02 (192.0.2.104)' can't be established.
RSA key fingerprint is fb:f9:ec:56:6b:8c:5b:a0:90:82:20:36:51:b8:59:af.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'qr01cel02,192.0.2.104' (RSA) to the list of known
hosts.
celladmin@qr01cel02's password: <welcome>
qr01cel01: ssh key added
qr01cel02: ssh key added
qr01cel03: ssh key added
[oracle@qr01db01 ~]$
```

You have now completed the one-time configuration required to enable `dcli` between the `oracle` user on `qr01db01` and the `celladmin` user on each Exadata cell. After `dcli` is executed once with the `-k` option, subsequent `dcli` commands between the same servers and user accounts do not require the `-k` option and do not require a password.

5. The primary use for `dcli` is to simultaneously execute CellCLI commands across multiple cells. Use the following command to check on the status of your Exadata cells.

```
[oracle@qr01db01 ~]$ dcli -g mycells cellcli -e list cell
qr01cel01: qr01cel01     online
qr01cel02: qr01cel02     online
qr01cel03: qr01cel03     online
[oracle@qr01db01 ~]$
```

6. `dcli` can also be used to execute any non-interactive operating system commands on multiple cells and/or database servers. You can use quotes to surround compound commands and commands which contain pipes. Execute the following example command or construct an alternative command of your own.

```
[oracle@qr01db01 ~]$ dcli -g mycells "cat /proc/meminfo | grep Mem"
qr01cel01: MemTotal:        1993500 kB
qr01cel01: MemFree:           62564 kB
qr01cel02: MemTotal:        1993500 kB
qr01cel02: MemFree:           71484 kB
qr01cel03: MemTotal:        1993500 kB
qr01cel03: MemFree:           73124 kB
[oracle@qr01db01 ~]$
```

7. `dcli` is not just limited to monitoring. It is often used to ensure that consistent settings are applied across multiple systems. Use the following commands to view and adjust IORM settings on your Exadata cells.

```
[oracle@qr01db01 ~]$ dcli -g mycells cellcli -e \
> list iormplan attributes objective
qr01cel01: basic
qr01cel02: basic
qr01cel03: basic
[oracle@qr01db01 ~]$ dcli -g mycells cellcli -e alter iormplan objective=auto
qr01cel01: IORMPLAN successfully altered
qr01cel02: IORMPLAN successfully altered
qr01cel03: IORMPLAN successfully altered
[oracle@qr01db01 ~]$ dcli -g mycells cellcli -e \
> list iormplan attributes objective
qr01cel01: auto
qr01cel02: auto
qr01cel03: auto
[oracle@qr01db01 ~]$ dcli -g mycells cellcli -e alter iormplan objective=basic
qr01cel01: IORMPLAN successfully altered
qr01cel02: IORMPLAN successfully altered
qr01cel03: IORMPLAN successfully altered
[oracle@qr01db01 ~]$ dcli -g mycells cellcli -e \
> list iormplan attributes objective
qr01cel01: basic
qr01cel02: basic
qr01cel03: basic
[oracle@qr01db01 ~]$
```

8. Sometimes `dcli` commands return a lot of output. The following command uses wildcards and a `WHERE` condition to return the current metric observations for small write I/O requests for every disk-based cell disk across all the cells. Execute the command and examine the output for your cells.

```
[oracle@qr01db01 ~]$ dcli -g mycells "cellcli -e list metriccurrent \
> where name like \'CD_IO_RQ_W_S.?\' and metricobjectname like \'CD.*\'"
qr01cel01: CD_IO_RQ_W_SM         CD_00_qr01cel01         13,855 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_01_qr01cel01         1,393 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_02_qr01cel01         1,424 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_03_qr01cel01         2,965 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_04_qr01cel01         4,364 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_05_qr01cel01         1,624 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_06_qr01cel01         1,049 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_07_qr01cel01         3,648 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_08_qr01cel01         1,567 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_09_qr01cel01         2,352 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_10_qr01cel01         1,323 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_11_qr01cel01         1,591 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_00_qr01cel02         14,336 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_01_qr01cel02         4,056 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_02_qr01cel02         846 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_03_qr01cel02         1,732 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_04_qr01cel02         3,031 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_05_qr01cel02         1,150 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_06_qr01cel02         1,862 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_07_qr01cel02         1,506 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_08_qr01cel02         3,420 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_09_qr01cel02         1,283 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_10_qr01cel02         3,313 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_11_qr01cel02         2,835 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_00_qr01cel03         13,564 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_01_qr01cel03         590 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_02_qr01cel03         4,480 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_03_qr01cel03         4,783 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_04_qr01cel03         3,063 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_05_qr01cel03         1,074 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_06_qr01cel03         695 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_07_qr01cel03         5,879 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_08_qr01cel03         562 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_09_qr01cel03         1,449 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_10_qr01cel03         1,269 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_11_qr01cel03         547 IO requests
[oracle@qr01db01 ~]$
```

9. This command is essentially the same as the command you just executed in step 8. It uses the `-r` option along with a regular expression string to specify which output `dcli` should delete from the output. The result is that only the output that does not match the `-r` regular expression is returned to the user. Execute the command and examine the output for your cells.

```
[oracle@qr01db01 ~]$ dcli -g mycells -r '.*CD_0.*' "cellcli -e list \
> metriccurrent where name like \'CD_IO_RQ_W_S.?\' \
> and metricobjectname like \'CD.*\'"
.*CD_0.*: ['qr01cel01', 'qr01cel02', 'qr01cel03']
qr01cel01: CD_IO_RQ_W_SM         CD_10_qr01cel01         1,323 IO requests
qr01cel01: CD_IO_RQ_W_SM         CD_11_qr01cel01         1,591 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_10_qr01cel02         3,313 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_11_qr01cel02         2,835 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_10_qr01cel03         1,269 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_11_qr01cel03         547 IO requests
[oracle@qr01db01 ~]$
```

10. While the `-r` option specifies what to restrict from the `dcli` output, there is no `dcli` option to explicitly define the output that should be returned from a long list. To achieve this aim use the `grep` command in conjunction with `dcli`. Execute the following command and examine the output for your cells.

```
[oracle@qr01db01 ~]$ dcli -g mycells "cellcli -e list metriccurrent \
> where name like \'CD_IO_RQ_W_S.?\' \
> and metricobjectname like \'CD.*\' | grep CD_00"
qr01cel01: CD_IO_RQ_W_SM         CD_00_qr01cel01         14,133 IO requests
qr01cel02: CD_IO_RQ_W_SM         CD_00_qr01cel02         14,571 IO requests
qr01cel03: CD_IO_RQ_W_SM         CD_00_qr01cel03         13,663 IO requests
[oracle@qr01db01 ~]$
```

`dcli` can also be used to copy files to numerous remote systems. You will exercise this capability in the next series of steps.

11. Create a small text file that contains a short message identifying you. Name the file according to your assigned student account. Confirm the existence and contents of the file.

```
[oracle@qr01db01 ~]$ cat << END > message.txt
> Hello World!
> END
[oracle@qr01db01 ~]$ cat message.txt
Hello World!
[oracle@qr01db01 ~]$
```

12. Use `dcli` with the `-f` option to copy your file to the default home directory of the `celladmin` user on your Exadata cells.

```
[oracle@qr01db01 ~]$ dcli -g mycells -f message.txt
[oracle@qr01db01 ~]$
```

13. Use the following command to confirm the success of the operation in step 12.

```
[oracle@qr01db01 ~]$ dcli -g mycells cat message.txt
qr01cel01: Hello World!
qr01cel02: Hello World!
qr01cel03: Hello World!
[oracle@qr01db01 ~]$
```

In addition to copying a file to multiple remote locations, `dcli` can copy a file and execute it simultaneously on the specified remote systems. You will exercise this capability in the next series of steps.

14. Create a simple shell script such as the one shown below. Name the file according to your assigned student account.

```
[oracle@qr01db01 ~]$ cat << END > script.sh
> HST=\`hostname -s\`
> DTE=\`date\`
> echo -n \`cat message.txt\`
> echo " on \${HST} at \${DTE}."
> END
[oracle@qr01db01 ~]$
```

15. Use the `chmod` command to make your newly created script file executable.

```
[oracle@qr01db01 ~]$ chmod +x script.sh
[oracle@qr01db01 ~]$
```

16. Use `dcli` with the `-x` option to copy and run the script you just created.

```
[oracle@qr01db01 ~]$ dcli -g mycells -x script.sh
qr01cel01: Hello World! on qr01cel01 at Wed Jul 17 21:27:28 EDT 2013.
qr01cel02: Hello World! on qr01cel02 at Wed Jul 17 21:27:29 EDT 2013.
qr01cel03: Hello World! on qr01cel03 at Wed Jul 17 21:27:29 EDT 2013.
[oracle@qr01db01 ~]$
```

Note that script files with the `.scl` extension are run by the CellCLI utility on the remote server.

17. Use `dcli` in conjunction with `rm` to delete the files you copied to your cells during this practice. Please be careful not to mistakenly delete any other files.

```
[oracle@qr01db01 ~]$ dcli -g mycells rm message.txt script.sh
[oracle@qr01db01 ~]$
```

Practices for Lesson 7: I/O Resource Management

Chapter 7

Practices for Lesson 7

Practices Overview

There is no practice for Lesson 3.

Practices for Lesson 8: Recommendations for Optimizing Database Performance

Chapter 8

Practices for Lesson 8

Practices Overview

In these practices, you will explore the following performance optimization techniques and technologies:

- Configuring write back flash cache
- Using Exadata Hybrid Columnar Compression
- Testing index elimination

Practice 8-1: Configuring Write Back Flash Cache

Overview

In this practice you will reconfigure Exadata Smart Flash Cache so that write operations can be serviced by flash only, instead of using disks. This mode of operation is known as write back flash cache.

Tasks

Reconfiguring Exadata Storage Servers to enable write back flash cache can be achieved in a rolling manner (one cell at a time) or all-at-once. In this practice you will reconfigure all the cells at once, which requires that Oracle Database and Grid Infrastructure are shut down on all database servers. To configure write back flash cache in a rolling manner some additional steps and checks are recommended. Refer to My Oracle Support bulletin 1500257.1 for details.

1. Establish a terminal connection to `qr01db01` as the `grid` user.
2. Use the `su` command to assume the privileges of the `root` user. Enter `oracle` when prompted for the password.

```
[grid@qr01db01 ~]$ su
Password: <oracle>
[root@qr01db01 grid]#
```

3. Shut down the Oracle Database, ASM and Grid Infrastructure. Note that on a Database Machine you would need to perform this step on every server in the ASM cluster. **Do not proceed to the next step until this step completes.**

```
[root@qr01db01 grid]# crsctl stop crs
CRS-2791: Starting shutdown of Oracle High Availability Services-managed
resources on 'qr01db01'
CRS-2673: Attempting to stop 'ora.crsd' on 'qr01db01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed resources on
'qr01db01'
CRS-2673: Attempting to stop 'ora.oc4j' on 'qr01db01'
CRS-2673: Attempting to stop 'ora.qr01db02.vip' on 'qr01db01'
...
CRS-2673: Attempting to stop 'ora.net1.network' on 'qr01db01'
CRS-2677: Stop of 'ora.net1.network' on 'qr01db01' succeeded
CRS-2792: Shutdown of Cluster Ready Services-managed resources on 'qr01db01'
has completed
CRS-2677: Stop of 'ora.crsd' on 'qr01db01' succeeded
CRS-2673: Attempting to stop 'ora.drivers.acfs' on 'qr01db01'
...
CRS-2677: Stop of 'ora.diskmon' on 'qr01db01' succeeded
CRS-2677: Stop of 'ora.gpnpd' on 'qr01db01' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed resources on
'qr01db01' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[root@qr01db01 grid]#
```

4. Leave the `root` terminal session active and establish a separate terminal connection to the `qr01cel01` Exadata cell as the `celladmin` user.

5. Examine the cell `flashCacheMode` attribute setting on all of the cells. By default, each cell is configured in `WriteThrough` mode, which means that write operations must be persisted to disk regardless of whether or not the data resides inside Exadata Smart Flash Cache.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> list cell attributes flashCacheMode
qr01cel01: WriteThrough
qr01cel02: WriteThrough
qr01cel03: WriteThrough
[celladmin@qr01cel01 ~]$
```

6. Drop the existing Exadata Smart Flash Cache.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> drop flashcache
qr01cel01: Flash cache qr01cel01_FLASHCACHE successfully dropped
qr01cel02: Flash cache qr01cel02_FLASHCACHE successfully dropped
qr01cel03: Flash cache qr01cel03_FLASHCACHE successfully dropped
[celladmin@qr01cel01 ~]$
```

7. Stop `cellsrv` on all of the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> alter cell shutdown services cellsrv
qr01cel01:
qr01cel01: Stopping CELLSRV services...
qr01cel01: The SHUTDOWN of CELLSRV services was successful.
qr01cel02:
qr01cel02: Stopping CELLSRV services...
qr01cel02: The SHUTDOWN of CELLSRV services was successful.
qr01cel03:
qr01cel03: Stopping CELLSRV services...
qr01cel03: The SHUTDOWN of CELLSRV services was successful.
[celladmin@qr01cel01 ~]$
```

8. Commencing with Exadata storage server release 11.2.3.2.0, the `flashCacheMode` attribute can be set to `WriteBack`, which means that write operations can be serviced by flash instead of using disks. Enable write back flash cache on all of the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> alter cell flashCacheMode = WriteBack
qr01cel01: Cell qr01cel01 successfully altered
qr01cel02: Cell qr01cel02 successfully altered
qr01cel03: Cell qr01cel03 successfully altered
[celladmin@qr01cel01 ~]$
```

9. Restart `cellsrv` on all of the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> alter cell startup services cellsrv
qr01cel01:
qr01cel01: Starting CELLSRV services...
qr01cel01: The STARTUP of CELLSRV services was successful.
qr01cel02:
qr01cel02: Starting CELLSRV services...
```

```
qr01cel02: The STARTUP of CELLSRV services was successful.
qr01cel03:
qr01cel03: Starting CELLSRV services...
qr01cel03: The STARTUP of CELLSRV services was successful.
[celladmin@qr01cel01 ~]$
```

10. Re-create Exadata Smart Flash Cache on all the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> create flashcache all
qr01cel01: Flash cache qr01cel01_FLASHCACHE successfully created
qr01cel02: Flash cache qr01cel02_FLASHCACHE successfully created
qr01cel03: Flash cache qr01cel03_FLASHCACHE successfully created
[celladmin@qr01cel01 ~]$
```

11. Verify the cell `flashCacheMode` attribute setting on all of the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> list cell attributes flashCacheMode
qr01cel01: WriteBack
qr01cel02: WriteBack
qr01cel03: WriteBack
[celladmin@qr01cel01 ~]$
```

Now that write back flash cache is enabled, each cache may contain updated data that is different from the data copy held on disk. This is called dirty data, and you can monitor the amount of dirty data inside Exadata Smart Flash Cache using cell metrics.

12. Use the following command to view the amount of dirty data currently inside Exadata Smart Flash Cache on each cell. Because the caches have only just been created and no databases are currently using the cells, the amount should be zero.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> list metriccurrent FC_BY_DIRTY
qr01cel01: FC_BY_DIRTY   FLASHCACHE      0.000 MB
qr01cel02: FC_BY_DIRTY   FLASHCACHE      0.000 MB
qr01cel03: FC_BY_DIRTY   FLASHCACHE      0.000 MB
[celladmin@qr01cel01 ~]$
```

13. Return to you root terminal session and restart Grid Infrastructure, ASM and Oracle Database.

```
[root@qr01db01 grid]# crsctl start crs
CRS-4123: Oracle High Availability Services has been started.
[root@qr01db01 grid]#
```

14. Wait for a few minutes until your database restarts. Execute the following command to monitor the status of the database. Note that you may see an error similar to that shown below if you execute this command before clusterware is restarted. **Proceed to the next stop only after your database is started on `qr01db01`.**

```
[root@qr01db01 grid]# srvctl status database -d dbm
PRCD-1027 : Failed to retrieve database dbm
PRCR-1115 : Failed to find entities of type resource that match
filters ((NAME == ora.dbm.db) && (TYPE == ora.database.type))
and contain attributes VERSION,ORACLE_HOME,DATABASE_TYPE
Cannot communicate with crsd
[root@qr01db01 grid]# srvctl status database -d dbm
Instance dbm1 is not running on node qr01db01
Instance dbm2 is not running on node qr01db02
[root@qr01db01 grid]# srvctl status database -d dbm
Instance dbm1 is running on node qr01db01
Instance dbm2 is not running on node qr01db02
[root@qr01db01 grid]#
```

15. Return to your `celladmin` terminal session and view the amount of dirty data in each cell cache.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> list metriccurrent FC_BY_DIRTY
qr01cel01: FC_BY_DIRTY    FLASHCACHE     0.961 MB
qr01cel02: FC_BY_DIRTY    FLASHCACHE     0.961 MB
qr01cel03: FC_BY_DIRTY    FLASHCACHE     0.922 MB
[celladmin@qr01cel01 ~]$
```

At this point, you have configured write back flash cache and you have confirmed that it is being used. Over time you would see more dirty data reported in the cache as you transact against your databases.

In the final part of this practice, you will reconfigure the cells once more and revert back to write through flash cache.

16. Return to you root terminal session and shut down the Oracle Database, ASM, and Grid Infrastructure. **Do not proceed to the next step until this step completes.**

```
[root@qr01db01 grid]# crsctl stop crs
CRS-2791: Starting shutdown of Oracle High Availability Services-managed
resources on 'qr01db01'
CRS-2673: Attempting to stop 'ora.crsd' on 'qr01db01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed resources on
'qr01db01'
CRS-2673: Attempting to stop 'ora.oc4j' on 'qr01db01'
CRS-2673: Attempting to stop 'ora.qr01db02.vip' on 'qr01db01'
...
CRS-2673: Attempting to stop 'ora.net1.network' on 'qr01db01'
CRS-2677: Stop of 'ora.net1.network' on 'qr01db01' succeeded
CRS-2792: Shutdown of Cluster Ready Services-managed resources on 'qr01db01'
has completed
CRS-2677: Stop of 'ora.crsd' on 'qr01db01' succeeded
CRS-2673: Attempting to stop 'ora.drivers.acfs' on 'qr01db01'
```

```
...
CRS-2677: Stop of 'ora.diskmon' on 'qr01db01' succeeded
CRS-2677: Stop of 'ora.gpnpd' on 'qr01db01' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed resources on
'qr01db01' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[root@qr01db01 grid]#
```

Switching back to write through flash cache involves essentially the same steps as switching to write back flash cache. However, because write back flash cache is currently enabled note that you must first flush the flash cache before dropping it. Flushing Exadata Smart Flash Cache ensures that all modified data is written to disk.

17. Flush the write back flash cache on all of the cells. Note that the flush operation may take a few minutes to complete. If you receive a message indicating that the flush operation timed out for some of the flash disks, repeat this step until your output matches the example shown below.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> alter flashcache all flush
qr01cel01: Flash cache qr01cel01_FLASHCACHE altered successfully
qr01cel02: Flash cache qr01cel02_FLASHCACHE altered successfully
qr01cel03: Flash cache qr01cel03_FLASHCACHE altered successfully
```

18. Drop the Exadata Smart Flash Cache on all of the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> drop flashcache
qr01cel01: Flash cache qr01cel01_FLASHCACHE successfully dropped
qr01cel02: Flash cache qr01cel02_FLASHCACHE successfully dropped
qr01cel03: Flash cache qr01cel03_FLASHCACHE successfully dropped
[celladmin@qr01cel01 ~]$
```

19. Stop `cellsrv` on all of the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> alter cell shutdown services cellsrv
qr01cel01:
qr01cel01: Stopping CELLSRV services...
qr01cel01: The SHUTDOWN of CELLSRV services was successful.
qr01cel02:
qr01cel02: Stopping CELLSRV services...
qr01cel02: The SHUTDOWN of CELLSRV services was successful.
qr01cel03:
qr01cel03: Stopping CELLSRV services...
qr01cel03: The SHUTDOWN of CELLSRV services was successful.
[celladmin@qr01cel01 ~]$
```

20. Set the `flashCacheMode` attribute back to `WriteThrough`.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> alter cell flashCacheMode = WriteThrough
qr01cel01: Cell qr01cel01 successfully altered
qr01cel02: Cell qr01cel02 successfully altered
qr01cel03: Cell qr01cel03 successfully altered
 [celladmin@qr01cel01 ~]$
```

21. Restart `cellsrv` on all of the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> alter cell startup services cellsrv
qr01cel01:
qr01cel01: Starting CELLSRV services...
qr01cel01: The STARTUP of CELLSRV services was successful.
qr01cel02:
qr01cel02: Starting CELLSRV services...
qr01cel02: The STARTUP of CELLSRV services was successful.
qr01cel03:
qr01cel03: Starting CELLSRV services...
qr01cel03: The STARTUP of CELLSRV services was successful.
[celladmin@qr01cel01 ~]$
```

22. Re-create Exadata Smart Flash Cache on all the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> create flashcache all
qr01cel01: Flash cache qr01cel01_FLASHCACHE successfully created
qr01cel02: Flash cache qr01cel02_FLASHCACHE successfully created
qr01cel03: Flash cache qr01cel03_FLASHCACHE successfully created
[celladmin@qr01cel01 ~]$
```

23. Verify the cell `flashCacheMode` attribute setting on all of the cells.

```
[celladmin@qr01cel01 ~]$ dcli -c qr01cel01,qr01cel02,qr01cel03 cellcli -e \
> list cell attributes flashCacheMode
qr01cel01: WriteThrough
qr01cel02: WriteThrough
qr01cel03: WriteThrough
[celladmin@qr01cel01 ~]$
```

24. Return to you root terminal session and restart Grid Infrastructure, ASM, and Oracle Database.

```
[root@qr01db01 grid]# crsctl start crs
CRS-4123: Oracle High Availability Services has been started.
[root@qr01db01 grid]#
```

25. Wait for a few minutes until your database restarts. Execute the following command to monitor the status of the database. **Proceed to the next practice only after your database is started on `qr01db01`.**

```
[root@qr01db01 grid]# srvctl status database -d dbm
Instance dbm1 is running on node qr01db01
Instance dbm2 is not running on node qr01db02
[root@qr01db01 grid]#
```

26. Exit all of your terminal sessions.

Practice 8-2: Using Exadata Hybrid Columnar Compression

Overview

In this practice, you will examine the performance of Exadata Hybrid Columnar Compression. You will compare predicted and actual compression ratios using an example dataset. You will also examine how bulk data loading and query operations are affected using the different compression modes.

Tasks

1. Establish a terminal connection to your database server as the `oracle` user.
2. Connect to your database with SQL*Plus. Log in as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Configure the session to display server output and timing statistics.

```
SQL> set serveroutput on
SQL> set timing on
SQL>
```

4. Use `dbms_compression.get_compression_ratio` to predict the expected compression ratio for the `MYCUSTOMERS` table using all the different Exadata Hybrid Columnar Compression modes (use the SQL script `/home/oracle/labs/lab08-02-04.sql` if you prefer). Note that this step can take more than 10 minutes to complete.

```
SQL> declare
  2    b_cmp number;
  3    b_ucmp number;
  4    r_cmp number;
  5    r_ucmp number;
  6    cmp_ratio number(6,2);
  7    cmp_type varchar2(1024);
  8  begin
  9    dbms_compression.get_compression_ratio('USERS','SALES',
 10      'MYCUSTOMERS',NULL,DBMS_COMPRESSION.COMP_FOR_QUERY_LOW,
 11      b_cmp,b_ucmp, r_cmp, r_ucmp, cmp_ratio, cmp_type);
 12    dbms_output.put_line('Table: MYCUSTOMERS');
 13    dbms_output.put_line('Compression Ratio: '||cmp_ratio);
 14    dbms_output.put_line('Compression Type: '|| cmp_type);
 15  end;
 16  /
Compression Advisor self-check validation successful. select count(*) on both
Uncompressed and EHCC Compressed format = 1000001 rows
Table: MYCUSTOMERS
Compression Ratio: 4.2
Compression Type: "Compress For Query Low"
```

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:49.58
SQL> declare
  2    b_cmp number;
  3    b_ucmp number;
  4    r_cmp number;
  5    r_ucmp number;
  6    cmp_ratio number(6,2);
  7    cmp_type varchar2(1024);
  8  begin
  9    dbms_compression.get_compression_ratio('USERS','SALES',
 10      'MYCUSTOMERS',NULL,DBMS_COMPRESSION.COMP_FOR_QUERY_HIGH,
 11      b_cmp,b_ucmp, r_cmp, r_ucmp, cmp_ratio, cmp_type);
 12    dbms_output.put_line('Table: MYCUSTOMERS');
 13    dbms_output.put_line('Compression Ratio: '||cmp_ratio);
 14    dbms_output.put_line('Compression Type: '|| cmp_type);
 15  end;
 16  /
Compression Advisor self-check validation successful. select count(*) on both
Uncompressed and EHCC Compressed format = 1000001 rows
Table: MYCUSTOMERS
Compression Ratio: 7
Compression Type: "Compress For Query High"

PL/SQL procedure successfully completed.

Elapsed: 00:02:47.54
SQL> declare
  2    b_cmp number;
  3    b_ucmp number;
  4    r_cmp number;
  5    r_ucmp number;
  6    cmp_ratio number(6,2);
  7    cmp_type varchar2(1024);
  8  begin
  9    dbms_compression.get_compression_ratio('USERS', 'SALES',
 10      'MYCUSTOMERS',NULL,DBMS_COMPRESSION.COMP_FOR_ARCHIVE_LOW,
 11      b_cmp,b_ucmp, r_cmp, r_ucmp, cmp_ratio, cmp_type);
 12    dbms_output.put_line('Table: MYCUSTOMERS');
 13    dbms_output.put_line('Compression Ratio: '||cmp_ratio);
 14    dbms_output.put_line('Compression Type: '|| cmp_type);
 15  end;
 16  /
Compression Advisor self-check validation successful. select count(*) on both
Uncompressed and EHCC Compressed format = 1000001 rows
Table: MYCUSTOMERS
Compression Ratio: 9.1
Compression Type: "Compress For Archive Low"
```

```
PL/SQL procedure successfully completed.

Elapsed: 00:02:32.60
SQL> declare
  2    b_cmp number;
  3    b_ucmp number;
  4    r_cmp number;
  5    r_ucmp number;
  6    cmp_ratio number(6,2);
  7    cmp_type varchar2(1024);
  8  begin
  9    dbms_compression.get_compression_ratio('USERS', 'SALES',
 10      'MYCUSTOMERS',NULL,DBMS_COMPRESSION.COMP_FOR_ARCHIVE_HIGH,
 11      b_cmp,b_ucmp, r_cmp, r_ucmp, cmp_ratio, cmp_type);
 12    dbms_output.put_line('Table: MYCUSTOMERS');
 13    dbms_output.put_line('Compression Ratio: '||cmp_ratio);
 14    dbms_output.put_line('Compression Type: '|| cmp_type);
 15  end;
 16  /
Compression Advisor self-check validation successful. select count(*) on both
Uncompressed and EHCC Compressed format = 1000001 rows
Table: MYCUSTOMERS
Compression Ratio: 11.1
Compression Type: "Compress For Archive High"

PL/SQL procedure successfully completed.

Elapsed: 00:03:32.62
SQL>
```

5. Exadata Hybrid Columnar Compression achieves its highest levels of compression with data that is direct-path inserted. Execute the following `ALTER SESSION` commands to ensure the use of direct-path inserts later in the practice.

```
SQL> alter session force parallel query;

Session altered.

Elapsed: 00:00:00.00
SQL> alter session force parallel ddl;

Session altered.

Elapsed: 00:00:00.00
SQL> alter session force parallel dml;

Session altered.

Elapsed: 00:00:00.01
```

6. Use the following commands (or execute the SQL script `/home/oracle/labs/lab08-02-06.sql`) to create compressed copies of the `MYCUSTOMERS` table. Notice the relative difference in the time taken to create each table by using the different compression modes.

```
SQL> create table mycust_query_low
  2  compress for query low
  3  nologging parallel 4
  4  as select * from mycustomers;

Table created.

Elapsed: 00:00:13.53
SQL>
SQL> create table mycust_query_high
  2  compress for query high
  3  nologging parallel 4
  4  as select * from mycustomers;

Table created.

Elapsed: 00:00:15.13
SQL>
SQL> create table mycust_archive_low
  2  compress for archive low
  3  nologging parallel 4
  4  as select * from mycustomers;

Table created.

Elapsed: 00:00:22.62
SQL>
SQL> create table mycust_archive_high
  2  compress for archive high
  3  nologging parallel 4
  4  as select * from mycustomers;

Table created.

Elapsed: 00:01:13.98
SQL>
```

7. Use the following query (or execute the SQL script `/home/oracle/labs/lab08-02-07.sql`) to compare the size of the original uncompressed table with the newly created compressed copies. Calculate the compression ratios achieved using the formula:
   `Compression Ratio = Uncompressed Size / Compressed Size`

Compare the actual compression ratios achieved with the ones predicted in step 4.

```
SQL> col segment_name format a30
SQL> select segment_name,sum(bytes)/1024/1024 MB
  2  from user_segments
  3  where segment_name like 'MYCUST%'
  4  group by segment_name;

SEGMENT_NAME                           MB
------------------------------ ----------
MYCUSTOMERS                           208
MYCUST_QUERY_LOW                       51
MYCUST_QUERY_HIGH                      31
MYCUST_ARCHIVE_LOW                     27
MYCUST_ARCHIVE_HIGH                    19

Elapsed: 00:00:02.74
SQL>
```

In the next part of the practice, you will compare direct path insert performance for compressed and uncompressed tables. On each occasion you will perform the same transaction twice. The first time will help to prime the system in order to ensure consistent result. You should take particular note of the timings for the second insert command. This will help you to determine the impact of Exadata Hybrid Columnar Compression on bulk data loading operations.

8. As a baseline, execute the following transactions to load data into the uncompressed `MYCUSTOMERS` table. Note the time taken to perform the second insert.

```
SQL> insert /*+APPEND */ into mycustomers
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:10.37
SQL> commit;

Commit complete.

Elapsed: 00:00:00.07
SQL> insert /*+APPEND */ into mycustomers
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:07.62
SQL> commit;
```

```
Commit complete.

Elapsed: 00:00:00.03
SQL>
```

9. Execute the same insert transactions against the `COMPRESS FOR QUERY LOW` copy of the table. Note the time taken to perform the second insert. You may observe that the time for this insert is better than the uncompressed insert in step 8. In this case, the cost of performing the compression is offset by the lower number of I/O operations that are required. This characteristic is one of the reasons why query compression is well suited to data warehouse environments where large data loads exist.

```
SQL> insert /*+APPEND */ into mycust_query_low
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:04.36
SQL> commit;

Commit complete.

Elapsed: 00:00:00.04
SQL> insert /*+APPEND */ into mycust_query_low
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:03.07
SQL> commit;

Commit complete.

Elapsed: 00:00:00.03
SQL>
```

10. Execute the same insert transactions against the `COMPRESS FOR QUERY HIGH` copy of the table. Note the time taken to perform the second insert and compare it with the previous results.

```
SQL> insert /*+APPEND */ into mycust_query_high
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:03.25
SQL> commit;

Commit complete.

Elapsed: 00:00:00.04
SQL> insert /*+APPEND */ into mycust_query_high
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:03.29
SQL> commit;

Commit complete.

Elapsed: 00:00:00.08
SQL>
```

11. Execute the same insert transactions against the `COMPRESS FOR ARCHIVE LOW` copy of the table. Note the time taken to perform the second insert. You should observe that the load times are steadily increasing as more aggressive compression modes are used.

```
SQL> insert /*+APPEND */ into mycust_archive_low
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:03.74
SQL> commit;

Commit complete.

Elapsed: 00:00:00.04
SQL> insert /*+APPEND */ into mycust_archive_low
  2  select * from seed_data;
```

```
200000 rows created.

Elapsed: 00:00:03.83
SQL> commit;

Commit complete.

Elapsed: 00:00:00.02
SQL>
```

12. Execute the insert transaction against the `COMPRESS FOR ARCHIVE HIGH` copy of the table. Note the time taken to perform the second insert. This time you should observe a more substantial cost for the data compression. This is because `COMPRESS FOR ARCHIVE HIGH` uses a more costly compression algorithm to achieve higher levels of compression. This extra cost is generally acceptable in archiving situations because the data does not change (or changes very little) after it is loaded.

```
SQL> insert /*+APPEND */ into mycust_archive_high
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:05.81
SQL> commit;

Commit complete.

Elapsed: 00:00:00.03
SQL> insert /*+APPEND */ into mycust_archive_high
  2  select * from seed_data;

200000 rows created.

Elapsed: 00:00:05.71
SQL> commit;

Commit complete.

Elapsed: 00:00:00.03
SQL>
```

In the final part of the practice, you will compare query performance for compressed and uncompressed tables.

13. Reconnect to your database as the `sales` user. This clears the session-level statistics, which will be used later to compare query performance.

```
SQL> connect sales/sales
Connected.
SQL>
```

14. Execute the following test query against the uncompressed table. Note the time taken to execute the query.

```
SQL> select avg(cust_credit_limit) from mycustomers;

AVG(CUST_CREDIT_LIMIT)
----------------------
            6176.17987

Elapsed: 00:00:09.07
SQL>
```

15. Examine the I/O statistics for the query you just ran (use the SQL script `/home/oracle/labs/lab08-02-15.sql` if you prefer). This will provide a baseline for later comparison.

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                        291.875
physical write total bytes                                             0
cell physical IO interconnect bytes                           18.3570175
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload         291.859375
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan    18.3413925
cell IO uncompressed bytes                                    291.859375

10 rows selected.

Elapsed: 00:00:00.14
SQL>
```

16. Reconnect to your database as the `sales` user.

```
SQL> connect sales/sales
Connected.
SQL>
```

17. Execute the following test query against the `COMPRESS FOR QUERY LOW` copy of the table. Compare the time taken to execute the query with the query performance observed in step 14.

```
SQL> select avg(cust_credit_limit) from mycust_query_low;

AVG(CUST_CREDIT_LIMIT)
----------------------
            6176.17987

Elapsed: 00:00:07.06
SQL>
```

18. Examine the I/O statistics for the query you just ran (use the SQL script `/home/oracle/labs/lab08-02-15.sql` if you prefer). Compare the statistics with the results observed in step 15. Note the substantial decrease in the I/O required to satisfy the query.

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                     79.6796875
physical write total bytes                                             0
cell physical IO interconnect bytes                             12.45961
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload             72.5625
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan     5.34242249
cell IO uncompressed bytes                                     247.398615

10 rows selected.

Elapsed: 00:00:00.01
SQL>
```

19. Reconnect to your database as the `sales` user.

```
SQL> connect sales/sales
Connected.
SQL>
```

20. Execute the following test query against the `COMPRESS FOR QUERY HIGH` copy of the table. Compare the time taken to execute the query with the query performance observed previously.

```
SQL> select avg(cust_credit_limit) from mycust_query_high;

AVG(CUST_CREDIT_LIMIT)
----------------------
             6176.17987

Elapsed: 00:00:04.23
SQL>
```

21. Examine the I/O statistics for the query you just ran (use the SQL script `/home/oracle/labs/lab08-02-15.sql` if you prefer). Compare the statistics with the results observed previously. Note again the decline in the I/O required to satisfy the query.

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                          52.96875
physical write total bytes                                                0
cell physical IO interconnect bytes                              11.8664856
cell physical IO bytes saved during optimized file creation               0
cell physical IO bytes saved during optimized RMAN file restore           0
cell physical IO bytes eligible for predicate offload              45.984375
cell physical IO bytes saved by storage index                             0
cell physical IO bytes sent directly to DB node to balance CPU            0
cell physical IO interconnect bytes returned by smart scan         4.8821106
cell IO uncompressed bytes                                       248.547052

10 rows selected.

Elapsed: 00:00:00.01
SQL>
```

22. Reconnect to your database as the `sales` user.

```
SQL> connect sales/sales
Connected.
SQL>
```

23. Execute the following test query against the `COMPRESS FOR ARCHIVE LOW` copy of the table. Compare the time taken to execute the query with the query performance observed in previously.

```
SQL> select avg(cust_credit_limit) from mycust_archive_low;

AVG(CUST_CREDIT_LIMIT)
----------------------
            6176.17987

Elapsed: 00:00:03.50
SQL>
```

24. Examine the I/O statistics for the query you just ran (use the SQL script `/home/oracle/labs/lab08-02-15.sql` if you prefer). Compare the statistics with the results observed previously. Note the continued decline in the I/O required to satisfy the query.

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                     49.3203125
physical write total bytes                                             0
cell physical IO interconnect bytes                           11.5118637
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload           42.046875
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan    4.23842621
cell IO uncompressed bytes                                    249.937677

10 rows selected.

Elapsed: 00:00:00.01
SQL>
```

25. Reconnect to your database as the `sales` user.

```
SQL> connect sales/sales
Connected.
SQL>
```

26. Execute the following test query against the `COMPRESS FOR ARCHIVE HIGH` copy of the table. Compare the time taken to execute the query with the query performance observed previously. Note that in all cases, the queries against the compressed tables outperformed the query against the uncompressed table. With compression, you will often observe improved query performance for scanning queries because less I/O is required.

```
SQL> select avg(cust_credit_limit) from mycust_archive_high;

AVG(CUST_CREDIT_LIMIT)
----------------------
            6176.17987

Elapsed: 00:00:03.53
SQL>
```

27. Examine the I/O statistics for the query you just ran (use the SQL script `/home/oracle/labs/lab08-02-15.sql` if you prefer). Compare the statistics with the results previously observed. Note again the decline in the I/O required to satisfy the query.

```
SQL> select a.name, b.value/1024/1024 MB
  2  from v$sysstat a, v$mystat b
  3  where a.statistic# = b.statistic# and
  4  (a.name in ('physical read total bytes',
  5              'physical write total bytes',
  6              'cell IO uncompressed bytes')
  7  or a.name like 'cell phy%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                         34.125
physical write total bytes                                             0
cell physical IO interconnect bytes                           9.19754791
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload           28.265625
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan    3.33817291
cell IO uncompressed bytes                                    247.281427

10 rows selected.

Elapsed: 00:00:00.01
SQL>
```

28. Exit your SQL*Plus session.

Practice 8-3: Testing Index Elimination

Overview

In this practice, you make an index invisible so that you can test the effect of removing the index on your queries without actually dropping the index.

Tasks

1. Establish a terminal connection to your database server.
2. Connect to your database with SQL*Plus. Log in as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Configure your session to display timing statistics and execution plans. Then flush the database buffer cache to ensure consistent results in the following steps.

```
SQL> set timing on
SQL> set autotrace on explain
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.47
SQL>
```

4. Execute the following query. Note that the execution plan uses an index range scan on the CUSTOMERS_PK index. Note also the time taken to execute the query using the index.

```
SQL> select avg(cust_credit_limit) from customers
  2  where cust_id between 200000 and 320000;

AVG(CUST_CREDIT_LIMIT)
----------------------
            7682.74014

Elapsed: 00:00:06.43

Execution Plan
----------------------------------------------------------
Plan hash value: 3995619262

-------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     1 |    10 |  9837
|   1 |  SORT AGGREGATE              |              |     1 |    10 |
|   2 |   TABLE ACCESS BY INDEX ROWID| CUSTOMERS    |  120K|  1171K|  9837
|*  3 |    INDEX RANGE SCAN          | CUSTOMERS_PK |  120K|       |   260
-------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("CUST_ID">=200000 AND "CUST_ID"<=320000)

SQL>
```

5. Reconfigure your session to disable the automatic output of execution plans.

```
SQL> set autotrace off
SQL>
```

6. Make the CUSTOMERS_PK index invisible. An invisible index still exists and is maintained by DML operations, but it is not used by the optimizer for queries.

```
SQL> alter index customers_pk invisible;

Index altered.

Elapsed: 00:00:00.35
SQL>
```

7. The index you have just made invisible is associated with a primary key constraint. Use the following query to check the status of the constraint. Note that even though the index is invisible, the associated constraint is still enabled.

```
SQL> select status from user_constraints
  2  where constraint_name='CUSTOMERS_PK';

STATUS
--------
ENABLED

Elapsed: 00:00:00.64
SQL>
```

8. Reconfigure your session to automatically show execution plans.

```
SQL> set autotrace on explain
SQL>
```

9. Re-execute the query from step 4. Notice that an Exadata Smart Scan is used rather than an index range scan.

```
SQL> select avg(cust_credit_limit) from customers
  2  where cust_id between 200000 and 320000;

AVG(CUST_CREDIT_LIMIT)
----------------------
            7682.74014

Elapsed: 00:00:19.45

Execution Plan
----------------------------------------------------------
Plan hash value: 296924608

-------------------------------------------------------------------------------
| Id  | Operation                 | Name      | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |           |     1 |    10 | 12093   (1)|
|   1 |  SORT AGGREGATE           |           |     1 |    10 |            |
|*  2 |   TABLE ACCESS STORAGE FULL| CUSTOMERS |  120K|  1171K| 12093   (1)|
-------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - storage("CUST_ID"<=320000 AND "CUST_ID">=200000)
       filter("CUST_ID"<=320000 AND "CUST_ID">=200000)

SQL>
```

Compare the time taken to execute the query with and without the index. Note that in some cases, Exadata Smart Scan may deliver better query performance than using an index. However, this may not always be the case. Even in cases where an index delivers better query performance you might choose to remove it if you determine that the un-indexed query performance is acceptable and the index is otherwise unnecessary. Removing unnecessary indexes saves space and improves DML performance by eliminating the maintenance operations associated with the index. If you decide not to remove the index, you can quickly and easily make it visible.

10. Make the index visible again.

```
SQL> alter index customers_pk visible;

Index altered.

Elapsed: 00:00:00.04
SQL>
```

11. Exit your SQL*Plus session.

Practices for Lesson 9: Using Smart Scan

Chapter 9

Practices for Lesson 9

Practices Overview

In these practices, you will exercise Exadata Smart Scan and examine various statistics and wait events to determine what is occurring.

Practice 9-1: Monitoring Exadata Smart Scan

Overview

In this practice, you will examine various Exadata Smart Scan statistics and measures which can be observed within Oracle Database.

Tasks

1. Establish a terminal connection to `qr01db01` as the `oracle` user.
2. Connect to your database with SQL*Plus. Login as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Flush the buffer cache to ensure a consistent starting point for this practice, and reconnect to the database to reset the session statistics. Then, configure your session to display query execution plans and statement timings.

```
SQL> alter system flush buffer_cache;

System altered.

SQL> connect sales/sales
Connected.
SQL> set autotrace on explain
SQL> set timing on
SQL>
```

4. Execute the following query. You can identify whether Smart Scan is possible by examining the query execution plan. Smart Scan is indicated for this query by the `TABLE ACCESS STORAGE FULL` operation. You can also see that the `WHERE` clause predicate (`occupation = 'Farming'`) can be evaluated by Exadata.

```
SQL> select count(*) from cust_info where occupation = 'Farming';

  COUNT(*)
----------
     13845

Elapsed: 00:00:03.58

Execution Plan
----------------------------------------------------------
Plan hash value: 1273666552

-------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |     1 |     7 |  4176   (1)|
|   1 |  SORT AGGREGATE            |           |     1 |     7 |            |
|*  2 |   TABLE ACCESS STORAGE FULL| CUST_INFO |   115K|   788K|  4176   (1)|
-------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - storage("OCCUPATION"='Farming')
       filter("OCCUPATION"='Farming')

SQL>
```

5. Reconfigure your session to disable the automatic output of execution plans.

```
SQL> set autotrace off
SQL>
```

6. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-01-06.sql`) and examine the statistics for the current session. Since the query in step 4 is the only query that has been executed during this session we can safely assume that the statistics relate to that query. The statistics show that approximately 120 MB of I/O was performed to scan the `cust_info` table (`physical read total bytes`), and that almost all of the I/O was eligible for Smart Scan (`cell physical IO bytes eligible for predicate offload`). The statistics further show that Smart Scan returned approximately 200 KB of data back to the database server (`cell physical IO interconnect bytes returned by smart scan`).

```
SQL> SELECT s.name, m.value/1024/1024 MB FROM V$SYSSTAT s, V$MYSTAT m
  2  WHERE s.statistic# = m.statistic# AND
  3  (s.name LIKE 'physical%total bytes' OR s.name LIKE 'cell phys%'
  4  OR s.name LIKE 'cell IO%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                        119.875
physical write total bytes                                             0
cell physical IO interconnect bytes                              .555992126
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload            119.523438
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan       .204429626
cell IO uncompressed bytes                                       119.523438

10 rows selected.

Elapsed: 00:00:00.11
SQL>
```

7. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-01-07.sql`) and examine the cell wait events associated with the current session. Note that the amount of time associated with the `cell smart table scan` wait event accounts for most of the execution time observed in step 4. This is normal for a query using Smart Scan. You may also see some other wait events that relate to other activities in the session, such as executing the statistics queries in this step and the prior step.

```
SQL> SELECT DISTINCT event, total_waits,
  2  time_waited/100 wait_secs, average_wait/100 avg_wait_secs
  3  FROM V$SESSION_EVENT e, V$MYSTAT s
  4  WHERE event LIKE 'cell%' AND e.sid = s.sid;

EVENT                                                            TOTAL_WAITS
---------------------------------------------------------------- -----------
 WAIT_SECS AVG_WAIT_SECS
---------- -------------
cell smart table scan                                                     37
      3.05          .0824

cell single block physical read                                           75
      1.09          .0145


Elapsed: 00:00:00.10
SQL>
```

Examining the statistics and wait events associated with the query executed at step 4 indicates that the query did make efficient use of Exadata Smart Scan just as the query execution plan suggested. In the next part of this practice you will consider a scenario where the query execution plan indicates the use of Smart Scan but the statistics and wait events suggest something more.

8. Leave your current SQL session active and establish a second terminal connection to `qr01db01` as the `oracle` user.
9. In the second terminal, connect to your database with SQL*Plus. Log in as the `sales` user and set the SQL prompt to `UPDATE>` so that you can easily distinguish this session from your other SQL session.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL> set sqlprompt "UPDT> "
UPDT>
```

10. In the second terminal, execute the following command to update a substantial number of customer records. Following the update, flush the buffer cache. This simulates a long running update transaction where the updated blocks (and associated rollback segment blocks) have been aged out of the buffer cache. **Leave the transaction in this terminal window uncommitted for now. Do not proceed to the next step until the update command completes and the buffer cache is flushed.**

```
UPDT> update cust_info set
  2  affinity_card = 0
  3  where occupation = 'Farming';

13845 rows updated.

UPDT> alter system flush buffer_cache;

System altered.

UPDT>
```

11. Switch back to your first SQL session, leaving the second terminal session in the background for now.
12. Back in the first SQL session, reconnect to the database to establish a fresh database session as the `sales` user.

```
SQL> connect sales/sales
Connected.
SQL>
```

13. Configure your newly created session to display query execution plans.

```
SQL> set autotrace on explain
SQL>
```

14. Re-execute the query from step 4. Notice again that the query execution plan indicates the use of Smart Scan. Notice also that the execution time increases substantially compared with step 4.

```
SQL> select count(*) from cust_info where occupation = 'Farming';

  COUNT(*)
----------
     13845

Elapsed: 00:00:06.05

Execution Plan
----------------------------------------------------------
Plan hash value: 1273666552

-------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |     1 |     7 |  4176   (1)|
|   1 |  SORT AGGREGATE            |           |     1 |     7 |            |
|*  2 |   TABLE ACCESS STORAGE FULL| CUST_INFO |   115K|   788K|  4176   (1)|
-------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - storage("OCCUPATION"='Farming')
       filter("OCCUPATION"='Farming')

SQL>
```

15. Reconfigure your session to disable the automatic output of execution plans.

```
SQL> set autotrace off
SQL>
```

16. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-01-06.sql`) and examine the statistics for the current session. Compare the output with the observations at step 6. This time the same amount of I/O is eligible for Smart Scan (`cell physical IO bytes eligible for predicate offload`); however, substantially more data was transported to the database server (`cell physical IO interconnect bytes returned by smart scan`).

```
SQL> SELECT s.name, m.value/1024/1024 MB FROM V$SYSSTAT s, V$MYSTAT m
  2  WHERE s.statistic# = m.statistic# AND
  3  (s.name LIKE 'physical%total bytes' OR s.name LIKE 'cell phys%'
  4  OR s.name LIKE 'cell IO%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                      121.90625
physical write total bytes                                             0
cell physical IO interconnect bytes                            73.2464371
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload          119.523438
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan     70.8636246
cell IO uncompressed bytes                                     119.523438

10 rows selected.

Elapsed: 00:00:00.00
SQL>
```

17. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-01-07.sql`) and examine the cell wait events associated with the current session. Compare the output with the observations at step 7. Notice that this time a significant amount of time is associated with the `cell single block physical read` wait event. This is because the uncommitted update transaction has forced a substantial number of reads to be transferred to the traditional buffer cache read-consistency path and since the required blocks were not in the buffer cache a large number of single block physical reads were required. Notice also the amount of time associated with the different wait events and how they correlate with the overall query execution time. Clearly, the efficiency and performance of Smart Scan were severely compromised by the pending transaction.

```
SQL> SELECT DISTINCT event, total_waits,
  2  time_waited/100 wait_secs, average_wait/100 avg_wait_secs
  3  FROM V$SESSION_EVENT e, V$MYSTAT s
  4  WHERE event LIKE 'cell%' AND e.sid = s.sid;

EVENT                                                            TOTAL_WAITS
---------------------------------------------------------------- -----------
 WAIT_SECS AVG_WAIT_SECS
---------- -------------
cell smart table scan                                                     65
      1.54          .0236

cell single block physical read                                          305
      4.34          .0142


Elapsed: 00:00:00.11
SQL>
```

In the final part of this practice, you will consider another scenario where the query execution plan indicates the use of Smart Scan but the statistics and wait events suggest something different.

18. Switch back to the second SQL session which contains the update transaction.

19. Commit the current transaction and then execute another update command to update a substantial number of customer records. This time, do not flush the buffer cache after the update command. **Like you did in step 10, leave the transaction in this terminal window uncommitted for now. Do not proceed to the next step until the update command completes.**

```
UPDT> commit;

Commit complete.

UPDT> update cust_info set
  2  affinity_card = 1
  3  where occupation = 'Farming';

13845 rows updated.

UPDT>
```

20. Back in the first SQL session, reconnect to the database to establish a fresh database session as the `sales` user.

```
SQL> connect sales/sales
Connected.
SQL>
```

21. Configure your newly created session to display query execution plans.

```
SQL> set autotrace on explain
SQL>
```

22. Re-execute the query from step 4. Notice again that the query execution plan indicates the use of Smart Scan. Notice also the execution time difference compared with the previous runs.

```
SQL> select count(*) from cust_info where occupation = 'Farming';

  COUNT(*)
----------
     13845

Elapsed: 00:00:00.35

Execution Plan
----------------------------------------------------------
Plan hash value: 1273666552

-------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |     1 |     7 |  4176   (1)|
|   1 |  SORT AGGREGATE            |           |     1 |     7 |            |
|*  2 |   TABLE ACCESS STORAGE FULL| CUST_INFO |   115K|   788K|  4176   (1)|
-------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   2 - storage("OCCUPATION"='Farming')
       filter("OCCUPATION"='Farming')

SQL>
```

23. Reconfigure your session to disable the automatic output of execution plans.

```
SQL> set autotrace off
SQL>
```

24. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-01-06.sql`) and examine the statistics for the current session. Now the statistics relate to the query at step 22. Notice that all the statistics are zero (or very near to zero).

```
SQL> SELECT s.name, m.value/1024/1024 MB FROM V$SYSSTAT s, V$MYSTAT m
  2  WHERE s.statistic# = m.statistic# AND
  3  (s.name LIKE 'physical%total bytes' OR s.name LIKE 'cell phys%'
  4  OR s.name LIKE 'cell IO%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                              0
physical write total bytes                                             0
cell physical IO interconnect bytes                                    0
cell physical IO bytes saved during optimized file creation            0
cell physical IO bytes saved during optimized RMAN file restore        0
cell physical IO bytes eligible for predicate offload                  0
cell physical IO bytes saved by storage index                          0
cell physical IO bytes sent directly to DB node to balance CPU         0
cell physical IO interconnect bytes returned by smart scan             0
cell IO uncompressed bytes                                             0

10 rows selected.

Elapsed: 00:00:00.00
SQL>
```

25. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-01-07.sql`) and examine the cell wait events associated with the current session. This time the amount of time waiting for the cells is zero (or very close to zero). This time, the database kernel realized that the query could be satisfied using blocks in the buffer cache. So even though the query plan indicted the use of Smart Scan, the database kernel used the buffer cache at runtime and avoided the need for I/O against the cells.

```
SQL> SELECT DISTINCT event, total_waits,
  2  time_waited/100 wait_secs, average_wait/100 avg_wait_secs
  3  FROM V$SESSION_EVENT e, V$MYSTAT s
  4  WHERE event LIKE 'cell%' AND e.sid = s.sid;

no rows selected

Elapsed: 00:00:00.08
SQL>
```

In this practice you have seen a variety of scenarios where the optimizer indicated the use of Smart Scan. However you have also seen that depending on the situation, the performance of Smart Scan may be impacted by other concurrent transactions, or Smart Scan may be skipped, partially or completely, if the database kernel can make use of information in the buffer cache to avoid I/O operations.

26. Switch back to the second SQL session which contains the update transaction.

27. Commit the transaction.

```
UPDATE> commit;

Commit complete.

UPDATE>
```

28. Exit all your terminal sessions.

Practice 9-2: Monitoring Cell Wait Events for Parallel Query

Overview

In this practice, you consider strategies for monitoring cell wait events when parallel query is used.

Tasks

1. Establish a terminal connection to `qr01db01` as the `oracle` user.
2. Connect to your database with SQL*Plus. Log in as the `sales` user.

```
[oracle@qr01db01 ~]$ sqlplus sales/sales

SQL*Plus: Release 11.2.0.3.0 Production...

SQL>
```

3. Configure your session to display query execution plans and statement timings.

```
SQL> set autotrace on explain
SQL> set timing on
SQL>
```

4. Typically Exadata Smart Scan is used in conjunction with parallel query. Configure your session to force the use of parallel query.

```
SQL> alter session force parallel query parallel 2;

Session altered.

Elapsed: 00:00:00.00
SQL>
```

5. Execute the following query. The query execution plan indicates the use of parallel query and Exadata Smart Scan.

```
SQL> select count(*) from cust_info where occupation = 'Farming';

  COUNT(*)
----------
     13845

Elapsed: 00:01:19.23

Execution Plan
----------------------------------------------------------
Plan hash value: 3555626242

-------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows  | Bytes | Cost
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |     1 |     7 |  2317
|   1 |  SORT AGGREGATE              |           |     1 |     7 |
|   2 |   PX COORDINATOR             |           |       |       |
|   3 |    PX SEND QC (RANDOM)       | :TQ10000  |     1 |     7 |
|   4 |     SORT AGGREGATE           |           |     1 |     7 |
|   5 |      PX BLOCK ITERATOR       |           |   115K|   788K|  2317
|*  6 |       TABLE ACCESS STORAGE FULL| CUST_INFO |   115K|   788K|  2317
-------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   6 - storage("OCCUPATION"='Farming')
       filter("OCCUPATION"='Farming')

SQL>
```

6. Reconfigure your session to disable the automatic output of execution plans.

```
SQL> set autotrace off
SQL>
```

7. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-02-07.sql`) and examine the statistics for the current session. The statistics confirm the use of Smart Scan for the query in step 5.

```
SQL> SELECT s.name, m.value/1024/1024 MB FROM V$SYSSTAT s, V$MYSTAT m
  2  WHERE s.statistic# = m.statistic# AND
  3  (s.name LIKE 'physical%total bytes' OR s.name LIKE 'cell phys%'
  4  OR s.name LIKE 'cell IO%');

NAME                                                                  MB
---------------------------------------------------------------- ----------
physical read total bytes                                       119.53125
physical write total bytes                                              0
cell physical IO interconnect bytes                            .219009399
cell physical IO bytes saved during optimized file creation             0
cell physical IO bytes saved during optimized RMAN file restore         0
cell physical IO bytes eligible for predicate offload           119.523438
cell physical IO bytes saved by storage index                           0
cell physical IO bytes sent directly to DB node to balance CPU          0
cell physical IO interconnect bytes returned by smart scan     .211196899
cell IO uncompressed bytes                                      119.523438

10 rows selected.

Elapsed: 00:00:00.03
SQL>
```

8. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-02-08.sql`) and examine the cell wait events associated with the current session. Notice that there are very few (if any) waits.

```
SQL> SELECT DISTINCT event, total_waits,
  2  time_waited/100 wait_secs, average_wait/100 avg_wait_secs
  3  FROM V$SESSION_EVENT e, V$MYSTAT s
  4  WHERE event LIKE 'cell%' AND e.sid = s.sid;

no rows selected

Elapsed: 00:00:00.06
SQL>
```

Based on the statistics in step 7, you might reasonably expect to see wait events for `cell smart table scan` in step 8. What happened? Because parallel query was used, the query I/O was performed by parallel server processes. The associated wait events are connected to the parallel server sessions, not the current session. Note that this behavior is symptomatic of parallel query and is not Exadata-specific. So when parallel query is used, the wait events must be observed differently. The rest of the practice shows two alternative strategies for observing the wait events.

9. Execute the following query (or execute the SQL script `/home/oracle/labs/lab09-02-09.sql`) to display the cell wait events across the entire system.

```
SQL> select event, total_waits,
  2  time_waited/100 wait_secs, average_wait/100 avg_wait_secs
  3  from v$system_event where event like 'cell%';

EVENT                                                            TOTAL_WAITS
---------------------------------------------------------------- -----------
 WAIT_SECS AVG_WAIT_SECS
---------- -------------
cell smart table scan                                                   9277
    550.44         .0593

cell single block physical read                                         6099
    216.74         .0355

cell multiblock physical read                                           1826
     64.57         .0354

cell list of blocks physical read                                         37
     12.37         .3342


Elapsed: 00:00:00.24
SQL>
```

10. Re-execute the parallel query.

```
SQL> select count(*) from cust_info where occupation =
'Farming';



  COUNT(*)
----------
     13845

Elapsed: 00:00:02.91
SQL>
```

11. Re-execute the following query (or execute the SQL script `/home/oracle/labs/lab09-02-09.sql`) to again display the cell wait events across the entire system. Compare the output with the output from step 9. The differences are the cell wait events associated with the query at step 10.

```
SQL> select event, total_waits,
  2  time_waited/100 wait_secs, average_wait/100 avg_wait_secs
  3  from v$system_event where event like 'cell%';

EVENT                                                            TOTAL_WAITS
---------------------------------------------------------------- -----------
 WAIT_SECS AVG_WAIT_SECS
---------- -------------
cell smart table scan                                                   9404
    556.14          .0591

cell single block physical read                                        6099
    216.74          .0355

cell multiblock physical read                                          1826
     64.57          .0354

cell list of blocks physical read                                        37
     12.37          .3342


Elapsed: 00:00:00.00
SQL>
```

Using system-level wait event statistics is a simple way to monitor parallel query wait events as long as you are the only user of the system and you do not wish to monitor concurrent operations. Often this is not the case. The final part of this practice shows another method which can be used to isolate the wait events associated with a specific parallel query operation regardless of concurrency.

12. Execute the following query to determine the default trace file for the current session. Take note of the directory path since that location will also be the default location for other trace files.

```
SQL> select value from v$diag_info
  2  where name = 'Default Trace File';

VALUE
-----------------------------------------------------------------
/u01/app/oracle/diag/rdbms/dbm/dbm1/trace/dbm1_ora_5387.trc

Elapsed: 00:00:00.40
SQL>
```

13. Use the `dbms_session.set_identifier` procedure to set a client identifier (`PQ1`) for the current session. The client identifier will help to locate trace information associated with the current session and any parallel query sessions that perform work in behalf of the current session.

```
SQL> exec dbms_session.set_identifier(client_id=>'PQ1')

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.17
SQL>
```

14. Use the `dbms_session.client_id_trace_enable` procedure to start recording trace information for the `PQ1` client identifier. Notice that `waits=>true` is specified to ensure that wait information is recorded in the trace.

```
SQL> exec dbms_monitor.client_id_trace_enable(client_id=>'PQ1', waits=>true,
binds=>false)

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.20
SQL>
```

15. Re-execute the parallel query.

```
SQL> select count(*) from cust_info where occupation =
'Farming';


  COUNT(*)
----------
     13845


Elapsed: 00:00:03.20
SQL>
```

16. Stop the trace gathering started in step 14 and exit SQL*Plus.

```
SQL> exec dbms_monitor.client_id_trace_disable(client_id=>'PQ1')

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.00
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.3.0...
[oracle@qr01db01 ~]$
```

17. Change directories to the location of the trace file observed in step 12.

```
[oracle@qr01db01 ~]$ cd /u01/app/oracle/diag/rdbms/dbm/dbm1/trace
[oracle@qr01db01 trace]$
```

18. Search for trace files containing `CLIENT ID:(PQ1)`. The resulting output will include the trace file for the client SQL session that was listed in the query output from step 12. It will also include trace files associated with any parallel query servers that performed work for the same client.

```
[oracle@qr01db01 trace]$ grep "CLIENT ID:(PQ1)" *
dbm1_ora_5387.trc:*** CLIENT ID:(PQ1) 2013-07-18 02:40:33.367
dbm1_p000_28409.trc:*** CLIENT ID:(PQ1) 2013-07-18 02:40:45.726
dbm1_p001_28413.trc:*** CLIENT ID:(PQ1) 2013-07-18 02:40:45.727
[oracle@qr01db01 trace]$
```

19. Examine the trace files listed in step 18. The parallel query server trace files will display the `cell smart table scan` events associated with the parallel query.

```
[oracle@qr01db01 trace]$ grep cell dbm1_p001_28413.trc
...
WAIT #140046319890584: nam='cell smart table scan' ela= 60022
cellhash#=1662637845 p2=0 p3=0 obj#=77120 tim=1374129647458319
WAIT #140046319890584: nam='cell smart table scan' ela= 330
cellhash#=3713325327 p2=0 p3=0 obj#=77120 tim=1374129647459253
WAIT #140046319890584: nam='cell smart table scan' ela= 171120
cellhash#=3713325327 p2=0 p3=0 obj#=77120 tim=1374129647630414
WAIT #140046319890584: nam='cell smart table scan' ela= 1004
cellhash#=1662637845 p2=0 p3=0 obj#=77120 tim=1374129647632049
WAIT #140046319890584: nam='cell smart table scan' ela= 3 cellhash#=2749642338
p2=0 p3=0 obj#=77120 tim=1374129647632089
WAIT #140046319890584: nam='cell smart table scan' ela= 131068
cellhash#=2749642338 p2=0 p3=0 obj#=77120 tim=1374129647763208
WAIT #140046319890584: nam='cell smart table scan' ela= 15
cellhash#=1662637845 p2=0 p3=0 obj#=77120 tim=1374129647763363
WAIT #140046319890584: nam='cell smart table scan' ela= 98818
cellhash#=1662637845 p2=0 p3=0 obj#=77120 tim=1374129647862204
WAIT #140046319890584: nam='cell smart table scan' ela= 288
cellhash#=2749642338 p2=0 p3=0 obj#=77120 tim=1374129647863147
WAIT #140046319890584: nam='cell smart table scan' ela= 246864
cellhash#=2749642338 p2=0 p3=0 obj#=77120 tim=1374129648110052
WAIT #140046319890584: nam='cell smart table scan' ela= 349
cellhash#=3713325327 p2=0 p3=0 obj#=77120 tim=1374129648111076
WAIT #140046319890584: nam='cell smart table scan' ela= 164544
cellhash#=3713325327 p2=0 p3=0 obj#=77120 tim=1374129648275729
WAIT #140046319890584: nam='cell smart table scan' ela= 395
cellhash#=1662637845 p2=0 p3=0 obj#=77120 tim=1374129648276811
WAIT #140046319890584: nam='cell smart table scan' ela= 3 cellhash#=2749642338
p2=0 p3=0 obj#=77120 tim=1374129648276841
WAIT #140046319890584: nam='cell smart table scan' ela= 301395
cellhash#=2749642338 p2=0 p3=0 obj#=77120 tim=1374129648578285
WAIT #140046319890584: nam='cell smart table scan' ela= 29
cellhash#=2749642338 p2=0 p3=0 obj#=77120 tim=1374129648578539
WAIT #140046319890584: nam='cell smart table scan' ela= 25091
cellhash#=2749642338 p2=0 p3=0 obj#=77120 tim=1374129648603655
WAIT #140046319890584: nam='cell smart table scan' ela= 332
cellhash#=3713325327 p2=0 p3=0 obj#=77120 tim=1374129648604616
WAIT #140046319890584: nam='cell smart table scan' ela= 86345
cellhash#=3713325327 p2=0 p3=0 obj#=77120 tim=1374129648691003
WAIT #0: nam='cell smart table scan' ela= 1030 cellhash#=2749642338 p2=0 p3=0
obj#=77120 tim=1374129648770459
WAIT #0: nam='cell smart table scan' ela= 459 cellhash#=3713325327 p2=0 p3=0
obj#=77120 tim=1374129648770946
WAIT #0: nam='cell smart table scan' ela= 550 cellhash#=1662637845 p2=0 p3=0
obj#=77120 tim=1374129648771519
[oracle@qr01db01 trace]$
```

20. Exit your terminal session.

Practices for Lesson 10: Consolidation Options and Recommendations

Chapter 10

Practices for Lesson 10

Practices Overview

There is no practice for Lesson 10.