

**Hardware and Software**  
**Engineered to Work Together**



# Oracle Database 12c: New Features for Administrators

Activity Guide – Volume II

D77758GC20

Edition 2.0 | November 2014 | D87751

Learn more from Oracle University at [oracle.com/education/](http://oracle.com/education/)

**Copyright © 2014, Oracle and/or its affiliates. All rights reserved.**

#### **Disclaimer**

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

#### **Restricted Rights Notice**

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

##### **U.S. GOVERNMENT RIGHTS**

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

#### **Trademark Notice**

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

#### **Authors**

Dominique Jeunot, Jean-François Verrier

#### **Technical Contributors and Reviewers**

Andy Rivenes, James Spiller, Donna Keesling, Maria Billings, Lachlan Williams, Peter Fusek, Mark Fuller, Gregg Christman, Dimpi Sarmah, Kevin Jernigan, Branislav Valny, Frank Fu, Joel Goodman, Gerlinde Frenzen, Harald Van Breederode, Hermann Baer, Jim Stenoish, Mark Drake, Beda Hammerschmidt, Prabhaker Gongloor, Patrick Wheeler, Maria Colgan, Jesse Kamp, Paul Needham, Pat Huey, Roy F Swonger, Ron Soltani, Sue Lee, Sharath Bhujani

#### **Editors**

Anwasha Ray, Malavika Jinka, Smita Kommini

#### **Graphic Designers**

Seema Bopaiah, Maheshwari Krishnamurthy

#### **Publishers**

Jobi Varghese, Pavithran Adka, Joseph Fernandez

**This book was published using: Oracle Tutor**

# Table of Contents

<b>Practices for Lesson 1: Introduction .....</b>	<b>1-1</b>
Practices for Lesson 1 .....	1-2
<b>Practices for Lesson 2: Enterprise Manager Cloud Control and Other Tools .....</b>	<b>2-1</b>
Practices for Lesson 2: Overview .....	2-2
Practice 2-1: Accessing Enterprise Manager .....	2-3
Practice 2-2: Adding a Database Instance as a New Target Monitored by EM Cloud Control .....	2-6
Practice 2-3: Creating New Named Credentials .....	2-9
Practice 2-4: Testing the Named Credential .....	2-10
<b>Practices for Lesson 3: Basics of Multitenant Container Database and Pluggable Databases .....</b>	<b>3-1</b>
Practices for Lesson 3: Overview .....	3-2
Practice 3-1: Exploring CDB Architecture and Structures .....	3-3
<b>Practices for Lesson 4: Creating a Multitenant Container Database and Pluggable Databases .....</b>	<b>4-1</b>
Practices for Lesson 4: Overview .....	4-2
Practice 4-1: Creating a New CDB .....	4-3
Practice 4-2: Exploring CDB and PDB Structures .....	4-7
Practice 4-3: Creating a PDB from Seed .....	4-18
Practice 4-4: Cloning PDB Within the Same CDB .....	4-26
Practice 4-5: Cloning a Non-CDB into a CDB .....	4-38
Practice 4-6: Merging All PDBs of CDBs into a Single CDB .....	4-45
<b>Practices for Lesson 5: Managing a Multitenant Container Database and Pluggable Databases .....</b>	<b>5-1</b>
Practices for Lesson 5: Overview .....	5-2
Practice 5-1: Shutdown and Startup of the CDB .....	5-3
Practice 5-2: Closing and Opening a PDB .....	5-8
Practice 5-3: Changing PDBs' Open Mode .....	5-14
Practice 5-4: Instance Parameter Changes: Impact on PDBs (Optional) .....	5-16
<b>Practices for Lesson 6: Managing Tablespaces and Users in a CDB and PDBs .....</b>	<b>6-1</b>
Practices for Lesson 6: Overview .....	6-2
Practice 6-1: Managing Tablespaces .....	6-3
Practice 6-2: Managing Common and Local Users .....	6-10
Practice 6-3: Managing Local and Common Roles .....	6-19
Practice 6-4: Managing Local and Common Privileges .....	6-27
<b>Practices for Lesson 7: Backup, Recovery, Flashback CDB and PDBs .....</b>	<b>7-1</b>
Practices for Lesson 7: Overview .....	7-2
Practice 7-1: Cold CDB Backup .....	7-3
Practice 7-2: RMAN Whole CDB Backup .....	7-5
Practice 7-3: RMAN CDB / PDB Backup .....	7-10
Practice 7-4: RMAN Recovery from PDB Data File Loss .....	7-12
Practice 7-5: SQL PDB Hot Backup (Optional) .....	7-17
Practice 7-6: SQL Control File Backup (Optional) .....	7-19
Practice 7-7: RMAN Recovery from Control File Loss (Optional) .....	7-21
Practice 7-8: RMAN Recovery from Redo Log File Member Loss (Optional) .....	7-26
Practice 7-9: RMAN Recovery from SYSTEM Root Data File Loss (Optional) .....	7-29
Practice 7-10: RMAN Recovery from Non-Essential Root Data File Loss (Optional) .....	7-34
Practice 7-11: PITR on PDB Tablespaces (Optional) .....	7-39
Practice 7-12: Flashback from Common User Drop (Optional) .....	7-52

<b>Practices for Lesson 8: Heat Map, Automatic Data Optimization and Online Datafile Move.....</b>	<b>8-1</b>
Practices for Lesson 8: Overview.....	8-2
Practice 8-1: Enabling Heat Map .....	8-3
Practice 8-2: Automatic Data Optimization – Creating a TIER Policy.....	8-5
Practice 8-3: Automatic Data Optimization – Creating a COMPRESS Policy .....	8-13
Practice 8-4: Cleanup ADO Policies and Heat Map Statistics.....	8-21
Practice 8-5: Moving Data File Online.....	8-23
<b>Practices for Lesson 9: In-Database Archiving and Temporal Validity .....</b>	<b>9-1</b>
Practices for Lesson 9: Overview.....	9-2
Practice 9-1: In-Database Archiving – Row-Archival .....	9-3
Practice 9-2: Temporal Validity .....	9-8
Practice 9-3: Collecting User Context in FDA History Tables (Optional).....	9-17
Practice 9-4: Cleaning Up FDA.....	9-23
<b>Practices for Lesson 10: Auditing .....</b>	<b>10-1</b>
Practices for Lesson 10: Overview.....	10-2
Practice 10-1: Enabling Unified Auditing .....	10-3
Practice 10-2: Auditing RMAN Backup and Recovery Operations .....	10-9
<i>Practice 10-3: Auditing SYS and End-Users (Optional).....</i>	<i>10-13</i>
<i>Practice 10-4: Excluding DBSNMP Login Events (Optional).....</i>	<i>10-18</i>
<b>Practices for Lesson 11: Privileges .....</b>	<b>11-1</b>
Practices for Lesson 11: Overview.....	11-2
Practice 11-1: Managing Password File with SYSBACKUP Entry .....	11-3
Practice 11-2: Capturing Privileges .....	11-8
<i>Practice 11-3: Capturing Privileges Used Through Roles (Optional).....</i>	<i>11-17</i>
<i>Practice 11-4: Capturing Privileges Used In Contexts (Optional) .....</i>	<i>11-21</i>
<i>Practice 11-5: Using INHERIT PRIVILEGES Privilege (Optional) .....</i>	<i>11-24</i>
<i>Practice 11-6: Using BEQUEATH Views (Optional) .....</i>	<i>11-28</i>
<b>Practices for Lesson 12: Oracle Data Redaction.....</b>	<b>12-1</b>
Practices for Lesson 12: Overview.....	12-2
Practice 12-1: Redacting Protected Column Values with FULL Redaction .....	12-3
Practice 12-2: Redacting Protected Column Values with PARTIAL Redaction (Optional).....	12-7
Practice 12-3: Cleaning Up Redaction Policies.....	12-10
Practice 12-4: Changing the Default Value for FULL Redaction (Optional) .....	12-12
<b>Practices for Lesson 13: Recovery Manager - New Features and Temporal History Enhancements .....</b>	<b>13-1</b>
Practices for Lesson 13: Overview.....	13-2
Practice 13-1: Using SYSBACKUP in RMAN .....	13-3
Practice 13-2: Recovering a Table by Using Table Recovery.....	13-8
<b>Practices for Lesson 14: Real-Time Database Operation Monitoring .....</b>	<b>14-1</b>
Practices for Lesson 14: Overview.....	14-2
Practice 14-1: Starting Enterprise Manager Database Express .....	14-3
Practice 14-2: Identifying and Starting Database Operations .....	14-7
<i>Practice 14-3: Identifying and Starting Database Load Operations (Optional) .....</i>	<i>14-14</i>
Practice 14-4: Cleaning Up.....	14-16
<b>Practices for Lesson 15: Emergency Monitoring and Compare Period ADDM.....</b>	<b>15-1</b>
Practices for Lesson 15: Overview.....	15-2
Practice 15-1: Using Emergency Monitoring.....	15-3
Practice 15-2: Cleaning Up.....	15-7
<i>Practice 15-3: Using Compare Period ADDM (Optional) .....</i>	<i>15-8</i>

<b>Practices for Lesson 16: ADR and Network Enhancements .....</b>	<b>16-1</b>
Practices for Lesson 16: Overview.....	16-2
Practice 16-1: Viewing ADR DDL Log File.....	16-3
<b>Practices for Lesson 17: In-Memory Column Store .....</b>	<b>17-1</b>
Practices for Lesson 17: Overview.....	17-2
Practice 17-1: Configuring IM Column Store.....	17-3
Practice 17-2: Configuring In-Memory Tables.....	17-6
Practice 17-3: Querying In-Memory Tables .....	17-12
Practice 17-4: Exporting and Importing In-Memory Tables (Optional) .....	17-20
Lesson 17-5: Using In-Memory Column Store (Demonstration).....	17-25
<b>Practices for Lesson 18: In-Memory Caching .....</b>	<b>18-1</b>
Practices for Lesson 18: Overview.....	18-2
Practice 18-1: Using Automatic Big Table Caching.....	18-3
Practice 18-2: Setting and Monitoring Force Full Database Caching .....	18-12
<b>Practices for Lesson 19: SQL Tuning Enhancements .....</b>	<b>19-1</b>
Practices for Lesson 19: Overview.....	19-2
Practice 19-1: Using Dynamic Plans .....	19-3
Practice 19-2: Using Re-Optimization .....	19-9
<b>Practices for Lesson 20: Resource Manager and Other Performance Enhancements .....</b>	<b>20-1</b>
Practices for Lesson 20: Overview.....	20-2
Practice 20-1: Using CDB Resource Manager Plans and Directives.....	20-3
Practice 20-2: Using Multi-Process Multi-Threaded Architecture .....	20-14
<b>Practices for Lesson 21: Tables, Indexes and Online Operations .....</b>	<b>21-1</b>
Practices for Lesson 21: Overview.....	21-2
Practice 21-1: Using Invisible Table Columns.....	21-3
Practice 21-2: Using Advanced Row Compression.....	21-7
<b>Practices for Lesson 22: Oracle Data Pump, SQL*Loader, and External Tables.....</b>	<b>22-1</b>
Practices for Lesson 22: Overview.....	22-2
Practice 22-1: Exporting/Importing Databases in FULL TRANSPORTABLE Mode.....	22-3
<i>Practice 22-2: Loading Data Using SQL*Loader Express Mode (Optional) .....</i>	<i>22-15</i>
<b>Practices for Lesson 23: Partitioning Enhancements.....</b>	<b>23-1</b>
Practices for Lesson 23: Overview.....	23-2
Practice 23-1: Local and Global Partial Indexing on Partitioned Tables .....	23-3
<b>Practices for Lesson 24: SQL Enhancements and Migration Assistant for Unicode .....</b>	<b>24-1</b>
Practices for Lesson 24: Overview.....	24-2
Practice 24-1: Using 32K VARCHAR2 Data Type.....	24-3
Practice 24-2: Querying a Table Using a SQL Row-Limiting Clause (Optional).....	24-7
<b>Practices for Appendix C: Schema and Data Changes Management .....</b>	<b>25-1</b>
Practices for Appendix C: Overview.....	25-2
Appendix C-1: Using Schema Change Plans .....	25-3



## **Practices for Lesson 15: Emergency Monitoring and Compare Period ADDM**

### **Chapter 15**

## Practices for Lesson 15: Overview

---

### Practices Overview

In these practices, you will use Emergency Monitoring to troubleshoot a hanging situation discovered in `orcl` database instance.

Then you can optionally run the Compare Period ADDM demonstration to know how to use this new feature.



## Practice 15-1: Using Emergency Monitoring

### Assumption

You are managing the target `orcl` database and are already connected to Enterprise Manager Cloud Control in the target `orcl` database. Make sure you restart Enterprise Manager Cloud Control (you stopped it in a previous practice to enable Unified Auditing).

- a. Restart the Enterprise Manager Repository Database `em12rep` unless it was already restarted in the previous practice.

```
$ . oraenv
ORACLE_SID = [orcl] ? em12rep
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to an idle instance.
SQL> startup
ORACLE instance started.

Total System Global Area      503316480 bytes
Fixed Size                     2916056 bytes
Variable Size                  272630056 bytes
Database Buffers               222298112 bytes
Redo Buffers                    5472256 bytes
Database mounted.
Database opened.
SQL> EXIT
$
```

- b. Restart the OMS unless it was already restarted in the previous practice.

```
$ export OMS_HOME=/u01/app/oracle/product/middleware/oms
$ $OMS_HOME/bin/emctl start oms
Oracle Enterprise Manager Cloud Control 12c Release 4
Copyright (c) 1996, 2014 Oracle Corporation. All rights
reserved.
Starting Oracle Management Server...
Starting WebTier...
WebTier Successfully Started
Oracle Management Server Successfully Started
Oracle Management Server is Up
Starting BI Publisher Server ...
BI Publisher Server Already Started
BI Publisher Server is Up
$
```

- c. Use <https://localhost:7802/em> to get the Enterprise Manager Cloud Control Console appear, enter **sysman** in the User Name field and **oracle123** in the Password field. Then click Login.  
The status of the **orcl** database agent might be in unreachable state because the oms was stopped in a previous practice. However this has no incidence on other practices.

## Overview

In this practice you will troubleshoot a hanging situation after users told you they could not connect to the **orcl** instance anymore.

You can use Emergency Monitoring only from Enterprise Manager Cloud Control.

## Tasks

1. Make sure you are already connected to Enterprise Manager Cloud Control in the target **orcl** database with **SYSDBA** credentials.
  - a. Connect to Enterprise Manager Cloud Control as **sysman** with **Oracle123** password.
  - b. After being connected, click “Targets” and then “Databases”.
  - c. From the right pane, click the **orcl** database instance. You are still not connected to **orcl**.
  - d. Therefore, go to any of the menus and try to execute any DBA operation. For example click the Security menu and then “Users” to create a new user or click the Administration menu, then Storage, then Tablespaces to create a tablespace.
  - e. Because you created a named credential in Practice 2-3, the named credential is proposed to make a connection as **SYSDBA** to the instance. You can use it or create a new credential to log in. Log in.
2. In a terminal window, make sure you are at the `~/labs/Emergency` directory and your environment points to the **orcl** instance.

```
$ cd ~/labs/Emergency
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

3. Execute the `Emergency_setup.sh` shell script.

```
$ ./Emergency_setup.sh
DROP TRIGGER trig_after_logon
*
ERROR at line 1:
ORA-04080: trigger 'TRIG_AFTER_LOGON' does not exist
```

Leave the `Emergency_setup.sh` shell script pending.

4. Meanwhile you create a new user. From another terminal window, connect to the `orcl` database as `SYSTEM` with `oracle_4U` password.

```
$ sqlplus system

SQL*Plus: Release 12.1.0.2.0 Production on Wed May 21 05:26:26
2014

Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter password: *****
```

But the session is pending.

5. From another terminal window, connect to the `orcl` database AS `SYSDBA` with `oracle_4U` password.

```
$ sqlplus / AS SYSDBA

SQL*Plus: Release 12.1.0.2.0 Production on Wed May 21 05:27:05
2014

Copyright (c) 1982, 2014, Oracle. All rights reserved.
```

But the session is pending.

6. You will use Emergency Monitoring to quickly solve the hanging situation. Click Performance from the menu, then Emergency Monitoring from the list of options.

**Please Provide Credentials**

**DB Host Credentials**

Please provide Database Host credentials. These credentials are used for performing kill operation on offending sessions in emergency situations.

**Credential** ☐ Preferred ☐ Named ☒ New

\***UserName**

\***Password**

☐ Save As

In DB Host Credentials, choose New and enter `oracle` for both the Username and the Password. Then click Submit.

7. The Hanging Analysis page is displayed. It shows the blockers and blocked sessions. To release the blocked sessions, kill the blocking session. Click the Kill Session button. And click YES to approve the operation. A message appears to inform that the session has been killed. Click OK.

**Emergency Performance Page**

**Hang Analysis**

**Final Blockers** **Blocked Sessions**

**Top Final Blockers by cumulative blocking time**

**Kill Session**

Sessio...	Instance ID	Number...	Cumul b...	User Na...	Program	Service	Module	Action
48	1	5	941	SYS	sqlplus@EDRSI	SYS\$USERS	sqlplus@EDRSI	- No Value -

**Details of Session 48**

Session Serial #	: 5913	SQL ID	: No Value	P1	: 1650815232
P2	: 1	P3	: 0	P1 Text	: driver id
P2 Text	: #bytes	P3 Text	: No Value	OS Process Id	: 26206

**Waiters on Session 48**

Session ID	Instance ID	Waiting ...	User Name	Program	Module	Action	Wait Event
280	1	171	SYS	oracle@EDRSR4	- No Value -	- No Value -	enq: TM - conter
270	1	201	SYSTEM	sqlplus@EDRSR4	sqlplus@EDRSR4	- No Value -	enq: TM - conter
273	1	198	DBSNMP	JDBC Thin Client	JDBC Thin Client	- No Value -	enq: TM - conter
249	1	190	SYS	sqlplus@EDRSR4	sqlplus@EDRSR4	- No Value -	enq: TM - conter
49	1	181	SYS	oracle@EDRSR4	- No Value -	- No Value -	enq: TM - conter

8. Check that the sessions that were hanging are now released.

```
$ sqlplus system
```

```
Enter password: *****
```

```
SQL*Plus: Release 12.1.0.2.0 Production on Wed May 21 05:26:26
2014
```

```
Copyright (c) 1982, 2014, Oracle. All rights reserved.
```

```
SQL> EXIT
```

```
$
```

## Practice 15-2: Cleaning Up

---

### Overview

In this practice you clean up the environment of the `orcl` database.

### Tasks

1. From the session where you launched the `Emergency_setup.sh` shell script, clean up the `orcl` database. If the `Emergency_setup.sh` shell script does not end up (because of the 1800 seconds sleep), interrupt it with `CTRL C`.

```
$ ./Emergency_setup.sh
DROP TRIGGER trig_after_logon
*
ERROR at line 1:
ORA-04080: trigger 'TRIG_AFTER_LOGON' does not exist

CTRL C
$
```

2. From one of the released SQL\*Plus session, drop the trigger.

```
SQL> ALTER SYSTEM SET "_system_trig_enabled"=FALSE;

System altered.

SQL> DROP TRIGGER trig_after_logon;

Trigger dropped.

SQL> EXIT
$
```

## Practice 15-3: Using Compare Period ADDM (*Optional*)

---

### Overview

In this demonstration, you will see how to use the Compare Period ADDM with Enterprise Manager Cloud Control. You discover that during the current period of time, the performance is decreasing. You need to understand why the performance changed, and the root causes of this change, so as to perform appropriate actions to solve the issue.

In this practice, you use a browser to execute the Compare Period ADDM demonstration.

### Tasks

Launch a browser and enter in the file:

`////home/oracle/demos/Compare_Period_ADDM/Compare_Period_ADDM.html`

## **Practices for Lesson 16: ADR and Network Enhancements**

### **Chapter 16**

## Practices for Lesson 16: Overview

---

### Practices Overview

In this practice, you will familiarize yourself with viewing an ADR DDL log file and content.



## Practice 16-1: Viewing ADR DDL Log File

---

### Overview

In this practice, you will find and view the ADR DDL log file.

### Tasks

1. Set the `ENABLE_DDL_LOGGING` instance parameter to `TRUE` to activate the DDL logging.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> ALTER SYSTEM SET enable_ddl_logging=TRUE SCOPE=both;

System altered.

SQL> EXIT
$
```

2. The administrator is performing various administration tasks requiring DDL statements. Execute the `$HOME/labs/ADR/ddl.sql` script.

```
$ sqlplus system

Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter password: *****

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> @$HOME/labs/ADR/ddl.sql

Table created.
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```
Table created.

User created.

User dropped.

Table dropped.

Table dropped.

$
```

3. Check the existence of the DDL log file in the ADR directory.

```
$ cd /u01/app/oracle/diag/rdbms/orcl/orcl/log/ddl
$ ls -ltr
total 4
-rw-rw---- 1 oracle oinstall 1516 May 21 05:33 log.xml
$
```

4. Use ADRCI utility to view the content of the DLL log file.
  - a. Launch the `adrci` utility and execute the `SHOW LOG` command.

```
$ adrci
ADRCI: Release 12.1.0.2.0 - Production on Wed May 21 05:33:56
2014

Copyright (c) 1982, 2014, Oracle and/or its affiliates. All
rights reserved.

ADR base = "/u01/app/oracle"
adrci> SHOW LOG
```

- b. A vi editor page is displayed.

[illegible]

- c. To quit the editor, use the `:q` vi command.

```
:q
ADR Home = /u01/app/oracle/diag/rdbms/orcl/orcl:
*****
*****
Output the results to file: /tmp/utsout_10361_13987_1.ado
adrci> EXIT
$
```

5. View the content of the DDL log file with a UNIX command.

```
$ more /u01/app/oracle/diag/rdbms/orcl/orcl/log/ddl/log.xml
<msg time='2014-05-21T05:33:06.239+00:00' org_id='oracle'
comp_id='rdbms'
  msg_id='opiexe:4222:2946163730' type='UNKNOWN' group='diag_adl'
  level='16' host_id='EDRSR41P1' host_addr='139.185.35.141'
  version='1'>
  <txt>CREATE TABLE scott.test1 (c NUMBER)
  </txt>
</msg>
<msg time='2014-05-21T05:33:11.925+00:00' org_id='oracle'
comp_id='rdbms'
  msg_id='opiexe:4222:2946163730' type='UNKNOWN' group='diag_adl'
  level='16' host_id='EDRSR41P1' host_addr='139.185.35.141'>
  <txt>CREATE TABLE scott.test2 (c VARCHAR2(10))
  </txt>
</msg>
<msg time='2014-05-21T05:33:20.152+00:00' org_id='oracle'
comp_id='rdbms'
  msg_id='opiexe:4222:2946163730' type='UNKNOWN' group='diag_adl'
  level='16' host_id='EDRSR41P1' host_addr='139.185.35.141'>
  <txt>DROP USER new_u1 CASCADE
  </txt>
</msg>
<msg time='2014-05-21T05:33:23.755+00:00' org_id='oracle'
comp_id='rdbms'
  msg_id='opiexe:4222:2946163730' type='UNKNOWN' group='diag_adl'
  level='16' host_id='EDRSR41P1' host_addr='139.185.35.141'>
  <txt>DROP TABLE scott.test2
  </txt>
</msg>
<msg time='2014-05-21T05:33:27.236+00:00' org_id='oracle'
comp_id='rdbms'
  msg_id='opiexe:4222:2946163730' type='UNKNOWN' group='diag_adl'
  level='16' host_id='EDRSR41P1' host_addr='139.185.35.141'>
  <txt>DROP TABLE scott.test1
  </txt>
</msg>
$
```

## **Practices for Lesson 17: In-Memory Column Store**

### **Chapter 17**

## Practices for Lesson 17: Overview

---

### Practices Overview

In the practice for this lesson, you will configure the database instance to use the IM column store and set in-memory attributes on tables so that querying on the in-memory tables show a performance difference.

The performance of queries using the IM column store and those using the buffer cache might not be as spectacular as expected due to the limitations in memory and disk space of the servers.

A demonstration is available to show you how performance of queries on in-memory segments is improved compared to queries using the buffer cache.

## Practice 17-1: Configuring IM Column Store

---

### Overview

In this practice, you set up the appropriate initialization parameter to configure IM column store in the `orcl` database instance.

### Tasks

1. Make sure you are in the `~/labs/Memory` directory and your environment points to the `orcl` instance.

```
$ cd ~/labs/Memory
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

2. Run the `Tables_setup.sh` script to create the `SSB` user identified by the `oracle_4U` password, the `LINEORDER`, `SUPPLIER`, and `DATE_DIM` tables and to load the tables. The `LINEORDER` table contains 700000 rows. While the loading is taking place, continue with the next steps in another session.

```
$ ./Tables_setup.sh
...
Commit point reached - logical record count 699974
Commit point reached - logical record count 700000

Table LINEORDER:
  700000 Rows successfully loaded.
...
Commit point reached - logical record count 1984
Commit point reached - logical record count 2000

Table SUPPLIER:
  2000 Rows successfully loaded.
...
Commit point reached - logical record count 2378
Commit point reached - logical record count 2436
Commit point reached - logical record count 2494
Commit point reached - logical record count 2552
Commit point reached - logical record count 2556

Table DATE_DIM:
  2556 Rows successfully loaded.
```

```

Check the log file:
    control_date.log
for more information about the load.
$

```

3. Set the SGA size to 2 Gb and the IM column store to 1500 Mb.

```

$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> ALTER SYSTEM SET sga_target=2000M SCOPE=spfile;

System altered.

SQL> ALTER SYSTEM SET inmemory_size=1500M SCOPE=spfile;

System altered.

SQL>

```

4. When the Tables\_setup.sh script has completed, restart the instance.

```

SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area      2097152000 bytes
Fixed Size                     2917384 bytes
Variable Size                  402656248 bytes
Database Buffers               100663296 bytes
Redo Buffers                   13856768 bytes
In-Memory Area                1577058304 bytes
Database mounted.
Database opened.
SQL>

```



5. Verify the total size of the SGA and the size of the IM column store.

```
SQL> SHOW PARAMETER sga_target
```

NAME	TYPE	VALUE
sga_target	big integer	2000M

```
SQL> SHOW PARAMETER inmemory_size
```

NAME	TYPE	VALUE
inmemory_size	big integer	1504M

```
SQL> EXIT
```

```
$
```

## Practice 17-2: Configuring In-Memory Tables

### Overview

In this practice, you will assign in-memory attributes on the `LINEORDER` and `SUPPLIER` tables.

### Tasks

1. Start a new SQL\*Plus session under `SSB` user and check whether the `LINEORDER` and `SUPPLIER` tables are in-memory tables.

```
$ sqlplus ssb

Enter password: *****
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> SELECT table_name, inmemory_compression,
              inmemory_priority "PRIORITY", inmemory_distribute
              FROM   user_tables;
 2      3

TABLE_NAME INMEMORY_COMPRESSION PRIORITY INMEMORY_DISTRIBUTE
-----
DATE_DIM
SUPPLIER
LINEORDER

SQL>
```

2. You decide that the default in-memory compression level of all tables that you will define as in-memory tables is `FOR QUERY LOW`. By default, the `INMEMORY_CLAUSE_DEFAULT` initialization parameter is set to an empty value which means `INMEMORY MEMCOMPRESS FOR QUERY LOW`.

```
SQL> SHOW PARAMETER inmemory_clause_default

NAME                                TYPE    VALUE
-----
inmemory_clause_default             string

SQL>
SQL> ALTER SYSTEM SET inmemory_clause_default="INMEMORY
MEMCOMPRESS FOR QUERY LOW";

System altered.
```

```
SQL> SHOW PARAMETER inmemory_clause_default
```

```
NAME                                TYPE      VALUE
-----                                -
inmemory_clause_default  string INMEMORY MEMCOMPRESS FOR QUERY
LOW
SQL>
```

3. Set the `LINEORDER` and `SUPPLIER` tables as in-memory tables by assigning them the same in-memory compression level of `FOR QUERY LOW`. Set an in-memory priority of `CRITICAL` to the `LINEORDER` table and an in-memory priority of `NONE` to the `SUPPLIER` table.

If you omit to specify the in-memory compression level, the in-memory compression level will be set to the default. If you omit to specify the in-memory priority level, the in-memory priority level will be set to `NONE`. This means that no automatic population takes place until you query the table.

```
SQL> ALTER TABLE ssb.lineorder INMEMORY;
```

Table altered.

```
SQL> SELECT table_name, inmemory_compression,
           inmemory_priority "PRIORITY", inmemory_distribute
        FROM   user_tables;
2          3
```

```
TABLE_NAME INMEMORY_COMPRESSION PRIORITY INMEMORY_DISTRIBUTE
-----
DATE_DIM
LINEORDER  FOR QUERY LOW          NONE      AUTO
SUPPLIER
```

```
SQL> ALTER TABLE ssb.lineorder INMEMORY PRIORITY CRITICAL;
```

Table altered.

```
SQL> ALTER TABLE ssb.supplier INMEMORY;
```

Table altered.

```
SQL> SELECT table_name, inmemory_compression,
           inmemory_priority "PRIORITY", inmemory_distribute
        FROM   user_tables;
2          3
```

```

TABLE_NAME INMEMORY_COMPRESSION PRIORITY INMEMORY_DISTRIBUTE
-----
DATE_DIM
LINEORDER  FOR QUERY LOW          CRITICAL AUTO
SUPPLIER    FOR QUERY LOW          NONE      AUTO

SQL>

```

4. Check now that the data of tables with a priority other than NONE is automatically populated into the IM column store.

```

SQL> SELECT segment_name, bytes, inmemory_size,
           bytes_not_populated
       FROM v$im_segments;

2      3
no rows selected

SQL>

```

The result means that the IMCO background process and the workers are not ready to execute the population tasks. The IMCO process wakes up every 2 minutes to tell the SMCO background process to ask the workers to perform the population tasks.

- a. Re-execute the query until the BYTES\_NOT\_POPULATED shows 0. As long as you re-execute the query, you might see the number of bytes populated into the IM column store increasing.

```

SQL> SELECT segment_name, bytes, inmemory_size,
           bytes_not_populated
       FROM v$im_segments;

SEGMENT_NAME      BYTES INMEMORY_SIZE BYTES_NOT_POPULATED
-----
LINEORDER         83886080      34799616              0

SQL>

```

- b. You can eventually see the workers performing the population.

```

SQL> !pgrep -lf orcl
23218 ora_pmon_orcl
23220 ora_psp0_orcl
...
23278 ora_smco_orcl
23280 ora_w000_orcl
23282 ora_w001_orcl
23284 ora_imco_orcl
...

```

```
26410 ora_w002_orcl
26416 ora_w003_orcl
...
```

```
SQL>
```

5. The SUPPLIER table is not candidate for population yet because the NONE priority requires a manual intervention. Query the table to get it populated into the IM column store.

```
SQL> SELECT COUNT(*) FROM ssb.supplier;
```

```

COUNT(*)
-----
        2000

```

```
SQL> SELECT segment_name, bytes, inmemory_size,
           bytes_not_populated
       FROM v$im_segments;
```

SEGMENT_NAME	BYTES	INMEMORY_SIZE	BYTES_NOT_POPULATED
LINEORDER	83886080	34799616	0
SUPPLIER	327680	1179648	0

```
SQL>
```

Now, the SUPPLIER table is populated into the IM column store.

6. Change the in-memory compression level of the LINEORDER table to FOR CAPACITY HIGH and compute the in-memory compression ratio difference.

What is the current in-memory compression ratio?

```
SQL> SELECT segment_name, bytes Disk, inmemory_size,
           inmemory_compression COMPRESSION,
           bytes / inmemory_size COMP_RATIO
       FROM v$im_segments;
```

2	3	4			
SEGMENT_NAME	DISK	INMEMORY_SIZE	COMPRESSION	COMP_RATIO	
SUPPLIER	327680	1179648	FOR QUERY LOW	.277777778	
LINEORDER	83886080	<b>34799616</b>	FOR QUERY LOW	<b>2.41054614</b>	

```
SQL>
```

The compression ratio says that 2.5 bytes of disk are converted under 1 byte of in-memory column store.

- a. Change the in-memory compression level of the `LINEORDER` table.

```
SQL> ALTER TABLE ssb.lineorder
      INMEMORY MEMCOMPRESS FOR CAPACITY HIGH
      PRIORITY CRITICAL;

2
Table altered.

SQL>
```

- b. Check the new in-memory compression level.

```
SQL> SELECT segment_name, bytes Disk, inmemory_size,
      inmemory_compression COMPRESSION,
      bytes / inmemory_size COMP_RATIO
      FROM v$im_segments;

2      3      4
SEGMENT_NAME      DISK INMEMORY_SIZE COMPRESSION      COMP_RATIO
-----
SUPPLIER          327680      1179648 FOR QUERY LOW .277777778

SQL>
```

Note that the `LINEORDER` table is no more in the IM column store. When the in-memory compression level is updated, the information is updated in the data dictionary with the new segment in-memory compression level. The IMCUs data repopulation with the new in-memory compression level takes place only if the segment is re-queried or manually re-populated. When none of these actions is used, the new in-memory compression level is only reflected in NEWLY populated data.

- c. Query the table to get it **fully** populated into the IM column store.

```
SQL> SELECT COUNT(*) FROM ssb.lineorder;

COUNT (*)
-----
700000

SQL> SELECT segment_name, bytes, inmemory_size,
      bytes_not_populated
      FROM v$im_segments;

2      3
SEGMENT_NAME      BYTES INMEMORY_SIZE BYTES_NOT_POPULATED
-----
SUPPLIER          327680      1179648      0

SQL> SELECT segment_name, bytes, inmemory_size,
      bytes_not_populated
```

```

      FROM    v$im_segments;

2      3
SEGMENT_NAME      BYTES INMEMORY_SIZE BYTES_NOT_POPULATED
-----
LINEORDER      83886080      18022400      0
SUPPLIER      327680      1179648      0

SQL>

```

- d. Compute the new in-memory compression ratio.

```

SQL> SELECT segment_name, bytes Disk, inmemory_size,
           inmemory_compression COMPRESSION,
           bytes / inmemory_size COMP_RATIO
      FROM    v$im_segments;

2      3      4
SEGMENT_NAME      DISK INMEMORY_SIZE COMPRESSION      COMP_RATIO
-----
SUPPLIER      327680      1179648 FOR QUERY LOW      .2777777778
LINEORDER      83886080      18022400 FOR CAPACITY HIGH 4.65454545

SQL>

```

Note that the compression ratio is better because it now stores 5 bytes of disk for 1 byte of in-memory column store.

## Practice 17-3: Querying In-Memory Tables

### Overview

In this practice, you will perform queries and see the execution plan difference when you query tables populated into the IM column store or tables cached in the buffer cache only. You will also see the new IM statistics.

### Tasks

1. Start another terminal window session that will be in-memory disabled. A user can enable or disable in-memory queries at the session level by using the `INMEMORY_QUERY` parameter to test the performance difference. The default is "ENABLE".  
In the practice environment, the performance of queries using the IM column store and those using the buffer cache might not be as spectacular as expected due to the limitations in memory and disk space of the servers. You will focus on the execution plans. Use demonstrations in Practice 17-5 to see performance tests on servers with sufficient disk and memory resources.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus ssb
Enter password: *****
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing options and Unified Auditing options
SQL>
```

2. In the second session, set `INMEMORY_QUERY` to "DISABLE" to test the execution of the same query against the buffer cache.

```
SQL> ALTER SESSION SET INMEMORY_QUERY="DISABLE";

Session altered.

SQL>
```

3. Run the following query:

```
SQL> SELECT max(lo_ordtotalprice) most_expensive_order
        FROM ssb.lineorder;

2
MOST_EXPENSIVE_ORDER
-----
                50450906
```



```
SQL> SELECT * FROM table(dbms_xplan.display_cursor());

PLAN_TABLE_OUTPUT
-----
SQL_ID      147v17hqr0mvm, child number 0
-----
SELECT max(lo_ordtotalprice) most_expensive_order      FROM
ssb.lineorder

Plan hash value: 2267213921
-----
| Id | Operation          | Name          | Rows  | Bytes | Cost
(%CPU)| Time              |              |      |      |
-----
|  0 | SELECT STATEMENT   |              |      |      | 2744
(100)|                  |              |      |      |
|  1 |   SORT AGGREGATE   |              |      |      |
|  2 |    TABLE ACCESS FULL| LINEORDER     | 666K | 8466K | 2744
(1)| 00:00:01         |              |      |      |
-----

Note
-----
- dynamic statistics used: dynamic sampling (level=2)

19 rows selected.

SQL>
```

The execution plan shows a traditional TABLE ACCESS FULL operation.

4. Back in the first session, show the default value of the INMEMORY\_QUERY initialization parameter.

```
SQL> SHOW PARAMETER INMEMORY_QUERY

NAME                                TYPE        VALUE
-----
inmemory_query                      string      ENABLE

SQL>
```

## 5. Run the same query:

```
SQL> SELECT max(lo_ordtotalprice) most_expensive_order
        FROM ssb.lineorder;

2
MOST_EXPENSIVE_ORDER
-----
                50450906

SQL> SELECT * FROM table(dbms_xplan.display_cursor());

PLAN_TABLE_OUTPUT
-----
SQL_ID      147v17hqr0mvm, child number 1
-----
SELECT max(lo_ordtotalprice) most_expensive_order      FROM
ssb.lineorder

Plan hash value: 2267213921

-----
| Id | Operation                      | Name | Rows | Bytes | Cost (%CPU)| Time |
|----|-----|-----|-----|-----|-----|-----|
| 0  | SELECT STATEMENT                |      |      |      | 4 (100) |      |
| 1  | SORT AGGREGATE                  |      | 1    | 13    |          |      |
| 2  |  TABLE ACCESS INMEMORY FULL    | LINEORDER | 666K | 8466K | 4 (100) |      |
00:00:01|
-----

Note
-----
- dynamic statistics used: dynamic sampling (level=2)

19 rows selected.

SQL>
```

The execution plan shows the new TABLE ACCESS INMEMORY FULL operation.

6. View new statistics for IM column store.
  - a. Still in the first session, reconnect to reset the session statistics. Then execute a new query.

```
SQL> CONNECT ssb
Enter password: *****
Connected.
SQL> SELECT lo_orderkey, lo_custkey, lo_revenue
        FROM   lineorder
        WHERE  lo_orderkey = 357;
  2      3
LO_ORDERKEY LO_CUSTKEY LO_REVENUE
-----
          357          12079          2825606
          357          12079          6360563
          357          12079          5690272

SQL> SELECT display_name, value
        FROM   v$mystat m, v$statname n
        WHERE  m.statistic# = n.statistic#
        AND    display_name IN (
                    'session logical reads - IM',
                    'IM scan rows', 'IM scan rows valid',
                    'IM scan blocks cache',
                    'IM scan CUs columns accessed' );
  2      3      4      5      6      7      8
DISPLAY_NAME                                VALUE
-----
session logical reads- IM                    10097
IM scan CUs columns accessed                     3
IM scan rows                                700000
IM scan rows valid                            700000
IM scan blocks cache                           0

SQL> save stat1
Created file stat1.sql
SQL>
```

Statistics report how the SQL was executed. 'session logical reads - IM' statistics is the number of blocks scanned in an IMCU. All other statistics show that columns were accessed from the IM Column store. 700000 valid rows were scanned from the IM column store. 'IM scan blocks cache' shows the number of blocks fetched from disk because they were on the IMCU fetch list. The value of 0 means that all blocks were already populated from disk into the IM column store when the query executed.

```
SQL> SELECT display_name, value
        FROM v$mystat m, v$statname n
        WHERE m.statistic# = n.statistic#
        AND display_name IN (
                                'IM scan segments minmax eligible',
                                'IM scan CUs pruned',
                                'IM scan CUs optimized read',
                                'IM scan CUs predicates optimized');
2      3      4      5      6      7      8
DISPLAY_NAME                                VALUE
-----
IM scan CUs predicates optimized              0
IM scan CUs optimized read                   0
IM scan CUs pruned                          0
IM scan segments minmax eligible             1

SQL> save stat2
Created file stat2.sql
SQL>
```

In this example, there are 1 'IM scan segments minmax eligible' which means that there is 1 CU that is eligible for minmax pruning. 0 'IM scan CUs pruned' means that 0 IMCUs did not contain any rows matching the value of LO\_ORDERKEY 357. This means that 0 IMCUs were never scanned. 0 of 'IM scan CUs predicates optimized' means that there were 0 predicates for which no rows pass minmax comparison. Then the CU contained the value of LO\_ORDERKEY 357.

- b. Back in the second "disabled" session, reconnect to reset the session statistics and execute the query.

```
SQL> CONNECT ssb
Enter password: *****
Connected.
SQL> ALTER SESSION SET INMEMORY_QUERY="DISABLE";

Session altered.

SQL> SELECT lo_orderkey, lo_custkey, lo_revenue
        FROM lineorder
        WHERE lo_orderkey = 357;
2      3
```

LO_ORDERKEY	LO_CUSTKEY	LO_REVENUE
-----	-----	-----
357	12079	2825606
357	12079	6360563

  

357	12079	5690272
-----	-------	---------

  

```
SQL> @stat1
```

DISPLAY_NAME	VALUE
-----	-----
session logical reads - IM	0
IM scan CUs columns accessed	0
IM scan rows	0
IM scan rows valid	0
IM scan blocks cache	0

```
SQL>
```

All IM statistics report values of 0 which means that all blocks were read from disk or the buffer cache when the query executed.

```
SQL> @stat2
```

DISPLAY_NAME	VALUE
-----	-----
IM scan CUs predicates optimized	0
IM scan CUs optimized read	0
IM scan CUs pruned	0
IM scan segments minmax eligible	0

  

```
SQL> EXIT
$
```

7. Back into the enabled in-memory session, you will query an in-memory table and a non in-memory table and then display the execution plan.
- a. View the in-memory attributes of the tables.

```
SQL> SELECT table_name, inmemory_compression,
           inmemory_priority
        FROM   user_tables;

 2      3
TABLE_NAME INMEMORY_COMPRESSION INMEMORY_PRIORITY
-----
DATE_DIM
LINEORDER  FOR CAPACITY HIGH      CRITICAL
SUPPLIER   FOR QUERY LOW          NONE

SQL>
```

- b. Execute the \$HOME/labs/Memory/query1.sql script to select data from the in-memory LINEORDER table and the non in-memory DATE\_DIM table.

```
SQL> @query1

      REVENUE
-----
5397325863

SQL>
```

- c. Display the execution plan.

```
SQL> SELECT * FROM table(dbms_xplan.display_cursor());

PLAN_TABLE_OUTPUT
-----
SQL_ID      g8w8ztujuzg2x, child number 0
-----
SELECT SUM(lo_extendedprice * lo_discount) revenue FROM
lineorder l, date_dim d      WHERE  l.lo_orderdate=d.d_datekey
AND      d.d_date='December 24, 1996'

Plan hash value: 2403472142

-----
| Id | Operation                                | Name | Rows | Bytes | Cost (%CPU) | Time |
|----|-----|-----|-----|-----|-----|-----|
| 0  | SELECT STATEMENT                        |      |      |      | 15 (100)    |      |
| 1  | SORT AGGREGATE                          |      | 1    | 72    |              |      |
```

*	2	HASH JOIN		757	54504	15	(14)	00:00:01
	3	JOIN FILTER CREATE	:BF0000	1	33	13	(0)	00:00:01
*	4	TABLE ACCESS FULL	DATE_DIM	1	33	13	(0)	00:00:01
	5	JOIN FILTER USE	:BF0000	666K	24M			
*	6	TABLE ACCESS INMEMORY FULL	LINEORDER	666K	24M			

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

```

2 - access("L"."LO_ORDERDATE"="D"."D_DATEKEY")
4 - filter("D"."D_DATE"='December 24, 1996')
6 - inmemory(SYS_OP_BLOOM_FILTER(:BF0000,"L"."LO_ORDERDATE"))
    filter(SYS_OP_BLOOM_FILTER(:BF0000,"L"."LO_ORDERDATE"))

```

```
Note
```

```
-----
```

```
- dynamic statistics used: dynamic sampling (level=2)
```

```
32 rows selected.
```

```
SQL> EXIT
```

```
$
```

The optimizer chooses to work both with the buffer cache AND the IM column store in different portions of the plan.

When the tables are joined via a hash join, the DATE\_DIM table is scanned first from the buffer cache. The rows that satisfy the WHERE clause predicate for the D\_DATE are used to create a hash table. During the hash table creation, a bloom filter is created based on the D\_DATEKEY join column. The bloom filter is then sent as an additional predicate to the LINEORDER table scan. The table is scanned in IM column store. The resulting rows have their join column LO\_ORDERDATE hashed and it is compared to D\_DATEKEY values in the bloom filter. If a match is found in the bloom filter, that row is sent to the hash join. If no match is found then the row is disregarded.

## Practice 17-4: Exporting and Importing In-Memory Tables (Optional)

### Overview

In this practice, you export and import an in-memory table.

### Tasks

1. Use Oracle Data Pump to export the SUPPLIER table. Execute the \$HOME/labs/Memory/supp.sql script to display the in-memory attributes of the SUPPLIER table.

```
$ sqlplus ssb

Enter password: *****

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> @$HOME/labs/Memory/supp.sql

TABLE_NAME INMEMORY_COMPRESSION PRIORITY In RAC
-----
SUPPLIER   FOR QUERY LOW          NONE     AUTO

SQL>
```

2. In another terminal window that we name Session 2, export the in-memory table.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ rm /u01/app/oracle/admin/orcl/dpdump/supp_exp.dmp
rm: cannot remove
`/u01/app/oracle/admin/orcl/dpdump/supp_exp.dmp': No such file
or directory
$ expdp ssb dumpfile=supp_exp tables=supplier

Password: *****

Connected to: Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
```



```

With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options
Starting "SSB"."SYS_EXPORT_TABLE_01":  ssb/*****
dumpfile=supp_exp tables=supplier
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 320 KB
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
. . exported "SSB"."SUPPLIER"                214.6 KB      2000 rows
Master table "SSB"."SYS_EXPORT_TABLE_01" successfully
loaded/unloaded
*****
Dump file set for SSB.SYS_EXPORT_TABLE_01 is:
  /u01/app/oracle/admin/orcl/dpdump/supp_exp.dmp
Job "SSB"."SYS_EXPORT_TABLE_01" successfully completed at Wed
May 21 05:59:44 2014 elapsed 0 00:00:19

$

```

- From Session 1, drop the table and verify that the table is re-created as an in-memory table with its original attributes.

```

SQL> DROP TABLE supplier PURGE;

Table dropped.

SQL>

```

- From Session 2, import the table.

```

$ impdp ssb dumpfile=supp_exp.dmp TRANSFORM=INMEMORY:Y

Password: *****

Master table "SSB"."SYS_IMPORT_FULL_01" successfully
loaded/unloaded
Starting "SSB"."SYS_IMPORT_FULL_01":  ssb/*****
dumpfile=supp_exp.dmp TRANSFORM=INMEMORY:Y
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "SSB"."SUPPLIER"                214.6 KB      2000 rows
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER

```

```
Job "SSB"."SYS_IMPORT_FULL_01" successfully completed at Wed May
21 06:00:44 2014 elapsed 0 00:00:22
```

```
$
```

5. From Session 1, verify that it is re-created as an in-memory table.

```
SQL> @$HOME/labs/Memory/supp.sql
```

```
TABLE_NAME INMEMORY_COMPRESSION PRIORITY In RAC
-----
SUPPLIER   FOR QUERY LOW          NONE      AUTO
```

```
SQL>
```

The value Y in the INMEMORY transform name is the default. Thus, it was not necessary to mention it in the import command.

6. Still from Session 1, drop the table.

```
SQL> DROP TABLE supplier PURGE;
```

```
Table dropped.
```

```
SQL>
```

7. From Session 2, import the table.

```
$ impdp ssb dumpfile=supp_exp.dmp TRANSFORM=INMEMORY:N
```

```
Password: *****
```

```
Master table "SSB"."SYS_IMPORT_FULL_01" successfully
loaded/unloaded
```

```
Starting "SSB"."SYS_IMPORT_FULL_01":  ssb/*****
dumpfile=supp_exp.dmp TRANSFORM=INMEMORY:N
```

```
Processing object type TABLE_EXPORT/TABLE/TABLE
```

```
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
```

```
. . imported "SSB"."SUPPLIER"          214.6 KB      2000 rows
```

```
Processing object type
```

```
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
```

```
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
```

```
Job "SSB"."SYS_IMPORT_FULL_01" successfully completed at Wed May
21 06:01:38 2014 elapsed 0 00:00:18
```

```
$
```

8. From Session 1, verify that it is re-created as a non in-memory table.

```
SQL> @$HOME/labs/Memory/supp.sql

TABLE_NAME INMEMORY_COMPRESSION PRIORITY In RAC
-----
SUPPLIER    FOR CAPACITY LOW      NONE      AUTO

SQL>
```

Why the table is still an in-memory table but with different in-memory attributes, like the in-memory compression?

The table inherited the in-memory attributes of the tablespace it is stored in. Check the tablespace default in-memory attributes.

```
SQL> COL tablespace_name FORMAT A15
SQL> SELECT tablespace_name, DEF_INMEMORY_PRIORITY,
           DEF_INMEMORY_COMPRESSION, DEF_INMEMORY_DISTRIBUTE
           FROM DBA_TABLESPACES
           WHERE tablespace_name = 'EXAMPLE';
      2      3      4
TABLESPACE_NAME DEF_INMEMORY_PRIORITY DEF_INMEMORY_COMPRESSION
-----
DEF_INMEMORY_DISTRIBUTE
-----
EXAMPLE          NONE          FOR CAPACITY LOW
AUTO
```

9. Still from Session 1, drop the table.

```
SQL> DROP TABLE supplier PURGE;

Table dropped.

SQL>
```

10. From Session 2, import the table by using another transform name.

```
$ impdp ssb dumpfile=supp_exp.dmp TRANSFORM=INMEMORY_CLAUSE:\ "NO
INMEMORY\"

Password: *****
Master table "SSB"."SYS_IMPORT_FULL_01" successfully
loaded/unloaded
Starting "SSB"."SYS_IMPORT_FULL_01":  ssb/*****
dumpfile=supp_exp.dmp TRANSFORM=INMEMORY_CLAUSE:"NO INMEMORY"
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "SSB"."SUPPLIER"                214.6
KB      2000 rows
Processing object type
TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/MARKER
Job "SSB"."SYS_IMPORT_FULL_01" successfully completed at Wed May
21 06:11:56 2014 elapsed 0 00:00:18

$
```

- a. From Session 1, verify that it is re-created as a non in-memory table.

```
SQL> @$HOME/labs/Memory/supp.sql

TABLE_NAME INMEMORY_COMPRESSION PRIORITY In RAC
-----
SUPPLIER

SQL> EXIT
$
```

## Lesson 17-5: Using In-Memory Column Store (*Demonstration*)

---

### Overview

In this practice you can use a browser to view two of the seven In-Memory Column Store demonstrations, available under Oracle Learning Library (OLL).

If you want to hear the explanations that accompany the demos, go to OLL.

The IMCS\_columns demonstration shows how columns are handled in in-memory objects in the IM column store. Columns can be defined as non in-memory in an in-memory table.

The IMCS\_Queries demonstration illustrates how queries on in-memory objects and columns data populated within the IM column store execute. It also shows how fast the queries execute against the IM column store compared to the buffer cache.

### Tasks

1. Launch a browser and enter:  
file:///home/oracle/demos/IMCS\_columns/IMCS\_columns\_player.html.
2. Launch a browser and enter:  
file:///home/oracle/demos/IMCS\_Queries/IMCS\_Queries\_player.html.



# **Practices for Lesson 18: In-Memory Caching**

## **Chapter 18**

## Practices for Lesson 18: Overview

---

### Practices Overview

This practice covers the Full Database In-Memory Caching and Automatic Big Table Caching features in Oracle Database 12c PS1 (12.1.0.2).



## Practice 18-1: Using Automatic Big Table Caching

### Overview

In this practice, you will configure the optional section of the buffer cache, called the automatic big table cache, and monitor the space used in the automatic big table cache by large tables. For ease of reading, Automatic Big Table Caching will be ABTC.

### Tasks

- Before starting the practice, reset the instance memory cache sizes.

```
$ cd ~/labs/Memory
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ ./ABTC_setup.sh
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area 1048576000 bytes
Fixed Size                  2922312 bytes
Variable Size               499123320 bytes
Database Buffers            541065216 bytes
Redo Buffers                 5464064 bytes
Database mounted.
Database opened.
DROP USER abt CASCADE
      *
ERROR at line 1:
ORA-01918: user 'ABT' does not exist

$
```

- Show the sizes of the caches.

```
$ sqlplus / AS SYSDBA

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options
```

```
SQL> SHOW PARAMETER sga_
```

NAME	TYPE	VALUE
sga_max_size	big integer	1000M
sga_target	big integer	844M
unified_audit_sga_queue_size	integer	1048576

```
SQL> SHOW PARAMETER db_cache_size
```

NAME	TYPE	VALUE
db_cache_size	big integer	200M

```
SQL>
```

3. Ensure that Force Full Database Caching is disabled. Force Full Database Caching is not compatible with ABTC.

```
SQL> SELECT force_full_db_caching FROM v$database;

FOR
---
NO

SQL>
```

4. Configure ABTC.

- a. Set the automatic DOP policy to true.

```
SQL> SHOW PARAMETER PARALLEL_DEGREE_POLICY
```

NAME	TYPE	VALUE
parallel_degree_policy	string	MANUAL

```
SQL> ALTER SYSTEM SET PARALLEL_DEGREE_POLICY=AUTO SCOPE=BOTH;

System altered.

SQL>
```

- b. Set the ABTC to 40% of the buffer cache

```
SQL> SHOW PARAMETER DB_BIG_TABLE_CACHE_PERCENT_TARGET
```

NAME	TYPE	VALUE
db_big_table_cache_percent_target	string	0

```
SQL> ALTER SYSTEM SET DB_BIG_TABLE_CACHE_PERCENT_TARGET=40
SCOPE=BOTH;

System altered.

SQL>
```

5. Run the `Query_tables.sql` script. The script connects under `ABT` user and performs a series of `SELECT` statements on small tables.

```
SQL> @Query_tables
Connected.

COUNT (*)
-----
      83

COUNT (*)
-----
      27

COUNT (*)
-----
      10

COUNT (*)
-----
      19

COUNT (*)
-----
      23

COUNT (*)
-----
       4

COUNT (*)
```

```

-----
          25

SQL>

```

6. Display the list of objects loaded into the tables loaded into the ABTC.

```

SQL> SELECT bt_cache_alloc, object_count, memory_buf_alloc
        FROM V$BT_SCAN_CACHE ;

2
BT_CACHE_ALLOC OBJECT_COUNT MEMORY_BUF_ALLOC
-----
          .4              0              0

SQL>

```

Observe that no objects are loaded into the ABTC.

.4 represents the current ratio of the Big Table cache section to the buffer cache when the target ratio is 40.

7. Query a large table in parallel mode.

```

SQL> SELECT /*+ full(LINEORDER) parallel(LINEORDER) */
        count(*)
        FROM SSB.LINEORDER;

2      3
COUNT(*)
-----
700000

SQL>

```

8. Display if the large object is loaded into the ABTC.

```

SQL> SELECT bt_cache_alloc, object_count, memory_buf_alloc
        FROM V$BT_SCAN_CACHE ;

2
BT_CACHE_ALLOC OBJECT_COUNT MEMORY_BUF_ALLOC
-----
          .4              1          10097

SQL>

```

Observe that the object is loaded into the ABTC. 9633 is the number of memory buffers allocated by the ABTC section to objects.

```

SQL> SELECT object_name, size_in_blks, temperature, policy,
        cached_in_mem
        FROM V$BT_SCAN_OBJ_TEMPS v, dba_objects o
        WHERE v.dataobj# = o.data_object_id;

```

```

      2      3      4
OBJECT_NAME  SIZE_IN_BKLS TEMPERATURE POLICY      CACHED_IN_MEM
-----
LINEORDER           9633           3000 MEM_ONLY           9633

SQL> save s1
Created file s1.sql
SQL>

```

Notice the MEM\_ONLY policy value. This means that the large table is fully loaded in the ABTC. This is also the reason why the size of the object being scanned on this instance, in blocks is equal to the number of blocks that are cached/allocated in memory for this object.

9. Now, reselect the large table. The performance of query against the ABTC might not be as spectacular as expected due to the limitations in memory and disk space of the servers. Verify that its temperature increases.

```

SQL> SELECT /*+ full(LINEORDER) parallel(LINEORDER) */
        count(*)
      FROM SSB.LINEORDER;

      2      3
COUNT(*)
-----
      700000

SQL> @s1

OBJECT_NAME  SIZE_IN_BKLS TEMPERATURE POLICY      CACHED_IN_MEM
-----
LINEORDER           9633           6000 MEM_ONLY           9633

SQL>

```

10. Create another big table. Query the new table and check if the new table is loaded into the ABTC.

```

SQL> DROP TABLE ABT.LINEORDER2;
DROP TABLE ABT.LINEORDER2
          *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> CREATE TABLE LINEORDER2 as select * from SSB.LINEORDER;

Table created.

SQL> SELECT /*+ full(LINEORDER2) parallel(LINEORDER2) */

```

```

                count (*)
      FROM ABT.LINEORDER2;
2          3
COUNT (*)
-----
      700000

SQL> SELECT bt_cache_alloc, object_count, memory_buf_alloc
      FROM V$BT_SCAN_CACHE ;
2
BT_CACHE_ALLOC OBJECT_COUNT MEMORY_BUF_ALLOC
-----
                .4                2                19712

SQL>

```

Note that the table is loaded into the ABTC and that the number of memory buffers allocated by the ABTC section to objects has increased.

```

SQL> SELECT object_name, size_in_blks, temperature, policy,
          cached_in_mem
      FROM V$BT_SCAN_OBJ_TEMPS v, dba_objects o
      WHERE v.dataobj# = o.data_object_id
      ORDER BY 1;
2          3          4
OBJECT_NAME  SIZE_IN_BKLS TEMPERATURE POLICY          CACHED_IN_MEM
-----
LINEORDER    10097          8000 MEM_ONLY          10097
LINEORDER2    9615          1000 MEM_ONLY          9615

SQL> save s2
Created file s2.sql
SQL>

```

11. Execute \$HOME/labs/Memory/l3.sql to create another big table and query the data.

```

SQL> @$HOME/labs/Memory/l3.sql

Table created.

700000 rows created.

1400000 rows created.

Commit complete.

```

```

SQL> SELECT /*+ full(LINEORDER3) parallel(LINEORDER3) */
        count(*)
      FROM ABT.LINEORDER3;

2      3
COUNT(*)
-----
2800000

SQL> SELECT bt_cache_alloc, object_count, memory_buf_alloc
      FROM V$BT_SCAN_CACHE ;

2
BT_CACHE_ALLOC OBJECT_COUNT MEMORY_BUF_ALLOC
-----
.400027579      3      29010

SQL>
SQL> @s2

OBJECT_NAME  SIZE_IN_BKLS  TEMPERATURE  POLICY  CACHED_IN_MEM
-----
LINEORDER    10097         10000      MEM_ONLY  10097
LINEORDER2    9615          1000       DISK      9155
LINEORDER3   38417          3000      MEM_PART  18911

SQL>

```

The `LINEORDER3` table is only partially cached in memory and some portion remains on disk. The table cannot be fully cached because the ABTC is full. The sum of the blocks of the three tables is 58129 whereas the number of memory buffers allocated by the ABTC section to objects is 29010. The `LINEORDER2` table has been evicted due to a low temperature.

12. Increase the big table cache percent to 60.

```

SQL> ALTER SYSTEM SET DB_BIG_TABLE_CACHE_PERCENT_TARGET=60;

System altered.

SQL>

```

## 13. Reselect data from the new table.

```
SQL> SELECT /*+ full(LINEORDER3) parallel(LINEORDER3) */
        count(*)
        FROM ABT.LINEORDER3;

2      3
COUNT(*)
-----
2800000

SQL> SELECT bt_cache_alloc, object_count, memory_buf_alloc
        FROM V$BT_SCAN_CACHE ;

2
BT_CACHE_ALLOC OBJECT_COUNT MEMORY_BUF_ALLOC
-----
.531867071      3      34012

SQL>
```

The number of memory buffers allocated by the ABTC section to objects has increased (34012) but is still insufficient to load the three tables.

```
SQL> @s2

OBJECT_NAME  SIZE_IN_BKLS  TEMPERATURE  POLICY  CACHED_IN_MEM
-----
LINEORDER    10097         10000  MEM_ONLY  10097
LINEORDER2   9615          1000  DISK      9155
LINEORDER3   38417         4000  MEM_ONLY  38417

SQL>
```

Now the LINEORDER3 table is now fully cached in memory.

Why the LINEORDER3 table is now fully cached whereas the LINEORDER2 table is no more in the ABTC? When space is required in the big table cache, according to the temperature of the objects, some objects free the space to hotter objects. The LINEORDER2 table is fully replaced by the LINEORDER3 object which has a higher temperature.

## 14. Increase the buffer cache.

```
SQL> ALTER SYSTEM SET DB_CACHE_SIZE=250M;

System altered.

SQL>
```



15. Reselect data from the `LINEORDER2` table.

```
SQL> SELECT /*+ full(LINEORDER2) parallel(LINEORDER2) */
        count(*)
      FROM ABT.LINEORDER2;

2      3

COUNT(*)
-----
700000

SQL> SELECT bt_cache_alloc, object_count, memory_buf_alloc
      FROM V$BT_SCAN_CACHE ;

2
BT_CACHE_ALLOC OBJECT_COUNT MEMORY_BUF_ALLOC
-----
.53892927      3      34472

SQL>
```

Notice that 34472 memory buffers allocated by the ABTC cache section to objects will not be sufficient because the three tables require 58129 memory buffers.

```
SQL> @s2

OBJECT_NAME  SIZE_IN_BLKs  TEMPERATURE  POLICY  CACHED_IN_MEM
-----
LINEORDER    10097         10000 MEM_ONLY  10097
LINEORDER2   9615          2000 MEM_ONLY  9615
LINEORDER3   38417         4000 MEM_ONLY  38417

SQL> EXIT
$
```

## Practice 18-2: Setting and Monitoring Force Full Database Caching

### Overview

In this practice, you will use configure and monitor the Force Full Database Caching feature.

### Tasks

1. Verify that the space used by application data is less than 80% of the instance buffer cache. It would be useless to set the database to Force Full Database Caching if the buffer cache is not large enough to satisfy the requirements of the Force Full Database Caching feature.
  - a. If you have performed the previous practice on In-Memory Column Store or the ABTC, execute the script to cleanup SSB and ABT users, reset the IM column store to 0 byte and the ABTC to 0%.

```
$ cd ~/labs/Memory
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ ./SSB_cleanup.sh
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  884998144 bytes
Fixed Size                  2920440 bytes
Variable Size              335544328 bytes
Database Buffers           541065216 bytes
Redo Buffers                5468160 bytes
Database mounted.
Database opened.
$
```

- b. Connect to the orcl instance.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL>
```

- c. Display the space used by application data, SYSTEM data and UNDO data.

```
SQL> SELECT distinct tablespace_name FROM dba_segments;

TABLESPACE_NAME
-----
UNDOTBS1
SYSAUX
USERS
SYSTEM
EXAMPLE

SQL> SELECT sum(bytes) SPACE_USED FROM dba_segments
        WHERE tablespace_name <> 'SYSAUX';

      2
SPACE_USED
-----
2194735104

SQL>
```

- d. Display the current size of the buffer cache.

```
SQL> SELECT sum(CNUM_SET * BLK_SIZE) CURRENT_CACHE_SIZE
        FROM   x$kcbwds;

      2
CURRENT_CACHE_SIZE
-----
          483409920

SQL>
```

- e. Is SPACE\_USED less than 80% of the instance buffer cache?

```
SQL> SELECT (483409920* 80 / 100) AS "80_OF_BUF" FROM dual;

      80_OF_BUF
-----
      386727936

SQL>
```

No, 2194735104 bytes (application and system storage size) is much more than 386727936 bytes (80% of the buffer cache).

2. Increase the SGA\_TARGET to enable Force Full Database Caching.

```
SQL> ALTER SYSTEM SET sga_target = 3000M scope=spfile;

System altered.
```

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

3. Configure the database for Force Full Database Caching usage.

a. Restart the database instance in mount mode.

```
SQL> STARTUP MOUNT
ORACLE instance started.

Total System Global Area 3154116608 bytes
Fixed Size                  2919328 bytes
Variable Size              402654304 bytes
Database Buffers          2734686208 bytes
Redo Buffers               13856768 bytes
Database mounted.
SQL>
```

b. Set the database in Force Full Database Caching mode.

```
SQL> ALTER DATABASE FORCE FULL DATABASE CACHING;

Database altered.

SQL> ALTER DATABASE OPEN;

Database altered.

SQL>
```

c. Verify that the database is in Force Full Database Caching mode.

```
SQL> SELECT force_full_db_caching FROM v$database;

FOR
---
YES

SQL>
```

d. Check the current buffer cache size.

```
SQL> SELECT sum(CNUM_SET*BLK_SIZE) CURRENT_CACHE_SIZE
        FROM   x$kcbwds;

2
CURRENT_CACHE_SIZE
```

```

-----
                2610413568

SQL>

```

- e. Is the space used less than 80% of the instance buffer cache?

```

SQL> SELECT (2610413568 * 80 / 100) AS "80_OF_BUF" FROM dual;

      80_OF_BUF
-----
    2088330854

SQL> EXIT
$

```

Now, 2194735104 (application and system storage size) is not far from 80% of the buffer cache (2088330854).

4. Verify that the buffer cache is now large enough to be able to load the data from disk. Monitor the Force Full Database Caching.
  - a. Display the 'db block gets from cache', 'consistent gets from cache' and 'physical reads cache' statistics.
    - The 'db block gets from cache' tells the number of times a CURRENT block was requested from the buffer cache.
    - The 'consistent gets from cache' relates the number of times a consistent read was requested for a block from the buffer cache
    - The 'physical reads cache' displays the total number of data blocks read from disk into buffer cache.

```

SQL> SELECT name, value FROM V$SYSSTAT
      WHERE name IN ('db block gets from cache',
                    'consistent gets from cache',
                    'physical reads cache');

2      3      4

NAME                                     VALUE
-----
db block gets from cache                  3033
consistent gets from cache               1785583
physical reads cache                     13112

SQL> save s3
Created file s3.sql
SQL>

```

- b. Run queries on all application tables.

```
SQL> @list_tab
```

```
...
COUNT (*)
-----
          72

COUNT (*)
-----
       1826

...

COUNT (*)
-----
          8
```

```
SQL>
```

- c. Then check the 'physical reads cache' statistics and calculate the buffer cache hit ratio:

```
1 - ('physical reads cache' / ('consistent gets from cache' +
'db block gets from cache'))
```

```
SQL> @s3
```

NAME	VALUE
db block gets from cache	3595
consistent gets from cache	1868971
physical reads cache	<u>20209</u>

```
SQL> SELECT 1 - (20209 / (1868971 + 3595)) Hit FROM dual;
```

```
          HIT
-----
.989207857
```

```
SQL>
```

5. After re-executing the same queries, observe if the 'physical reads cache' remains stable. All data is in the buffer cache and the buffer cache hit ratio remains excellent.

```
SQL> @list_tab
...
COUNT(*)
-----
          72

COUNT(*)
-----
        1826

...

COUNT(*)
-----
          8

SQL>
SQL> @s3

NAME                                VALUE
-----
db block gets from cache              3601
consistent gets from cache          1934530
physical reads cache                  20210

SQL> SELECT 1 - (20210 / (1934530 + 3601)) Hit FROM dual;

          HIT
-----
.989572428

SQL> EXIT
$
```

Regularly verify that the hit ratio remains stable. If the hit ratio decreases, check if the disk data has not increased. In this case, reconsider the buffer cache size.





# **Practices for Lesson 19: SQL Tuning Enhancements**

## **Chapter 19**

## Practices for Lesson 19: Overview

---

### Practices Overview

This practice covers the dynamic plans part of the Adaptive Execution Plans feature in Oracle Database 12c.

## Practice 19-1: Using Dynamic Plans

### Overview

In this practice, you will use the dynamic plans part of the Adaptive Execution Plans feature.

### Tasks

1. Make sure you are in the ~/labs/Tuning directory.

```
$ cd ~/labs/Tuning
$
```

2. After having configured the database instance for In-Memory Caching new features, reset the database instance without ABTC or Force Full DB caching.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT
ORACLE instance started.

Total System Global Area 3154116608 bytes
Fixed Size                  2919328 bytes
Variable Size               771753056 bytes
Database Buffers            2365587456 bytes
Redo Buffers                 13856768 bytes
Database mounted.
SQL> ALTER DATABASE NO FORCE FULL DATABASE CACHING;

Database altered.

SQL> ALTER SYSTEM SET db_big_table_cache_percent_target=0
SCOPE=BOTH;

System altered.
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```
SQL> ALTER DATABASE OPEN;
```

```
Database altered.
```

```
SQL> SELECT force_full_db_caching FROM V$DATABASE;
```

```
FOR
```

```
---
```

```
NO
```

```
SQL> SHOW PARAMETER db_big_table_cache_percent_target
```

NAME	TYPE	VALUE
-----		
db_big_table_cache_percent_target	string	0

```
SQL> EXIT
```

```
$
```

3. Use the `Tuning_setup.sh` script to create the OE1 and SH1 accounts in ORCL database, grant them the `SELECT ANY DICTIONARY` privilege and finally create OE1 and SH1 tables.

```
$ ./Tuning_setup.sh
```

```
SQL*Plus: Release 12.1.0.2.0 Production on Thu May 22 05:44:10
2014
```

```
Copyright (c) 1982, 2014, Oracle. All rights reserved.
```

```
Connected to:
```

```
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
```

```
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options
```

```
SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL>
SQL> SQL> SQL> SQL> SQL> SQL> SQL> Connected.
```

```
SQL> 2 SQL> Connected.
```

```
SQL> SQL> SQL> Disconnected from Oracle Database 12c Enterprise
Edition Release 12.1.0.2.0 - 64bit Production
```

```
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options
```

```
$
```

4. From the same SQL\*Plus session, connect as user OE1 and use the \$HOME/labs/Tuning/explain1.sql script to show the execution plan of the following query without executing it:

```
select /*+ monitor*/ product_name
from   order_items o, product_information p
where  o.unit_price = 15
       and quantity > 1
       and p.product_id = o.product_id;
```

```
$ sqlplus oe1
Enter password: *****

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options
SQL>
SQL> @explain1

Explained.

SQL> set pages 100
SQL> set lines 300
SQL> select * from table(dbms_xplan.display());

PLAN_TABLE_OUTPUT
-----
Plan hash value: 1255158658
-----
| Id | Operation                                | Name                | Rows  | Bytes |
Cost (%CPU)| Time          |                      |       |      |
-----|-----|-----|-----|-----|
| 0 | SELECT STATEMENT                        |                      | 4     | 128   |
(0) | 00:00:01 |                      |       |      |
| 1 |  NESTED LOOPS                          |                      | 4     | 128   |
7 (0) | 00:00:01 |                      |       |      |
| 2 |    NESTED LOOPS                        |                      | 4     | 128   |
7 (0) | 00:00:01 |                      |       |      |
|* 3 |      TABLE ACCESS FULL                | ORDER_ITEMS         | 4     | 48    |
3 (0) | 00:00:01 |                      |       |      |
|* 4 |        INDEX UNIQUE SCAN                | PRODUCT_INFORMATION_PK | 1     |       |
| 0 |          (0) | 00:00:01 |       |      |
| 5 |      TABLE ACCESS BY INDEX ROWID      | PRODUCT_INFORMATION | 1     | 20    |
| 1 |        (0) | 00:00:01 |       |      |
-----
```

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

```
-----
```

```
3 - filter("O"."UNIT_PRICE"=15 AND "QUANTITY">1)
```

```
4 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")
```

```
Note
```

```
-----
```

```
- this is an adaptive plan
```

```
22 rows selected.
```

```
SQL>
```

5. What do you observe?  
The plan is using a simple NESTED LOOP join.
6. Now, execute the \$HOME/labs/Tuning/sell.sql to execute the same query:

```
SQL> @sell
```

```
PRODUCT_NAME
```

```
-----
```

```
Screws <B.28.S>
```

```
Screws <B.28.S>
```

```
Screws <B.28.S>
```

```
Screws <B.28.S>
```

```
...
```

```
Screws <B.28.S>
```

```
Screws <B.28.S>
```

```
52 rows selected.
```

```
SQL>
```

7. Show the resulting execution plan:

```
SQL> select * from table(dbms_xplan.display_cursor());
```

```
PLAN_TABLE_OUTPUT
```

```
-----
```

```
SQL_ID      1h1hsr7dvauan, child number 0
```

```
-----
```

```
select /*+ monitor*/ product_name      from    order_items o,
product_information p                    where   o.unit_price = 15 and
quantity
> 1 and p.product_id = o.product_id
```



```

SQL> connect / as sysdba
Connected.
SQL> select d.directive_id, o.object_type, o.object_name,
           o.subobject_name col_name, d.type, d.state, d.reason
           from   dba_sql_plan_directives d, dba_sql_plan_dir_objects
           o
           where  d.DIRECTIVE_ID=o.DIRECTIVE_ID
           and    o.owner = 'OE1'
           order by d.directive_id;
           2      3      4      5      6
no rows selected

SQL> /

no rows selected

SQL> save d1
Created file d1.sql
SQL>

```

10. **-- You have to wait for a while before it is persisted. MMON is responsible for the flush.**  
**-- In Database 12c DML monitoring and column usage information flush has been transferred to MMON instead of SMON.**  
 If you do not want to wait, execute the `dbms_spd.flush_sql_plan_directive` procedure.

```

SQL> exec dbms_spd.flush_sql_plan_directive

PL/SQL procedure successfully completed.

SQL> @d1

```

DIRECTIVE_ID	OBJECT	OBJECT_NAME	COL_NAME
TYPE	STATE	REASON	
2783507998265655681	COLUMN	ORDER_ITEMS	UNIT_PRICE
DYNAMIC_SAMPLING	USABLE	SINGLE TABLE	CARDINALITY MISESTIMATE
2783507998265655681	TABLE	ORDER_ITEMS	
DYNAMIC_SAMPLING	USABLE	SINGLE TABLE	CARDINALITY MISESTIMATE
2783507998265655681	COLUMN	ORDER_ITEMS	QUANTITY
DYNAMIC_SAMPLING	USABLE	SINGLE TABLE	CARDINALITY MISESTIMATE

```

SQL>

```



## Practice 19-2: Using Re-Optimization

---

### Overview

In this practice, you discover how the re-optimization (Cardinality Feedback) part of the Adaptive Execution Plans feature in Oracle Database 12c works.

### Tasks

1. Execute the following query. Note the `gather_plan_statistics` hint is used to display the actual number of rows returned from each operation in the plan. This will allow you to compare the optimizer's estimates with the actual number of rows returned.

```
SELECT /*+ gather_plan_statistics HINT1 */ c.cust_first_name,
      c.cust_last_name, sum(s.amount_sold)
FROM   sh1.customers c, SH1.sales s
WHERE  c.cust_id=s.cust_id
AND    c.cust_city='Los Angeles'
AND    c.cust_state_province='CA'
AND    c.country_id=52790
AND    s.time_id='09-NOV-00'
GROUP BY c.cust_first_name, c.cust_last_name;
```

Use the `$HOME/labs/Tuning/sel2.sql` script to execute the query.

```
SQL> connect sh1
Enter password: *****
Connected.
SQL> set pages 9999
SQL> set lines 300
SQL> COL sql_text format a30
SQL> @sel2

no rows selected

SQL>
```

2. Display the associated execution plan. What do you observe?  
 There is a large difference between the estimated (E-Rows) and the actual number of rows returned (A-Rows). This statement looks like a candidate for re-optimization.

```
SQL> SELECT * FROM
table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'));

SQL_ID      bkc4p9hd7g3cj, child number 0
-----
select /*+ gather_plan_statistics HINT1 */
c.cust_first_name,
c.cust_last_name, sum(s.amount_sold)   from      sh1.customers c,
SH1.sales s       where c.cust_id=s.cust_id and
c.cust_city='Los
Angeles'          and   c.cust_state_province='CA'          and
c.country_id=52790 and   s.time_id='09-NOV-00' group by
c.cust_first_name, c.cust_last_name

Plan hash value: 2957067879

-----
| Id | Operation                      | Name           | Starts |
E-Rows | A-Rows | A-Time   | Buffers | Reads | OMem | lMem | Used-Mem |
-----
|  0 | SELECT STATEMENT                |                |        |
|00:00:00.03 | 1568 | 205 |          |          |          | 1 |          | 0
|  1 | HASH GROUP BY                   |                |        | | | | | |
|  0 |00:00:00.03 | 1568 | 205 | 909K | 909K |          | 1 | 1 |
|* 2 | HASH JOIN                       |                |        |
|  0 |00:00:00.03 | 1568 | 205 | 1368K | 1368K | 1304K (0) | 1 | 1 |
|* 3 | TABLE ACCESS FULL              | CUSTOMERS      |        |
| 1 | 932 |00:00:00.04 | 1521 | 0 |          |          | 1 |
|  4 | TABLE ACCESS BY INDEX ROWID BATCHED | SALES          |        |
|1260 | 1076 |00:00:00.01 | 47 | 205 |          |          | 1 |
|  5 | BITMAP CONVERSION TO ROWIDS     |                |        |
|1076 |00:00:00.01 | 2 | 7 |          |          | 1 |
|* 6 | BITMAP INDEX SINGLE VALUE       | SALES_TIME_BIX |        |
|  1 |00:00:00.01 | 2 | 7 |          |          | 1 |

-----

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

      2 - access("C"."CUST_ID"="S"."CUST_ID")
      3 - filter(("C"."CUST_CITY"='Los Angeles' AND
"C"."CUST_STATE_PROVINCE"='CA' AND "C"."COUNTRY_ID"=52790))
      6 - access("S"."TIME_ID"='09-NOV-00')

30 rows selected.

SQL>
```

3. How would you confirm this statement will be re-optimized?  
 You can confirm that by checking the value of the `is_reoptimizable` column in `v$sql`. This column indicates that this statement will be re-parsed on the next execution and information learnt on the first execution about the actual number of rows returned will be used to generate a better plan.

```
SQL> COL is_reopt FORMAT A8
SQL> select sql_id, child_number, sql_text,
           is_reoptimizable "is_reopt"
       from v$sql
       where sql_text like '%+ gather_plan_statistics
HINT1%SH1%';      2      3      4
SQL_ID          CHILD_NUMBER SQL_TEXT                      is_reopt
-----
--
5a297fd6x57cc          0 select sql_id, child_number, s N
                        ql_text,                      is_reopti
                        mizable "is_reopt"            from
                        v$sql      where sql_text li
                        ke '%+ gather_plan_statistics
                        HINT1%SH1%'

bkc4p9hd7g3cj          0 select /*+ gather_plan_statist y
                        ics HINT1 */      c.cust_
                        first_name, c.cust_last_name,
                        sum(s.amount_sold)      from
                        sh1.customers c, SH1.sales s
                        where c.cust_id=s.cust_id
                        and   c.cust_city='Los An
                        geles'      and   c.cust_state
                        _province='CA'      and   c.co
                        untry_id=52790      and   s.ti
                        me_id='09-NOV-00'      group b
                        y c.cust_first_name, c.cust_la
                        st_name

SQL>
```

4. Confirm your guess is correct:

```
SQL> @sel2

no rows selected

SQL> select * from
table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'));
```

## PLAN\_TABLE\_OUTPUT

SQL\_ID bkc4p9hd7g3cj, child number 1

```
select /*+ gather_plan_statistics HINT1 */
c.cust_first_name,
c.cust_last_name, sum(s.amount_sold) from sh1.customers c,
SH1.sales s where c.cust_id=s.cust_id and
c.cust_city='Los
Angeles' and c.cust_state_province='CA' and
c.country_id=52790 and s.time_id='09-NOV-00' group by
c.cust_first_name, c.cust_last_name
```

Plan hash value: 520406099

Id	Operation	Name	Starts				
E-Rows	A-Rows	A-Time	Buffers	OMem	lMem	Used-Mem	
0	SELECT STATEMENT		1				
0	00:00:00.01	1568					
1	HASH GROUP BY		1	1017			
0	00:00:00.01	1568	909K	909K			
* 2	HASH JOIN		1	1017			
0	00:00:00.01	1568	1817K	1817K	1640K (0)		
3	TABLE ACCESS BY INDEX ROWID BATCHED	SALES		1			
1260	1076	00:00:00.01	47				
4	BITMAP CONVERSION TO ROWIDS		1				
1076	00:00:00.01	2					
* 5	BITMAP INDEX SINGLE VALUE	SALES_TIME_BIX	1				
1	00:00:00.01	2					
* 6	TABLE ACCESS FULL	CUSTOMERS		1			
932	932	00:00:00.01	1521				

Predicate Information (identified by operation id):

```
2 - access("C"."CUST_ID"="S"."CUST_ID")
5 - access("S"."TIME_ID"='09-NOV-00')
6 - filter(("C"."CUST_CITY"='Los Angeles' AND
"C"."CUST_STATE_PROVINCE"='CA' AND "C"."COUNTRY_ID"=52790))
```

Note

- statistics feedback used for this statement

```
34 rows selected.
```

```
SQL>
```

5. Check if the new child cursor created used feedback statistics. If not proceed with steps 6 and 7.

```
SQL> COL USE_FEEDBACK_STATS FORMAT A18
SQL> select CHILD_NUMBER, USE_FEEDBACK_STATS
"USE_FEEDBACK_STATS"
      from v$sql_shared_cursor
      where SQL_ID = 'bkc4p9hd7g3cj' ;
```

```
      2      3
CHILD_NUMBER USE_FEEDBACK_STATS
-----
              0 Y
              1 Y
```

```
SQL>
```

6. Because the second child cursor created did not use feedback stats (set to "Y"), it is still not the best and need to be hard parsed again with better cardinality estimates.

```
SQL> @sel2
```

```
no rows selected
```

```
SQL> select * from
table(dbms_xplan.display_cursor(FORMAT=>'ALLSTATS LAST'));
```

```
PLAN_TABLE_OUTPUT
```

```
-----
-
SQL_ID      bkc4p9hd7g3cj, child number 2
-----
select /*+ gather_plan_statistics HINT1 */
c.cust_first_name,
c.cust_last_name, sum(s.amount_sold)   from      sh1.customers c,
sh1.sales s      where c.cust_id=s.cust_id and
c.cust_city='Los
Angeles'        and   c.cust_state_province='CA'        and
c.country_id=52790   and   s.time_id='09-NOV-00' group by
c.cust_first_name, c.cust_last_name
```

```
Plan hash value: 520406099
```

```
-----
| Id | Operation                               | Name           | Starts |
E-Rows | A-Rows | A-Time   | Buffers | OMem | lMem | Used-Mem |
-----
|  0 | SELECT STATEMENT                       |                |      1 |
  0 | 00:00:00.02 | 1568 |      |      |      |          |
|  1 | HASH GROUP BY                         |                |      1 | 1042 |
  0 | 00:00:00.02 | 1568 | 909K | 909K |      |          |
|*  2 | HASH JOIN                             |                |      1 | 1042 |
  0 | 00:00:00.02 | 1568 | 1817K | 1817K | 1609K (0) |
|  3 | TABLE ACCESS BY INDEX ROWID BATCHED | SALES          |      1 |
1260 | 1076 | 00:00:00.01 | 47 |      |      |          |
|  4 | BITMAP CONVERSION TO ROWIDS           |                |      1 |
1076 | 00:00:00.01 | 2 |      |      |      |          |
|*  5 | BITMAP INDEX SINGLE VALUE             | SALES_TIME_BIX |      1 |
  1 | 00:00:00.01 | 2 |      |      |      |          |
|*  6 | TABLE ACCESS FULL                   | CUSTOMERS      |      1 |
955 | 932 | 00:00:00.01 | 1521 |      |      |          |
-----

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

      2 - access("C"."CUST_ID"="S"."CUST_ID")
      5 - access("S"."TIME_ID"='09-NOV-00')
      6 - filter(("C"."CUST_CITY"='Los Angeles' AND
"C"."CUST_STATE_PROVINCE"='CA' AND "C"."COUNTRY_ID"=52790))

Note
-----
- dynamic statistics used: dynamic sampling (level=2)
- 1 Sql Plan Directive used for this statement

35 rows selected.

SQL>
```

7. Check that a new child cursor was created:

```
SQL> select CHILD_NUMBER, USE_FEEDBACK_STATS
"USE_FEEDBACK_STATS"
      from v$sql_shared_cursor
      where SQL_ID = 'bkc4p9hd7g3cj' ;

      2      3
CHILD_NUMBER USE_FEEDBACK_STATS
-----
          0 Y
          1 Y
          2 N
```

```
SQL>
```

The third child cursor created, because it already used feedback stats, is set to "N;" it will not be reparsed.

8. Force the SQL plan directive to be flushed and check to see it was persisted into the data dictionary:

```
SQL> connect / as sysdba
Connected.
SQL> col state format a5
SQL> col subobject_name format a11
SQL> col col_name format a20
SQL> col object_name format a20
SQL> col dir_id format a23
SQL> col owner format a5
SQL> col state format A20
SQL> set echo on
SQL>
SQL> exec dbms_spd.flush_sql_plan_directive

PL/SQL procedure successfully completed.

SQL> select to_char(d.directive_id) dir_id,
           o.object_name, o.subobject_name col_name,
           o.object_type, d.type, d.state, d.reason
       from   dba_sql_plan_directives d,
           dba_sql_plan_dir_objects o
       where  d.DIRECTIVE_ID=o.DIRECTIVE_ID
       and    o.owner in ('SH1')
       order by 1,2,3,4;

2      3      4      5      6      7      8
```

DIR_ID	OBJECT_NAME	COL_NAME	OBJECT
TYPE	STATE	REASON	
-----	-----	-----	-----
6718791204543937877	CUSTOMERS	COUNTRY_ID	COLUMN
DYNAMIC_SAMPLING	USABLE	SINGLE	TABLE CARDINALITY MISESTIMATE
6718791204543937877	CUSTOMERS	CUST_CITY	COLUMN
DYNAMIC_SAMPLING	USABLE	SINGLE	TABLE CARDINALITY MISESTIMATE
6718791204543937877	CUSTOMERS	CUST_STATE	PROVINCE COLUMN
DYNAMIC_SAMPLING	USABLE	SINGLE	TABLE CARDINALITY MISESTIMATE
6718791204543937877	CUSTOMERS		TABLE
DYNAMIC_SAMPLING	USABLE	SINGLE	TABLE CARDINALITY MISESTIMATE



```
SQL>
```

9. Drop OE1 and SH1 accounts.

```
SQL> DROP USER oe1 CASCADE;
```

```
User dropped.
```

```
SQL> DROP USER sh1 CASCADE;
```

```
User dropped.
```

```
SQL> EXIT
```

```
$
```



## **Practices for Lesson 20: Resource Manager and Other Performance Enhancements**

### **Chapter 20**

## Practices for Lesson 20: Overview

---

### Practices Overview

In this practice, you create two CDB Resource Manager plans and associated directives to limit CPU resources used by two PDBs.

## Practice 20-1: Using CDB Resource Manager Plans and Directives

---

### Overview

In this practice, you create two CDB Resource Manager plans and associated directives to limit CPU resources used by two PDBs.

### Tasks

1. Connect to the root of `cdb2` as `SYSDBA` and cleanup your environment by executing the `rsrc_cleanup.sql` script. The script will close all PDBs except `PDB1_1` and `PDB2_2`.
  - a. Make sure you are in the `~/labs/RM` directory and your environment points to the `cdb2` instance.

```
$ cd ~/labs/RM
$ . oraenv
ORACLE_SID = [orcl] ? cdb2
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

- b. Execute the `rsrc_cleanup.sql` script.
    - 1) Start up the multitenant container database instance if not already done.

```
$ sqlplus / as sysdba

Connected to an idle instance.

SQL> STARTUP
```

ORACLE instance started.

Total System Global Area 4697620480 bytes  
 Fixed Size 2924848 bytes  
 Variable Size 989859536 bytes  
 Database Buffers 3690987520 bytes  
 Redo Buffers 13848576 bytes

Database mounted.

Database opened.

SQL> @rsrc\_cleanup.sql

Pluggable database altered.

Pluggable database altered.

NAME	CON_ID	OPEN_MODE
-----	-----	-----
PDB\$SEED	2	READ ONLY
<b>PDB1_1</b>	6	<b>READ WRITE</b>
PDB2	3	MOUNTED
<b>PDB2_2</b>	4	<b>READ WRITE</b>
PDB_ORCL2	5	MOUNTED

System altered.

NAME  
 -----  
 DEFAULT\_MAINTENANCE\_PLAN

System altered.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

```
BEGIN
DBMS_Resource_Manager.Delete_CDB_Plan_Directive('fairplan',
'pdb1_1'); END;
```

\*

ERROR at line 1:

ORA-29358: resource plan FAIRPLAN does not exist  
 ORA-06512: at "SYS.DBMS\_RMIN\_SYS", line 3158

```

ORA-06512: at "SYS.DBMS_RESOURCE_MANAGER", line 1526
ORA-06512: at line 1

BEGIN
DBMS_Resource_Manager.Delete_CDB_Plan_Directive('fairplan',
'pdb2_2'); END;

*
ERROR at line 1:
ORA-29358: resource plan FAIRPLAN does not exist
ORA-06512: at "SYS.DBMS_RMIN_SYS", line 3158
ORA-06512: at "SYS.DBMS_RESOURCE_MANAGER", line 1526
ORA-06512: at line 1

BEGIN DBMS_Resource_Manager.Delete_CDB_Plan('fairplan'); END;

*
ERROR at line 1:
ORA-29358: resource plan FAIRPLAN does not exist
ORA-06512: at "SYS.DBMS_RMIN_SYS", line 2851
ORA-06512: at "SYS.DBMS_RESOURCE_MANAGER", line 1372
ORA-06512: at line 1

BEGIN
DBMS_Resource_Manager.Delete_CDB_Plan_Directive('unfairplan',
'pdb1_1'); END;

*
ERROR at line 1:
ORA-29358: resource plan UNFAIRPLAN does not exist
ORA-06512: at "SYS.DBMS_RMIN_SYS", line 3158
ORA-06512: at "SYS.DBMS_RESOURCE_MANAGER", line 1526
ORA-06512: at line 1

BEGIN
DBMS_Resource_Manager.Delete_CDB_Plan_Directive('unfairplan',
'pdb2_2'); END;

*
ERROR at line 1:
ORA-29358: resource plan UNFAIRPLAN does not exist
ORA-06512: at "SYS.DBMS_RMIN_SYS", line 3158
ORA-06512: at "SYS.DBMS_RESOURCE_MANAGER", line 1526

```

```
ORA-06512: at line 1

BEGIN DBMS_Resource_Manager.Delete_CDB_Plan('unfairplan'); END;

*
ERROR at line 1:
ORA-29358: resource plan UNFAIRPLAN does not exist
ORA-06512: at "SYS.DBMS_RMIN_SYS", line 2851
ORA-06512: at "SYS.DBMS_RESOURCE_MANAGER", line 1372
ORA-06512: at line 1

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

SQL> EXIT
$
```



2. Open a terminal window (it will be referred to as window1) to connect to `pdb1_1` in `cdb2` and create a PL/SQL procedure that burns CPU in `PDB1_1` as the `SYSTEM` user. You can use the `create_burn_cpu.sql` script to create the procedure after connecting to `PDB1_1`.

```
$ cd ~/labs/RM
$ . oraenv
ORACLE_SID = [orcl] ? cdb2
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
$ sqlplus system@pdb1_1

Enter password: *****

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> @create_burn_cpu.sql

Procedure created.

SQL>
```

3. Open a second terminal window (it will be referred to as window2) to connect to `pdb2_2` in `cdb2` and create a PL/SQL procedure that burns CPU in `PDB2_2` as the `SYSTEM` user. You can use the `create_burn_cpu.sql` script to create the procedure after connecting to `PDB2_2`.

```
$ cd ~/labs/RM
$ . oraenv
ORACLE_SID = [cdb1] ? cdb2
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus system@pdb2_2

Enter password: *****

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
```

With the Partitioning, OLAP, Advanced Analytics, Real Application Testing and Unified Auditing options

```
SQL> @create_burn_cpu.sql
```

```
Procedure created.
```

```
SQL>
```

4. From window1, execute the \$HOME/labs/RM/plan.sql to create two new CDB plans called FAIRPLAN and UNFAIRPLAN.  
FAIRPLAN should give one share to both PDB1\_1 and PDB2\_2, and UNFAIRPLAN should give one share to PDB1\_1 and five shares to PDB2\_2.

```
SQL> alter session set container = CDB$Root;
```

```
Session altered.
```

```
SQL> @plan
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
SQL>
```

5. Still from window1, make sure both plans and associated directives were created correctly. Execute the \$HOME/labs/RM/dir.sql script.

```
SQL> @dir

PLAN
-----
FAIRPLAN
UNFAIRPLAN

PLAN                                PLUGGABLE_DATABASE    SHARES
-----
FAIRPLAN                            PDB1_1                  1
FAIRPLAN                            PDB2_2                  1
UNFAIRPLAN                          PDB1_1                  1
UNFAIRPLAN                          PDB2_2                  5

SQL>
```

6. From window1, activate the CDB plan FAIRPLAN.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> alter system set resource_manager_plan = fairplan;

System altered.

SQL> select Name from v$Rsrc_Plan where Con_ID = 1;

NAME
-----
FAIRPLAN

SQL>
```

7. From window1, connect as the SYSTEM user in PDB1\_1 and set SERVEROUTPUT variable to ON.

```
SQL> CONNECT system@pdb1_1
Enter password: *****
Connected.
SQL> set serveroutput on
SQL>
```

8. From window2, connect as the SYSTEM user in PDB2\_2 and set SERVEROUTPUT variable to ON.

```
SQL> CONNECT system@pdb2_2
Enter password: *****
Connected.
SQL> set serveroutput on
SQL>
```

9. **DO NOT WAIT AND GO TO STEP 10 RIGHT AFTER:** From window1, execute the CPU burner procedure you created at step 2.

```
SQL> EXEC Burn_CPU_For_RM_Demo()

CPU:   109.6 Wall:  222.6 k: 2000000000
PL/SQL procedure successfully completed.

SQL>
```

10. From window2, execute the CPU burner procedure you created at step 3.

```
SQL> EXEC Burn_CPU_For_RM_Demo()

CPU:   109.9 Wall:  227.1 k: 2000000000
PL/SQL procedure successfully completed.

SQL>
```

11. What do you observe?  
Both procedures finish their execution almost at the same time, and have both consumed almost the same CPU and wall-clock time during their execution.  
This is expected because each PDB is receiving one share of CPU.
12. From window1, connect as user SYS in the root, and change the Resource Manager plan to UNFAIRPLAN.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> alter system set resource_manager_plan = unfairplan;

System altered.

SQL> select Name from v$rsrc_Plan where Con_ID = 1;

NAME
-----
UNFAIRPLAN

SQL>
```

13. **DO NOT WAIT AND GO TO STEP 14 RIGHT AFTER:** From window1, connect as user SYSTEM in PDB1\_1 and execute the CPU burner procedure you created at step 2.

```
SQL> CONNECT system@pdb1_1
Enter password: *****
Connected.
SQL>
SQL> set serveroutput on
SQL>
SQL> execute Burn_CPU_For_RM_Demo();
CPU:      109.0 Wall:   226.5 k: 2000000000

PL/SQL procedure successfully completed.

SQL>
```

14. From window2 execute the CPU burner procedure you created at step 3.

```
SQL> execute Burn_CPU_For_RM_Demo();
CPU:      109.8 Wall:   135.7 k: 2000000000

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

15. What do you observe?  
Now, execution of the CPU burner procedure takes much longer to execute in PDB1\_1 than in PDB2\_2.  
This is expected because PDB2\_2 is assigned five shares while PDB1\_1 only one.  
However, the difference is not five times slower simply because once the procedure executed in PDB2\_2, all CPU cycles goes to PDB1\_1.

16. From window1, connect as user `sys` in the root, and change the CDB Resource Manager plan back to its default and open all the pluggable databases.

```
SQL> CONNECT / as sysdba
Connected.

SQL> alter system set resource_manager_plan = '';

System altered.

SQL> select Name from v$Rsrc_Plan where Con_ID = 1;

NAME
-----
ORA$INTERNAL_CDB_PLAN

SQL> alter pluggable database all open;

Pluggable database altered.

SQL> EXIT
$
```

## Practice 20-2: Using Multi-Process Multi-Threaded Architecture

### Overview

In this practice, you switch `cdb2` to use the multi-process multi-threaded architecture.

1. From a terminal window, connected as the `oracle` user, list all processes and threads used to run your `cdb2` instance.

```
$ ps -eLo "pid tid comm args" |grep cdb2
2463 2463 ora_pmon_cdb2 ora_pmon_cdb2
2465 2465 ora_psp0_cdb2 ora_psp0_cdb2
2467 2467 ora_vktm_cdb2 ora_vktm_cdb2
2471 2471 ora_gen0_cdb2 ora_gen0_cdb2
2473 2473 ora_mman_cdb2 ora_mman_cdb2
2477 2477 ora_diag_cdb2 ora_diag_cdb2
2479 2479 ora_dbrm_cdb2 ora_dbrm_cdb2
2481 2481 ora_vkrm_cdb2 ora_vkrm_cdb2
2483 2483 ora_dia0_cdb2 ora_dia0_cdb2
2485 2485 ora_dbw0_cdb2 ora_dbw0_cdb2
2487 2487 ora_lgwr_cdb2 ora_lgwr_cdb2
2489 2489 ora_ckpt_cdb2 ora_ckpt_cdb2
2491 2491 ora_lg00_cdb2 ora_lg00_cdb2
2493 2493 ora_smon_cdb2 ora_smon_cdb2
2495 2495 ora_lg01_cdb2 ora_lg01_cdb2
2497 2497 ora_reco_cdb2 ora_reco_cdb2
2499 2499 ora_lreg_cdb2 ora_lreg_cdb2
2501 2501 ora_pxmnl_cdb2 ora_pxmnl_cdb2
2503 2503 ora_mmon_cdb2 ora_mmon_cdb2
2505 2505 ora_mmn1_cdb2 ora_mmn1_cdb2
2507 2507 ora_d000_cdb2 ora_d000_cdb2
2509 2509 ora_s000_cdb2 ora_s000_cdb2
2526 2526 ora_rvwr_cdb2 ora_rvwr_cdb2
2529 2529 ora_tmon_cdb2 ora_tmon_cdb2
2531 2531 ora_arc0_cdb2 ora_arc0_cdb2
2533 2533 ora_arc1_cdb2 ora_arc1_cdb2
2535 2535 ora_arc2_cdb2 ora_arc2_cdb2
2537 2537 ora_arc3_cdb2 ora_arc3_cdb2
2539 2539 ora_tt00_cdb2 ora_tt00_cdb2
2541 2541 ora_smco_cdb2 ora_smco_cdb2
2543 2543 ora_w000_cdb2 ora_w000_cdb2
2545 2545 ora_w001_cdb2 ora_w001_cdb2
2560 2560 ora_aqpc_cdb2 ora_aqpc_cdb2
2564 2564 ora_p000_cdb2 ora_p000_cdb2
```



```

2566 2566 ora_p001_cdb2 ora_p001_cdb2
2568 2568 ora_p002_cdb2 ora_p002_cdb2
2570 2570 ora_p003_cdb2 ora_p003_cdb2
2572 2572 ora_p004_cdb2 ora_p004_cdb2
2574 2574 ora_p005_cdb2 ora_p005_cdb2
2576 2576 ora_p006_cdb2 ora_p006_cdb2
2578 2578 ora_p007_cdb2 ora_p007_cdb2
2580 2580 ora_qm02_cdb2 ora_qm02_cdb2
2584 2584 ora_q002_cdb2 ora_q002_cdb2
2586 2586 ora_q003_cdb2 ora_q003_cdb2
2768 2768 ora_cjq0_cdb2 ora_cjq0_cdb2
2803 2803 ora_p008_cdb2 ora_p008_cdb2
2805 2805 ora_p009_cdb2 ora_p009_cdb2
2809 2809 ora_p00a_cdb2 ora_p00a_cdb2
2811 2811 ora_p00b_cdb2 ora_p00b_cdb2
2813 2813 ora_p00c_cdb2 ora_p00c_cdb2
2815 2815 ora_p00d_cdb2 ora_p00d_cdb2
2817 2817 ora_p00e_cdb2 ora_p00e_cdb2
2819 2819 ora_p00f_cdb2 ora_p00f_cdb2
3625 3625 ora_w002_cdb2 ora_w002_cdb2
3715 3715 ora_w003_cdb2 ora_w003_cdb2
3733 3733 ora_w004_cdb2 ora_w004_cdb2
5053 5053 ora_w005_cdb2 ora_w005_cdb2
11400 11400 oracle_11400_cd oracledb2 (LOCAL=NO)
11689 11689 ora_j000_cdb2 ora_j000_cdb2
11693 11693 ora_j001_cdb2 ora_j001_cdb2
11821 11821 grep grep cdb2
$

```

2. What do you observe?  
Each Oracle process is running into a different OS process.

3. Do the same as in step 1, but using Oracle Database dictionary views. Use the \$HOME/labs/RM/MPMT.sql script.

```
$ sqlplus / as sysdba
```

```
Connected to:
```

```
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -  
64bit Production
```

```
With the Partitioning, OLAP, Advanced Analytics, Real  
Application Testing and Unified Auditing options
```

```
SQL> @MPMT
```

SPID	STID	PNAME	USERNAME	PROGRAM
11832	11832		SYS	sqlplus@<YOUR_SERVER_NAME> (TNS V1-V3)
12119	12119	J000	SYS	oracle@<YOUR_SERVER_NAME> (J000)
12135	12135	J001	SYS	oracle@<YOUR_SERVER_NAME> (J001)
12137	12137	J002	SYS	oracle@<YOUR_SERVER_NAME> (J002)
12139	12139	J003	SYS	oracle@<YOUR_SERVER_NAME> (J003)
12141	12141	J004	SYS	oracle@<YOUR_SERVER_NAME> (J004)
12143	12143	J005	SYS	oracle@<YOUR_SERVER_NAME> (J005)
12145	12145	J006	SYS	oracle@<YOUR_SERVER_NAME> (J006)
12147	12147	J007	SYS	oracle@<YOUR_SERVER_NAME> (J007)
12149	12149	J008	SYS	oracle@<YOUR_SERVER_NAME> (J008)
12151	12151	J009	SYS	oracle@<YOUR_SERVER_NAME> (J009)
12153	12153	J010	SYS	oracle@<YOUR_SERVER_NAME> (J010)
12155	12155	J011	SYS	oracle@<YOUR_SERVER_NAME> (J011)
12157	12157	J012		oracle@<YOUR_SERVER_NAME> (J012)
12161	12161	J013	SYS	oracle@<YOUR_SERVER_NAME> (J013)
12163	12163	J014		oracle@<YOUR_SERVER_NAME> (J014)
12165	12165	J015	SYS	oracle@<YOUR_SERVER_NAME> (J015)
12167	12167	J016		oracle@<YOUR_SERVER_NAME> (J016)
12169	12169	J017	SYS	oracle@<YOUR_SERVER_NAME> (J017)
12171	12171	J018	SYS	oracle@<YOUR_SERVER_NAME> (J018)
12173	12173	J019	SYS	oracle@<YOUR_SERVER_NAME> (J019)
12175	12175	J020	SYS	oracle@<YOUR_SERVER_NAME> (J020)
12195	12195	J021	SYS	oracle@<YOUR_SERVER_NAME> (J021)
12197	12197	J022	SYS	oracle@<YOUR_SERVER_NAME> (J022)
12205	12205	J023		oracle@<YOUR_SERVER_NAME> (J023)
12207	12207	J024	SYS	oracle@<YOUR_SERVER_NAME> (J024)
12211	12211	J025	SYS	oracle@<YOUR_SERVER_NAME> (J025)
12213	12213	J026	SYS	oracle@<YOUR_SERVER_NAME> (J026)
12215	12215	J027	SYS	oracle@<YOUR_SERVER_NAME> (J027)

12219	12219	J028	oracle@<YOUR_SERVER_NAME>	(J028)
12221	12221	M000	oracle@<YOUR_SERVER_NAME>	(M000)
2463	2463	PMON	oracle@<YOUR_SERVER_NAME>	(PMON)
2465	2465	PSP0	oracle@<YOUR_SERVER_NAME>	(PSP0)
2467	2467	VKTM	oracle@<YOUR_SERVER_NAME>	(VKTM)
2471	2471	GEN0	oracle@<YOUR_SERVER_NAME>	(GEN0)
2473	2473	MMAN	oracle@<YOUR_SERVER_NAME>	(MMAN)
2477	2477	DIAG	oracle@<YOUR_SERVER_NAME>	(DIAG)
2479	2479	DBRM	oracle@<YOUR_SERVER_NAME>	(DBRM)
2481	2481	VKRM	oracle@<YOUR_SERVER_NAME>	(VKRM)
2483	2483	DIA0	oracle@<YOUR_SERVER_NAME>	(DIA0)
2485	2485	DBW0	oracle@<YOUR_SERVER_NAME>	(DBW0)
2487	2487	LGWR	oracle@<YOUR_SERVER_NAME>	(LGWR)
2489	2489	CKPT	oracle@<YOUR_SERVER_NAME>	(CKPT)
2491	2491	LG00	oracle@<YOUR_SERVER_NAME>	(LG00)
2493	2493	SMON	oracle@<YOUR_SERVER_NAME>	(SMON)
2495	2495	LG01	oracle@<YOUR_SERVER_NAME>	(LG01)
2497	2497	RECO	oracle@<YOUR_SERVER_NAME>	(RECO)
2499	2499	LREG	oracle@<YOUR_SERVER_NAME>	(LREG)
2501	2501	PXMN	oracle@<YOUR_SERVER_NAME>	(PXMN)
2503	2503	MMON	oracle@<YOUR_SERVER_NAME>	(MMON)
2505	2505	MMNL	oracle@<YOUR_SERVER_NAME>	(MMNL)
2526	2526	RVWR	oracle@<YOUR_SERVER_NAME>	(RVWR)
2529	2529	TMON	oracle@<YOUR_SERVER_NAME>	(TMON)
2531	2531	ARC0	oracle@<YOUR_SERVER_NAME>	(ARC0)
2533	2533	ARC1	oracle@<YOUR_SERVER_NAME>	(ARC1)
2535	2535	ARC2	oracle@<YOUR_SERVER_NAME>	(ARC2)
2537	2537	ARC3	oracle@<YOUR_SERVER_NAME>	(ARC3)
2539	2539	TT00	oracle@<YOUR_SERVER_NAME>	(TT00)
2541	2541	SMCO	oracle@<YOUR_SERVER_NAME>	(SMCO)
2543	2543	W000	oracle@<YOUR_SERVER_NAME>	(W000)
2545	2545	W001	oracle@<YOUR_SERVER_NAME>	(W001)
2560	2560	AQPC	oracle@<YOUR_SERVER_NAME>	(AQPC)
2580	2580	QM02	oracle@<YOUR_SERVER_NAME>	(QM02)
2584	2584	Q002	oracle@<YOUR_SERVER_NAME>	(Q002)
2586	2586	Q003	oracle@<YOUR_SERVER_NAME>	(Q003)
2768	2768	CJQ0	oracle@<YOUR_SERVER_NAME>	(CJQ0)
3625	3625	W002	oracle@<YOUR_SERVER_NAME>	(W002)
3715	3715	W003	oracle@<YOUR_SERVER_NAME>	(W003)
3733	3733	W004	oracle@<YOUR_SERVER_NAME>	(W004)
5053	5053	W005	oracle@<YOUR_SERVER_NAME>	(W005)

```
70 rows selected.
```

```
SQL>
```

4. Modify your SPFILE to prepare for MPMT architecture.

```
SQL> alter system set threaded_execution=true scope=spfile;
```

```
System altered.
```

```
SQL>
```

5. Still connected from the same session, shut down your cdb2 instance, and restart it again.

```
SQL> shutdown immediate
```

```
Database closed.
```

```
Database dismounted.
```

```
ORACLE instance shut down.
```

```
SQL>
```

```
SQL> startup
```

```
ORA-01017: invalid username/password; logon denied
```

```
SQL>
```

6. Why are you getting the “ORA-01017: invalid username/password; logon denied” error?

The reason is that when using MPMT architecture, you have to use password authentication for SYSDBA operations.

7. Start up your `cdb2` instance.

```
SQL> connect sys as sysdba
Enter password: *****
Connected.
SQL> startup
ORA-01081: cannot start already-running ORACLE - shut it down
first
SQL> alter database mount;

Database altered.

SQL> alter database open;

Database altered.

SQL>
```

8. Still connected from your SQL\*Plus session, list the OS processes and OS threads used to run your `cdb2` instance using OS commands.

```
SQL> ! ps -eLo "pid tid comm args" |grep cdb2
12372 12372 ora_pmon_cdb2    ora_pmon_cdb2
12374 12374 ora_psp0_cdb2    ora_psp0_cdb2
12376 12376 ora_vktm_cdb2    ora_vktm_cdb2
12380 12380 ora_scmn_cdb2    ora_u004_cdb2
12380 12381 oracle         ora_u004_cdb2
12380 12382 ora_gen0_cdb2    ora_u004_cdb2
12380 12383 ora_mman_cdb2    ora_u004_cdb2
12380 12389 ora_dbrm_cdb2    ora_u004_cdb2
12380 12394 ora_lgwr_cdb2    ora_u004_cdb2
12380 12395 ora_ckpt_cdb2    ora_u004_cdb2
12380 12396 ora_smon_cdb2    ora_u004_cdb2
12380 12398 ora_lreg_cdb2    ora_u004_cdb2
12380 17754 ora_rvwr_cdb2    ora_u004_cdb2
12386 12386 ora_scmn_cdb2    ora_u005_cdb2
12386 12387 oracle         ora_u005_cdb2
12386 12388 ora_diag_cdb2    ora_u005_cdb2
12386 12390 ora_vkrm_cdb2    ora_u005_cdb2
12386 12391 ora_dia0_cdb2    ora_u005_cdb2
12386 12397 ora_reco_cdb2    ora_u005_cdb2
12386 12399 ora_pxmnl_cdb2    ora_u005_cdb2
12386 12400 ora_mmon_cdb2    ora_u005_cdb2
12386 12401 ora_mmln_cdb2    ora_u005_cdb2
12386 12402 ora_d000_cdb2    ora_u005_cdb2
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

12386	12403	ora_s000_cdb2	ora_u005_cdb2
12386	12404	ora_n000_cdb2	ora_u005_cdb2
12386	17649	oracle_17649_cd	ora_u005_cdb2
12386	17755	ora_tmon_cdb2	ora_u005_cdb2
12386	17756	ora_arc0_cdb2	ora_u005_cdb2
12386	17757	ora_arc1_cdb2	ora_u005_cdb2
12386	17758	ora_arc2_cdb2	ora_u005_cdb2
12386	17759	ora_arc3_cdb2	ora_u005_cdb2
12386	17760	ora_tt00_cdb2	ora_u005_cdb2
12386	17765	ora_smco_cdb2	ora_u005_cdb2
12386	17766	ora_w000_cdb2	ora_u005_cdb2
12386	17767	ora_w001_cdb2	ora_u005_cdb2
12386	17769	ora_aqpc_cdb2	ora_u005_cdb2
12386	17776	ora_p000_cdb2	ora_u005_cdb2
12386	17777	ora_p001_cdb2	ora_u005_cdb2
12386	17778	ora_p002_cdb2	ora_u005_cdb2
12386	17779	ora_p003_cdb2	ora_u005_cdb2
12386	17970	ora_qm02_cdb2	ora_u005_cdb2
12386	17972	ora_q002_cdb2	ora_u005_cdb2
12386	17973	ora_q003_cdb2	ora_u005_cdb2
12386	17974	ora_q004_cdb2	ora_u005_cdb2
12386	17975	ora_p004_cdb2	ora_u005_cdb2
12386	17976	ora_p005_cdb2	ora_u005_cdb2
12386	18021	ora_p006_cdb2	ora_u005_cdb2
12386	18022	ora_p007_cdb2	ora_u005_cdb2
12386	18057	ora_cjq0_cdb2	ora_u005_cdb2
12386	18065	ora_p008_cdb2	ora_u005_cdb2
12386	18066	ora_p009_cdb2	ora_u005_cdb2
12386	18067	ora_p00a_cdb2	ora_u005_cdb2
12386	18068	ora_p00b_cdb2	ora_u005_cdb2
12386	18070	ora_p00c_cdb2	ora_u005_cdb2
12386	18071	ora_p00d_cdb2	ora_u005_cdb2
12386	18072	ora_p00e_cdb2	ora_u005_cdb2
12386	18073	ora_p00f_cdb2	ora_u005_cdb2
12386	18118	ora_p00g_cdb2	ora_u005_cdb2
12386	18119	ora_p00h_cdb2	ora_u005_cdb2
12386	18120	ora_p00i_cdb2	ora_u005_cdb2
12386	18121	ora_p00j_cdb2	ora_u005_cdb2
12386	18122	ora_p00k_cdb2	ora_u005_cdb2
12386	18123	ora_p00l_cdb2	ora_u005_cdb2
12386	18124	ora_p00m_cdb2	ora_u005_cdb2
12386	18125	ora_p00n_cdb2	ora_u005_cdb2

```

12386 18126 ora_j000_cdb2      ora_u005_cdb2
12386 18127 ora_j001_cdb2      ora_u005_cdb2
12386 18128 ora_j002_cdb2      ora_u005_cdb2
12386 18130 ora_j004_cdb2      ora_u005_cdb2
12386 18131 ora_j005_cdb2      ora_u005_cdb2
12386 18132 ora_j006_cdb2      ora_u005_cdb2
12386 18133 ora_j007_cdb2      ora_u005_cdb2
12386 18134 ora_j008_cdb2      ora_u005_cdb2
12386 18135 ora_j009_cdb2      ora_u005_cdb2
12386 18136 ora_j010_cdb2      ora_u005_cdb2
12386 18137 ora_j011_cdb2      ora_u005_cdb2
12386 18138 ora_j012_cdb2      ora_u005_cdb2
12386 18139 ora_j013_cdb2      ora_u005_cdb2
12386 18140 ora_j014_cdb2      ora_u005_cdb2
12386 18141 ora_j015_cdb2      ora_u005_cdb2
12386 18142 ora_j016_cdb2      ora_u005_cdb2
12386 18143 ora_j017_cdb2      ora_u005_cdb2
12386 18144 ora_j018_cdb2      ora_u005_cdb2
12386 18145 ora_j019_cdb2      ora_u005_cdb2
12386 18146 ora_j020_cdb2      ora_u005_cdb2
12386 18147 ora_j021_cdb2      ora_u005_cdb2
12386 18148 ora_j022_cdb2      ora_u005_cdb2
12386 18149 ora_j023_cdb2      ora_u005_cdb2
12386 18150 ora_j024_cdb2      ora_u005_cdb2
12386 18151 ora_j025_cdb2      ora_u005_cdb2
12386 18152 ora_j026_cdb2      ora_u005_cdb2
12386 18153 ora_j027_cdb2      ora_u005_cdb2
12386 18154 ora_j028_cdb2      ora_u005_cdb2
12386 18155 ora_j029_cdb2      ora_u005_cdb2
12386 18156 ora_j030_cdb2      ora_u005_cdb2
12386 18157 ora_j031_cdb2      ora_u005_cdb2
12386 18158 ora_j032_cdb2      ora_u005_cdb2
12386 18159 ora_j033_cdb2      ora_u005_cdb2
12386 18160 ora_j034_cdb2      ora_u005_cdb2
12386 18161 ora_j035_cdb2      ora_u005_cdb2
12386 18162 ora_j036_cdb2      ora_u005_cdb2
12393 12393 ora_dbw0_cdb2      ora_dbw0_cdb2
18269 18269 bash              /bin/bash -c ps -eLo "pid tid comm
args" |grep cdb2
18271 18271 grep               grep cdb2

SQL>

```

```
SQL> ! pgrep -lf cdb2
12372 ora_pmon_cdb2
12374 ora_psp0_cdb2
12376 ora_vktm_cdb2
12380 ora_u004_cdb2
12386 ora_u005_cdb2
12393 ora_dbw0_cdb2

SQL>
```

9. Do the same using the \$HOME/labs/RM/MPMT.sql script.

```
SQL> @MPMT

SPID   STID   PNAME USERNAME   PROGRAM
-----
12372  12372 PMON          oracle@EDRSR41P1 (PMON)
12374  12374 PSP0          oracle@EDRSR41P1 (PSP0)
12376  12376 VKTM          oracle@EDRSR41P1 (VKTM)
12380  12395 CKPT          oracle@EDRSR41P1 (CKPT)
12380  12394 LGWR          oracle@EDRSR41P1 (LGWR)
12380  17754 RVWR          oracle@EDRSR41P1 (RVWR)
12380  12389 DBRM          oracle@EDRSR41P1 (DBRM)
12380  12382 GEN0          oracle@EDRSR41P1 (GEN0)
12380  12380 SCMN          oracle@EDRSR41P1 (SCMN)
12380  12383 MMAN          oracle@EDRSR41P1 (MMAN)
12380  12398 LREG          oracle@EDRSR41P1 (LREG)
12380  12396 SMON          oracle@EDRSR41P1 (SMON)
12386  12401 MMNL          oracle@EDRSR41P1 (MMNL)
12386  12400 MMON          oracle@EDRSR41P1 (MMON)
12386  18057 CJQ0          oracle@EDRSR41P1 (CJQ0)
12386  12397 RECO          oracle@EDRSR41P1 (RECO)
12386  12391 DIA0          oracle@EDRSR41P1 (DIA0)
12386  12390 VKRM          oracle@EDRSR41P1 (VKRM)
12386  12386 SCMN          oracle@EDRSR41P1 (SCMN)
12386  12388 DIAG          oracle@EDRSR41P1 (DIAG)
12386  17649          SYS          sqlplus@EDRSR41P1 (TNS V1-V3)
12386  12399 PXMN          oracle@EDRSR41P1 (PXMN)
12386  17755 TMON          oracle@EDRSR41P1 (TMON)
12386  17756 ARC0          oracle@EDRSR41P1 (ARC0)
12386  17757 ARC1          oracle@EDRSR41P1 (ARC1)
12386  17758 ARC2          oracle@EDRSR41P1 (ARC2)
12386  17759 ARC3          oracle@EDRSR41P1 (ARC3)
```



```

12386 17760 TT00          oracle@EDRSR41P1 (TT00)
12386 17765 SMCO          oracle@EDRSR41P1 (SMCO)
12386 17766 W000          oracle@EDRSR41P1 (W000)
12386 17767 W001          oracle@EDRSR41P1 (W001)
12386 17769 AQPC          oracle@EDRSR41P1 (AQPC)
12386 17970 QM02          oracle@EDRSR41P1 (QM02)
12386 17972 Q002          oracle@EDRSR41P1 (Q002)
12386 17973 Q003          oracle@EDRSR41P1 (Q003)
12386 17974 Q004          oracle@EDRSR41P1 (Q004)
12393 12393 DBW0          oracle@EDRSR41P1 (DBW0)

```

```
37 rows selected.
```

```
SQL>
```

10. What do you observe?  
Now, many Oracle processes run as threads inside a small amount of OS processes.
11. Establish a remote connection to your `cds2` instance using SQL\*Plus, and list again OS processes and threads used to run all Oracle processes.

```
SQL> connect sys@cds2 as sysdba
```

```
Enter password: *****
```

```
Connected.
```

```
SQL> @MPMT
```

SPID	STID	PNAME	USERNAME	PROGRAM
12372	12372	PMON		oracle@EDRSR41P1 (PMON)
12374	12374	PSP0		oracle@EDRSR41P1 (PSP0)
12376	12376	VKTM		oracle@EDRSR41P1 (VKTM)
12380	12394	LGWR		oracle@EDRSR41P1 (LGWR)
12380	12398	LREG		oracle@EDRSR41P1 (LREG)
12380	12382	GEN0		oracle@EDRSR41P1 (GEN0)
12380	17754	RVWR		oracle@EDRSR41P1 (RVWR)
12380	12389	DBRM		oracle@EDRSR41P1 (DBRM)
12380	12395	CKPT		oracle@EDRSR41P1 (CKPT)
12380	12396	SMON		oracle@EDRSR41P1 (SMON)
12380	12380	SCMN		oracle@EDRSR41P1 (SCMN)
12380	12383	MMAN		oracle@EDRSR41P1 (MMAN)
12386	12397	RECO		oracle@EDRSR41P1 (RECO)
12386	12391	DIA0		oracle@EDRSR41P1 (DIA0)
12386	12390	VKRM		oracle@EDRSR41P1 (VKRM)
12386	12386	SCMN		oracle@EDRSR41P1 (SCMN)
12386	12388	DIAG		oracle@EDRSR41P1 (DIAG)

```

12386 12399 PXMN          oracle@EDRSR41P1 (PXMN)
12386 18057 CJQ0          oracle@EDRSR41P1 (CJQ0)
12386 18721 W002          oracle@EDRSR41P1 (W002)
12386 12400 MMON          oracle@EDRSR41P1 (MMON)
12386 12401 MMNL          oracle@EDRSR41P1 (MMNL)
12386 17755 TMON          oracle@EDRSR41P1 (TMON)
12386 17756 ARC0          oracle@EDRSR41P1 (ARC0)
12386 17757 ARC1          oracle@EDRSR41P1 (ARC1)
12386 17758 ARC2          oracle@EDRSR41P1 (ARC2)
12386 17759 ARC3          oracle@EDRSR41P1 (ARC3)
12386 17760 TT00          oracle@EDRSR41P1 (TT00)
12386 17765 SMCO          oracle@EDRSR41P1 (SMCO)
12386 17766 W000          oracle@EDRSR41P1 (W000)
12386 17767 W001          oracle@EDRSR41P1 (W001)
12386 17769 AQPC          oracle@EDRSR41P1 (AQPC)
12386 17970 QM02          oracle@EDRSR41P1 (QM02)
12386 17972 Q002          oracle@EDRSR41P1 (Q002)
12386 17973 Q003          oracle@EDRSR41P1 (Q003)
12386 17974 Q004          oracle@EDRSR41P1 (Q004)
12393 12393 DBW0          oracle@EDRSR41P1 (DBW0)
18760 18760          SYS      sqlplus@EDRSR41P1 (TNS V1-V3)

38 rows selected.

SQL>

```

12. What do you observe?  
Your foreground process that runs your SQL\*Plus connection is using one OS process, and not threads.
13. How would you make sure foreground processes are using threads?

```

SQL> EXIT
$

```

- a. View the tnsnames.ora file content.

```

$ cat $ORACLE_HOME/network/admin/tnsnames.ora
# tnsnames.ora Network Configuration File:
/u01/app/oracle/product/12.1.0/dbhome_1/network/admin/tnsnames.o
ra
# Generated by Oracle configuration tools.

LISTENER_CDB2 =

```

```

    (ADDRESS = (PROTOCOL = TCP) (HOST = <your_hostname>) (PORT =
1521))

PDB_ORCL2 =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521))
        )
        (CONNECT_DATA =
            (SERVICE_NAME = pdb_orcl2)
        )
    )

PDB2_2 =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521))
        )
        (CONNECT_DATA =
            (SERVICE_NAME = pdb2_2)
        )
    )

PDB2_1 =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521))
        )
        (CONNECT_DATA =
            (SERVICE_NAME = pdb2_1)
        )
    )

PDB2 =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521))
        )
        (CONNECT_DATA =

```

```

        (SERVICE_NAME = pdb2)
    )
)

CDB2 =
    (DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = <your_hostname>) (PORT =
1521))
        (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = cdb2)
        )
    )

CDB1 =
    (DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = <your_hostname>) (PORT =
1521))
        (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = cdb1)
        )
    )

ORCL2 =
    (DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = <your_hostname>) (PORT =
1521))
        (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = orcl2)
        )
    )

PDB1_1 =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521))
        )
        (CONNECT_DATA =
            (SERVICE_NAME = pdb1_1)
        )
    )

```

```

)

EM12REP =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = <your_hostname>) (PORT =
1521))
    (CONNECT_DATA =
      (SERVICE_NAME = em12rep)
    )
  )
)
ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = <your_hostname>) (PORT =
1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl)
    )
  )
)
$

```

- b. View the listener.ora file content.

```

$ cat $ORACLE_HOME/network/admin/listener.ora
# listener.ora Network Configuration File:
/u01/app/oracle/product/12.1.0/dbhome_1/network/admin/listener.o
ra
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC1521))
      (ADDRESS = (PROTOCOL = TCP) (HOST = <YOURSERVER> ) (PORT =
1521))
    )
  )
)
$

```

- c. Keep a copy of the listener.ora file.

```

$ cp $ORACLE_HOME/network/admin/listener.ora
$ORACLE_HOME/network/admin/listener.ora.bak
$

```

- d. Add the following parameter in the `listener.ora` file.

```
$ echo DEDICATED_THROUGH_BROKER_LISTENER=on >>
$ORACLE_HOME/network/admin/listener.ora
$
```

- e. Check that the `listener.ora` file is adequately modified.

```
$ cat $ORACLE_HOME/network/admin/listener.ora
# listener.ora Network Configuration File:
/u01/app/oracle/product/12.1.0/dbhome_1/network/admin/listener.o
ra
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC1521))
      (ADDRESS = (PROTOCOL = TCP) (HOST = <YOURSERVER>) (PORT =
1521))
    )
  )

DEDICATED_THROUGH_BROKER_LISTENER=on
$
```

- f. Restart the listener.

```
$ lsnrctl stop

LSNRCTL for Linux: Version 12.1.0.2.0 - Production on 22-MAY-
2014 08:35:55
Copyright (c) 1991, 2014, Oracle. All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) ( HOST =
<YOURSERVER>) (PORT=1521)))
The command completed successfully
$
```

```
$ lsnrctl start

LSNRCTL for Linux: Version 12.1.0.2.0 - Production on 22-MAY-
2014 08:35:59
Copyright (c) 1991, 2014, Oracle. All rights reserved.

...
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=<YOURSERVER>) (PORT=152
1)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc) (KEY=EXTPROC1521)))
```

```
The listener supports no services
The command completed successfully
$
```

- g. Set the LOCAL\_LISTENER parameter to cdb2.

```
$ sqlplus sys@cdb2 as sysdba

Enter password: *****

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> show parameter local_listener

NAME                                TYPE                                VALUE
-----                                -                                -
local_listener                      string
LISTENER_CDB2

SQL> alter system set local_listener=cdb2 scope=both;

System altered.

SQL> show parameter local_listener

NAME                                TYPE                                VALUE
-----                                -                                -
local_listener                      string                                CDB2

SQL>
```

14. Check that what you did is working.

**Tip:** Looking at the sqlplus program, it should be run as a thread in an existing OS process.

```
SQL> connect sys@cdb2 as sysdba
Enter password: *****
Connected.
SQL> @MPMT

SPID  STID  PNAME USERNAME  PROGRAM
-----
```

```

12372 12372 PMON                oracle@EDRSR41P1 (PMON)
12374 12374 PSP0                oracle@EDRSR41P1 (PSP0)
12376 12376 VKTM                oracle@EDRSR41P1 (VKTM)
12380 12395 CKPT                oracle@EDRSR41P1 (CKPT)
12380 12394 LGWR                oracle@EDRSR41P1 (LGWR)
12380 17754 RVWR                oracle@EDRSR41P1 (RVWR)
12380 12389 DBRM                oracle@EDRSR41P1 (DBRM)
12380 12382 GEN0                oracle@EDRSR41P1 (GEN0)
12380 12380 SCMN                oracle@EDRSR41P1 (SCMN)
12380 12383 MMAN                oracle@EDRSR41P1 (MMAN)
12380 12398 LREG                oracle@EDRSR41P1 (LREG)
12380 12396 SMON                oracle@EDRSR41P1 (SMON)
12386 12401 MMNL                oracle@EDRSR41P1 (MMNL)
12386 12400 MMON                oracle@EDRSR41P1 (MMON)
12386 18721 W002                oracle@EDRSR41P1 (W002)
12386 12397 RECO                oracle@EDRSR41P1 (RECO)
12386 12391 DIA0                oracle@EDRSR41P1 (DIA0)
12386 12390 VKRM                oracle@EDRSR41P1 (VKRM)
12386 12386 SCMN                oracle@EDRSR41P1 (SCMN)
12386 12388 DIAG                oracle@EDRSR41P1 (DIAG)
12386 19428 SYS                sqlplus@EDRSR41P1 (TNS V1-V3)
12386 12399 PXMN                oracle@EDRSR41P1 (PXMN)
12386 17755 TMON                oracle@EDRSR41P1 (TMON)
12386 17756 ARC0                oracle@EDRSR41P1 (ARC0)
12386 17757 ARC1                oracle@EDRSR41P1 (ARC1)
12386 17758 ARC2                oracle@EDRSR41P1 (ARC2)
12386 17759 ARC3                oracle@EDRSR41P1 (ARC3)
12386 17760 TT00                oracle@EDRSR41P1 (TT00)
12386 17765 SMCO                oracle@EDRSR41P1 (SMCO)
12386 17766 W000                oracle@EDRSR41P1 (W000)
12386 17767 W001                oracle@EDRSR41P1 (W001)
12386 17769 AQPC                oracle@EDRSR41P1 (AQPC)
12386 17970 QM02                oracle@EDRSR41P1 (QM02)
12386 17972 Q002                oracle@EDRSR41P1 (Q002)
12386 17973 Q003                oracle@EDRSR41P1 (Q003)
12386 18057 CJQ0                oracle@EDRSR41P1 (CJQ0)
12393 12393 DBW0                oracle@EDRSR41P1 (DBW0)

37 rows selected.

SQL>

```



15. Revert to the non-MPMT architecture. Keep the `cdb2` database closed for the moment so as to preserve resources on the machine.

```
SQL> alter system set threaded_execution=false scope=spfile;

System altered.

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
ERROR:
ORA-12514: TNS:listener does not currently know of service
requested in connect
descriptor

Warning: You are no longer connected to ORACLE.
SQL> EXIT
$
```

16. Restore the original `listener.ora` file.

```
$ cp $ORACLE_HOME/network/admin/listener.ora.bak
$ORACLE_HOME/network/admin/listener.ora
$
```



# **Practices for Lesson 21: Tables, Indexes and Online Operations**

## **Chapter 21**

## Practices for Lesson 21: Overview

---

### Practices Overview

In the practice for this lesson, you will use the invisible/visible table columns and the Advanced Row Compression.

## Practice 21-1: Using Invisible Table Columns

---

### Overview

In this practice, you will create a table with invisible columns. These columns are not necessarily useful for the current application but might become useful in a later application release.

### Tasks

1. Make sure you are in the `~/labs/Tables` directory and your environment points to the `orcl` instance.

```
$ cd ~/labs/Tables
$ . oraenv
ORACLE_SID = [cdb2] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

2. Run the `invisible_setup.sql` script to create a new user `STATS`

```
$ sqlplus / AS SYSDBA
```

```

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

```

```
SQL> @invisible_setup.sql
```

```
Connected.
```

```
DROP USER stats CASCADE
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01918: user 'STATS' does not exist
```

```
User created.
```

```
Grant succeeded.
```

```
SQL>
```

3. The STATS user creates a new table CENSUS. The table structure contains three columns, GENDER, COUNTRY, NUMBER and an invisible column REGION. The REGION column is not used by the application yet, but might become useful in a future application release.

```
SQL> CREATE TABLE stats.census (gender VARCHAR2(10), country
CHAR(2), nb NUMBER, region VARCHAR2(20) INVISIBLE );
```

```
Table created.
```

```
SQL>
```

4. Describe the structure of the CENSUS table.
- a. You see that the invisible column does not appear.

```
SQL> DESC stats.census
```

Name	Null?	Type
-----		
GENDER		VARCHAR2(10)
COUNTRY		CHAR(2)
NB		NUMBER

```
SQL>
```

- b. Describe the structure of the CENSUS table so that the invisible column appears.

```
SQL> SET COLINVISIBLE ON
SQL> DESC stats.census
  Name                               Null?      Type
  -----
  GENDER                             VARCHAR2(10)
  COUNTRY                             CHAR(2)
  NB                                  NUMBER
  REGION (INVISIBLE)                 VARCHAR2(20)

SQL>
```

5. Insert rows into the CENSUS table.

- a. You cannot insert a value for the invisible column unless you define it in the projection list.

```
SQL> INSERT INTO stats.census VALUES ('BOY','BR', 10000,
'BAHIA');
INSERT INTO stats.census VALUES ('BOY','BR', 10000, 'BAHIA')
          *
ERROR at line 1:
ORA-00913: too many values

SQL>
```

- b. Insert a row with values for the three visible columns.

```
SQL> INSERT INTO stats.census VALUES ('BOY','BR',100000);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM stats.census;

 GENDER      CO      NB
-----
BOY         BR      100000

SQL>
```

- c. Insert a row with a value for the invisible column.

```
SQL> INSERT INTO stats.census (gender, country, nb, region)
VALUES ('BOY','BR', 35000,'BAHIA');
```

1 row created.

```
SQL> COMMIT;
```

Commit complete.

```
SQL> SELECT gender, country, nb, region FROM stats.census;
```

GENDER	CO	NB	REGION
BOY	BR	100000	
BOY	BR	35000	BAHIA

SQL>

6. Make the invisible column visible.

```
SQL> ALTER TABLE stats.census MODIFY (region VISIBLE);
```

Table altered.

```
SQL> SELECT * FROM stats.census;
```

GENDER	CO	NB	REGION
BOY	BR	100000	
BOY	BR	35000	BAHIA

SQL>



## Practice 21-2: Using Advanced Row Compression

---

### Overview

In this practice, you will use the row store advanced compression. You will compare the storage requirements between a compressed table and an uncompressed table and verify the compression ratio between different tables with the same number of rows.

### Tasks

1. Connect as SH. You first create two copies of the SH.SALES table, the first being compressed and the second being uncompressed.

```
SQL> CONNECT sh
Enter password: *****
Connected.
SQL> drop table sales_nocompress purge;
drop table sales_nocompress purge
      *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> drop table sales_advcompress purge;
drop table sales_advcompress purge
      *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> set echo on
SQL> set timing on
SQL> CREATE TABLE sales_nocompress AS SELECT * FROM sales;

Table created.

Elapsed: 00:00:17.07
SQL> CREATE TABLE sales_advcompress ROW STORE COMPRESS ADVANCED
AS SELECT * FROM sales where 1=0;

Table created.

Elapsed: 00:00:00.12
SQL>

SQL> SELECT count(*) FROM sales_nocompress;

COUNT (*)
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```

-----
      1839684

Elapsed: 00:00:00.67
SQL> SELECT count(*) FROM sales_advcompress;

      COUNT(*)
-----
              0

Elapsed: 00:00:00.00
SQL>

```

2. Execute the \$HOME/labs/Tables/load.sql to load the SALES\_ADVCOMPRESS table.

```

SQL> @load
SQL> declare
  2  commit_after integer := 0 ;
  3  loop_variable integer ;
  4  cursor c_sales is
  5  select prod_id, cust_id, time_id, channel_id, promo_id,
quantity_sold, amount_sold
  6  from sales ;
  7  begin
  8  for r_sales in c_sales
  9  loop
10  if commit_after = 0
11  then
12  loop_variable := 0 ;
13  commit_after := round(dbms_random.value(1,1)) ;
14  end if ;
15  insert into sales_advcompress
16  (prod_id, cust_id, time_id, channel_id, promo_id,
quantity_sold, amount_sold)
17  values
18  (r_sales.prod_id, r_sales.cust_id, r_sales.time_id,
r_sales.channel_id,
19  r_sales.promo_id, r_sales.quantity_sold,
r_sales.amount_sold) ;
20  if loop_variable = commit_after
21  then
22  commit ;
23  commit_after := 0 ;
24  end if ;
25  loop_variable := loop_variable + 1 ;

```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```

26     end loop ;
27     end ;
28 /

```

PL/SQL procedure successfully completed.

Elapsed: 00:02:17.91

SQL>

3. Verify the number of rows in the two tables.

```
SQL> SELECT count(*) FROM sales_nocompress;
```

```

COUNT(*)
-----

```

```
1839684
```

Elapsed: 00:00:00.04

```
SQL> SELECT count(*) FROM sales_advcompress;
```

```

COUNT(*)
-----

```

```
1839684
```

Elapsed: 00:00:00.02

SQL>

4. Now you can compare the storage requirements between the two tables you just created. Use the \$HOME/labs/Tables/comp1.sql script.

```
SQL> @comp1
```

```
SQL> COL segment_name FORMAT A30
```

```
SQL> select segment_name, sum(bytes)/1024/1024 mb
```

```
2         from      dba_segments
```

```
3         where  owner = 'SH' and segment_name in
```

```
4             ('SALES_NOCOMPRESS', 'SALES_ADVCOMPRESS')
```

```
5         group by segment_name order by segment_name;
```

```

SEGMENT_NAME                                MB
-----

```

```
SALES_ADVCOMPRESS                           28
```

```
SALES_NOCOMPRESS                            72
```

Elapsed: 00:00:00.12

SQL>

5. Use the DBMS\_COMPRESSION package to get the compression ratio of the SALES\_ADVCOMPRESS table. Execute the \$HOME/labs/Tables/ratio.sql script.

```
SQL> @ratio
SQL> set serveroutput on
SQL> DECLARE
    blkcnt_cmp pls_integer;
    blkcnt_uncmp pls_integer;
    row_cmp pls_integer;
    row_uncmp pls_integer;
    cmp_ratio pls_integer;
    comptype_str varchar2(100);
BEGIN
    DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
        'USERS', 'SH', 'SALES_ADVCOMPRESS', NULL,
    DBMS_COMPRESSION.COMP_ADVANCED,
        blkcnt_cmp, blkcnt_uncmp, row_cmp, row_uncmp, cmp_ratio,
    comptype_str,1000,1);
    DBMS_OUTPUT.PUT_LINE('Table = SH.SALES_ADVCOMPRESS');
    DBMS_OUTPUT.PUT_LINE('Block count compressed = ' ||
    blkcnt_cmp);
    DBMS_OUTPUT.PUT_LINE('Block count uncompressed = ' ||
    blkcnt_uncmp);
    DBMS_OUTPUT.PUT_LINE('Row count per block compressed = ' ||
    row_cmp);
    DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed = ' ||
    row_uncmp);
    DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
    DBMS_OUTPUT.PUT_LINE('Compression ratio =
    ' || blkcnt_uncmp/blkcnt_cmp || ' to 1');
    DBMS_OUTPUT.PUT_LINE('Compression ratio org= ' || cmp_ratio);
END;
/
Table = SH.SALES_ADVCOMPRESS
Block count compressed = 2105
Block count uncompressed = 5646
Row count per block compressed = 556
Row count per block uncompressed = 207
Compression type = "Compress Advanced"
Compression ratio = 2.68218527315914489311163895486935866983 to
1
Compression ratio org= 3

PL/SQL procedure successfully completed.
```

```
Elapsed: 00:00:16.79
SQL>
```

6. Analyze the tables and note the number of rows compressed in the compressed table and compare the ratio of compression with the non-compressed table.

```
SQL> ANALYZE TABLE sh.sales_nocompress COMPUTE STATISTICS;

Table analyzed.

Elapsed: 00:00:16.79
SQL> ANALYZE TABLE sh.sales_advcompress COMPUTE STATISTICS;

Table analyzed.

Elapsed: 00:00:11.29
SQL>
```

To perform the comparison, execute the \$HOME/labs/Tables/comp2.sql script.

```
SQL> @comp2
SQL> COL object_name format A18
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT object_name , AVGROWSIZE_NC, AVGROWSIZE_C, NBLK_NC,
2          NBLK_ADVANCED, NROWS_NC, NROWS_ADVANCED
3          FROM      sys.compression_stat$ c , DBA_OBJECTS o
4          WHERE  c.obj# = o.object_id
5          AND  o.object_name = 'SALES_ADVCOMPRESS';

OBJECT_NAME          AVGROWSIZE_NC AVGROWSIZE_C    NBLK_NC
-----
NBLK_ADVANCED      NROWS_NC NROWS_ADVANCED
-----
SALES_ADVCOMPRESS          32.81      10.54          22
          3498          71328      1768356

Elapsed: 00:00:00.21
SQL> SELECT avg_row_len, num_rows FROM dba_tables WHERE
table_name='SALES_NOCOMPRESS';

AVG_ROW_LEN    NUM_ROWS
-----
          33      1839684

Elapsed: 00:00:00.05
SQL>
```

The non compressed table contains 1839684 uncompressed rows whereas the compressed table contains 1768356 compressed rows and only 71328 uncompressed rows with an average row size 3 times smaller.

7. Clean up the tables.

```
SQL> DROP TABLE sh.sales_nocompress PURGE;

Table dropped.

Elapsed: 00:00:00.07
SQL> DROP TABLE sh.sales_advcompress PURGE;

Table dropped.

Elapsed: 00:00:00.25

SQL> EXIT
$
```

## **Practices for Lesson 22: Oracle Data Pump, SQL\*Loader, and External Tables**

### **Chapter 22**

## Practices for Lesson 22: Overview

---

### Practices Overview

In these practices, you will perform a FULL TRANSPORTABLE from the non-CDB into the non-CDB `orcl2`.

You will also perform a data load by using SQL\*Loader Express Mode.



## Practice 22-1: Exporting/Importing Databases in FULL TRANSPORTABLE Mode

---

### Overview

In this practice, you export the `orcl` non-CDB database and import into another non-CDB database `orcl2` using the Oracle Data Pump FULL TRANSPORTABLE feature.

### Assumptions

The `orcl` and `orcl2` non-CDBs already exist. They are pre-created during the course setup.

### Tasks

1. Make sure you are in the `~/labs/Load` directory.

```
$ cd ~/labs/Load
$
```

2. The `orcl2` non-CDB is the target database. Start the `orcl2` non-CDB.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl2

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

$ sqlplus / as sysdba

Connected to an idle instance.

SQL> startup

ORACLE instance started.

Total System Global Area  503316480 bytes
Fixed Size                  2917144 bytes
Variable Size              272633064 bytes
Database Buffers           222298112 bytes
Redo Buffers                5468160 bytes
Database mounted.
Database opened.
SQL>
```

3. In the “target” `orcl2` database, drop the `EXAMPLE` tablespace. Use the `$HOME/labs/Load/droptbs.sql` script.

```
SQL> @droptbs
```

```
User dropped.

User dropped.

User dropped.

User dropped.

Tablespace dropped.

$
```

4. Prepare the source `orcl` database for FULL TRANSPORTABLE exportation. In another terminal session, connect to the source `orcl` database.

```
$ . oraenv
ORACLE_SID = [orcl2] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL>
```

5. Put the user-defined tablespaces in the source database `orcl` in read-only mode.
  - a. Create new tablespace to be transported with other tablespaces into `orcl2` database.

```
SQL> CREATE TABLESPACE test DATAFILE
'/u01/app/oracle/oradata/orcl/test01.dbf' size 5M;

Tablespace created.

SQL>
```

- b. Create a table `HR.TESTTAB` in the `TEST` tablespace, insert rows, and commit. You will check at the end of the FULL TRANSPORTABLE operation from `orcl` to `orcl2` database if the `HR.TESTTAB` has been transported in a `TEST` tablespace in the `orcl2` database. Execute the `$HOME/labs/Load/testtab.sql` script.

```
SQL> @testtab
```

```
Table created.

1 row created.

1 row created.

Commit complete.

SQL>
```

- c. Find the list of user-defined tablespaces to be put in read-only mode.

```
SQL> SELECT tablespace_name FROM dba_tablespaces ORDER BY 1;

TABLESPACE_NAME
-----
EXAMPLE
SYSAUX
SYSTEM
TEMP
TEST
UNDOTBS1
USERS

7 rows selected.

SQL>
```

- d. The list may be different from yours according to the tablespaces created during the training session. Make all tablespaces except SYSTEM, SYSAUX, TEMP, and UNDOTBS1 read-only.

```
SQL> ALTER TABLESPACE example READ ONLY;

Tablespace altered.

SQL> ALTER TABLESPACE test READ ONLY;

Tablespace altered.

SQL> ALTER TABLESPACE users READ ONLY;

Tablespace altered.

SQL>
```

- e. Find the list of data files associated to the read-only tablespaces that need to be transported.

```
SQL> SELECT file_name FROM dba_data_files
      WHERE tablespace_name IN
            ('EXAMPLE','TEST','USERS');

  2      3
FILE_NAME
-----
/u01/app/oracle/oradata/orcl/users01.dbf
/u01/app/oracle/oradata/orcl/example01.dbf
/u01/app/oracle/oradata/orcl/test01.dbf

SQL> EXIT
$
```

6. Export the orcl database in full transportable mode.

```
$ rm /u01/app/oracle/admin/orcl/dpdump/expfull.dmp
rm: cannot remove
`/u01/app/oracle/admin/orcl/dpdump/expfull.dmp': No such file or
directory
$ expdp system DUMPFILE=expfull.dmp FULL=Y TRANSPORTABLE=ALWAYS
LOGFILE=exp.log

Export: Release 12.1.0.2.0 - Production on Thu May 22 09:22:32
2014
...
```

```
Starting "SYSTEM"."SYS_EXPORT_FULL_01":  system/*****
DUMPFILE=expfull.dmp FULL=Y TRANSPORTABLE=ALWAYS LOGFILE=exp.log
Estimate in progress using BLOCKS method...
Processing object type
DATABASE_EXPORT/PLUGTS_FULL/FULL/PLUGTS_TABLESPACE
Processing object type DATABASE_EXPORT/PLUGTS_FULL/PLUGTS_BLK ...
Processing object type
DATABASE_EXPORT/SCHEMA/TABLE/INDEX/DOMAIN_INDEX/SECONDARY_TABLE/
INDEX/INDEX
ORA-39340: unsupported object,
INDEX:"SH"."SYS_IL0000096551C00006$$" will be skipped.
ORA-39340: unsupported object,
INDEX:"SH"."SYS_IL0000096554C00002$$" will be skipped.
...
. . exported "SYSTEM"."SCHEDULER_PROGRAM_ARGS"          9.515
KB          12 rows
. . exported "SYS"."AUDTAB$TBS$FOR_EXPORT"              5.953
KB           2 rows
```

```

. . exported "SYS"."NACL$_ACE_EXP"                      9.929
KB          1 rows
. . exported "SYS"."NACL$_HOST_EXP"                      6.914
KB          1 rows
. . exported "WMSYS"."WM$EXP_MAP"                        7.718
KB          3 rows
. . exported "SYS"."DBA_SENSITIVE_DATA"                  0
KB          0 rows
. . exported "SYS"."DBA_TSDP_POLICY_PROTECTION"          0
KB          0 rows
. . exported "SYS"."FGA_LOG$FOR_EXPORT"                  0
KB          0 rows
. . exported "SYS"."NACL$_WALLET_EXP"                    0
KB          0 rows
. . exported "SYSTEM"."SCHEDULER_JOB_ARGS"               0
KB          0 rows
Master table "SYSTEM"."SYS_EXPORT_FULL_01" successfully
loaded/unloaded
*****
*
Dump file set for SYSTEM.SYS_EXPORT_FULL_01 is:
  /u01/app/oracle/admin/orcl/dpdump/expfull.dmp
*****
*
Datafiles required for transportable tablespace EXAMPLE:
  /u01/app/oracle/oradata/orcl/example01.dbf
Datafiles required for transportable tablespace TEST:
  /u01/app/oracle/oradata/orcl/test01.dbf
Datafiles required for transportable tablespace USERS:
  /u01/app/oracle/oradata/orcl/users01.dbf
Job "SYSTEM"."SYS_EXPORT_FULL_01" completed with 2 error(s) at
Thu May 22 09:26:26 2014 elapsed 0 00:03:43

$

```

There might be errors during the export. These errors will not impact the import operation.

7. View the log file `exp.log` to get the list of data files to be transported before the full transportable import.
  - a. Find the `exp.log` file.

```

$ cd /u01/app/oracle/admin/orcl/dpdump
$ ls -ltr exp*
-rw-r--r-- 1 oracle oinstall    1171 Apr 30 05:51 export.log
-rw-r----- 1 oracle oinstall 7254016 May  2 04:44 expfull.dmp
-rw-r--r-- 1 oracle oinstall   12699 May  2 04:44 exp.log
$

```

- b. View the last lines of the `exp.log` file.

```
$ tail -20 exp.log
. . exported "SYSTEM"."SCHEDULER_PROGRAM_ARGS"          9.515
KB      12 rows
. . exported "SYS"."AUDTAB$TBS$FOR_EXPORT"              5.953
KB       2 rows
. . exported "SYS"."DBA_SENSITIVE_DATA"                  0
KB       0 rows
. . exported "SYS"."DBA_TSDP_POLICY_PROTECTION"          0
KB       0 rows
. . exported "SYS"."NACL$_ACE_EXP"                      9.929
KB       1 rows
. . exported "SYS"."NACL$_HOST_EXP"                     6.914
KB       1 rows
. . exported "SYS"."NACL$_WALLET_EXP"                    0
KB       0 rows
. . exported "WMSYS"."WM$EXP_MAP"                       7.718
KB       3 rows
Master table "SYSTEM"."SYS_EXPORT_FULL_01" successfully
loaded/unloaded
*****
*
Dump file set for SYSTEM.SYS_EXPORT_FULL_01 is:
/u01/app/oracle/admin/orcl/dpdump/expfull.dmp
*****
*
Datafiles required for transportable tablespace EXAMPLE:
/u01/app/oracle/oradata/orcl/example01.dbf
Datafiles required for transportable tablespace TEST:
/u01/app/oracle/oradata/orcl/test01.dbf
Datafiles required for transportable tablespace USERS:
/u01/app/oracle/oradata/orcl/users01.dbf
Job "SYSTEM"."SYS_EXPORT_FULL_01" completed with 2 error(s) at
Thu May 22 09:26:26 2014 elapsed 0 00:03:43
$
```

8. Copy the data files to the target locations `/u01/app/oracle/oradata/orcl2` and the export dump file to `/u01/app/oracle/admin/orcl2/dpdump`. Before proceeding, check that there are not any tablespaces in the target `orcl2` database having the same names as the tablespaces in the source `orcl` database.

- a. Set your environment to the target database `orcl2`.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl2
```

```

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL>

```

- b. Select the tablespace names.

```

SQL> SELECT tablespace_name FROM dba_tablespaces;

TABLESPACE_NAME
-----
SYSTEM
SYSAUX
UNDOTBS1
TEMP
USERS

SQL>

```

- c. Rename the USERS tablespace to USERS\_NEW and the data file  
/u01/app/oracle/oradata/orcl2/users01.dbf to  
/u01/app/oracle/oradata/orcl2/users\_new01.dbf.

```

SQL> ALTER TABLESPACE users RENAME TO users_new;

Tablespace altered.

SQL> ALTER TABLESPACE users_new OFFLINE;

Tablespace altered.

SQL> EXIT
$

```

```

$ mv /u01/app/oracle/oradata/orcl2/users01.dbf
/u01/app/oracle/oradata/orcl2/users_new01.dbf
$

```

```

$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> ALTER DATABASE RENAME FILE
'/u01/app/oracle/oradata/orcl2/users01.dbf' TO
'/u01/app/oracle/oradata/orcl2/users_new01.dbf';

Database altered.

SQL> ALTER TABLESPACE users_new ONLINE;

Tablespace altered.

SQL> SELECT tablespace_name FROM dba_tablespaces ORDER BY 1;

TABLESPACE_NAME
-----
SYSAUX
SYSTEM
TEMP
UNDOTBS1
USERS_NEW

SQL> EXIT
$

```

- d. Now you can copy the data files to the target locations  
 /u01/app/oracle/oradata/orcl2 and the export dump file to  
 /u01/app/oracle/admin/orcl2/dpdump.

```

$ cp /u01/app/oracle/oradata/orcl/test01.dbf
/u01/app/oracle/oradata/orcl/example01.dbf
/u01/app/oracle/oradata/orcl/users01.dbf
/u01/app/oracle/oradata/orcl2
$ cp /u01/app/oracle/admin/orcl/dpdump/expfull.dmp
/u01/app/oracle/admin/orcl2/dpdump/expfull.dmp
$

```



9. Import the `orcl1` database into the `orcl2` database in full transportable mode. There are many errors due to existing objects in the target `orcl2` database. These errors can be ignored.

```
$ rm /u01/app/oracle/admin/orcl2/dpdump/import.log
rm: cannot remove
`/u01/app/oracle/admin/orcl2/dpdump/import.log': No such file or
directory
$ impdp system FULL=Y dumpfile=expfull.dmp
TRANSPORT_DATAFILES='/u01/app/oracle/oradata/orcl2/test01.dbf','
/u01/app/oracle/oradata/orcl2/users01.dbf','/u01/app/oracle/orad
ata/orcl2/example01.dbf' logfile=import.log
...
Password: *****
...
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully
loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01":  system/***** FULL=Y
dumpfile=expfull.dmp
TRANSPORT_DATAFILES=/u01/app/oracle/oradata/orcl2/test01.dbf,/u0
1/app/oracle/oradata/orcl2/users01.dbf,/u01/app/oracle/oradata/o
rcl2/example01.dbf logfile=import.log
...
. . imported "WMSYS"."E$EXP_MAP"                                7.718
KB          3 rows
. . imported "SYS"."DP$DBA_SENSITIVE_DATA"                      0
KB          0 rows
. . imported "SYS"."DP$DBA_TSDP_POLICY_PROTECTION"              0
KB          0 rows
. . imported "SYS"."AMGT$DP$FGA_LOG$FOR_EXPORT"                 0
KB          0 rows
. . imported "SYS"."NACL$WALLET_IMP"                             0
KB          0 rows
. . imported "SYSTEM"."SCHEDULER_JOB_ARGS_TMP"                  0
KB          0 rows
Processing object type
DATABASE_EXPORT/NORMAL_POST_INSTANCE_IMPCALLOU/MARKER
Processing object type
DATABASE_EXPORT/SCHEMA/TABLE/PROCACT_INSTANCE
Processing object type DATABASE_EXPORT/SCHEMA/TABLE/TABLE
ORA-39151: Table "SCOTT"."DEPT" exists. All dependent metadata
and data will be skipped due to table_exists_action of skip
ORA-39151: Table "SCOTT"."EMP" exists. All dependent metadata
and data will be skipped due to table_exists_action of skip
...
ORA-39083: Object type
PROCOBJ:"APEX_040200"."ORACLE_APEX_DAILY_MAINTENANCE" failed to
create with error:
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```

ORA-27477: "APEX_040200"."ORACLE_APEX_DAILY_MAINTENANCE" already
exists
Failing sql is:
BEGIN
dbms_scheduler.create_job('"ORACLE_APEX_DAILY_MAINTENANCE"',
job_type=>'STORED_PROCEDURE', job_action=>
'WWV_FLOW_MAINT.DAILY_MAINTENANCE'
, number_of_arguments=>0,
start_date=>TO_TIMESTAMP_TZ('20-APR-2014 12.38.41.270594000 PM -
07:00','DD-MON-RRRR HH.MI.SSXXFF AM
TZR','NLS_DATE_LANGUAGE=english'), repeat_inter
Processing object type
DATABASE_EXPORT/SCHEMA/POST_SCHEMA/PROACT_SCHEMA
Processing object type
DATABASE_EXPORT/AUDIT_UNIFIED/AUDIT_POLICY_ENABLE
ORA-39083: Object type AUDIT_POLICY_ENABLE:"ORA_SECURECONFIG"
failed to create with error:
ORA-46350: Audit policy ORA_SECURECONFIG already applied with
the BY clause.
Failing sql is:
AUDIT POLICY "ORA_SECURECONFIG" EXCEPT "DBSNMP"
Processing object type
DATABASE_EXPORT/POST_SYSTEM_IMPCALLOUT/MARKER
Job "SYSTEM"."SYS_IMPORT_FULL_01" completed with 37 error(s) at
Thu May 22 09:35:59 2014 elapsed 0 00:03:41
$

```

10. Check in the target `orcl2` database that the tablespaces `TEST`, `EXAMPLE`, and `USERS` have been plugged and that the `HR.TESTTAB` table contains two rows as in the source `orcl` database.

```

$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> SELECT tablespace_name FROM dba_tablespaces;

TABLESPACE_NAME
-----
SYSTEM
SYSAUX
UNDOTBS1

```

```

TEMP
USERS_NEW
EXAMPLE
TEST
USERS

8 rows selected.

SQL> SELECT name FROM v$datafile;

NAME
-----
/u01/app/oracle/oradata/orcl2/system01.dbf
/u01/app/oracle/oradata/orcl2/test01.dbf
/u01/app/oracle/oradata/orcl2/sysaux01.dbf
/u01/app/oracle/oradata/orcl2/undotbs01.dbf
/u01/app/oracle/oradata/orcl2/example01.dbf
/u01/app/oracle/oradata/orcl2/users_new01.dbf
/u01/app/oracle/oradata/orcl2/users01.dbf

7 rows selected.

SQL> SELECT * FROM hr.testtab;

      ID LABEL
-----
      10 Skirt
      20 Trousers

SQL> EXIT
$

```

11. Put the user-defined tablespaces in the source database `orcl` back in read-write mode in order to let users work.

```

$ . oraenv
ORACLE_SID = [orcl2] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

```

```
SQL> ALTER TABLESPACE example READ WRITE;

Tablespace altered.

SQL> ALTER TABLESPACE test READ WRITE;

Tablespace altered.

SQL> ALTER TABLESPACE users  READ WRITE;

Tablespace altered.

SQL> EXIT
$
```

## Practice 22-2: Loading Data Using SQL\*Loader Express Mode (Optional)

---

### Overview

In this practice, you will load records from a `tab1.dat` file into the `HR.TAB1` table using SQL\*Loader in Express Mode.

### Tasks

1. Make sure that you are in the `~/labs/Load` directory and that your environment points to the `orcl` instance.

```
$ cd ~/labs/Load
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$
```

2. Execute the `$HOME/labs/Load/tab1.sql` to create an `HR.TAB1` table.

```
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> @tab1
DROP TABLE hr.tab1 PURGE;
DROP TABLE hr.tab1 PURGE
          *
ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

1 row created.

1 row created.

Commit complete.

$
```

3. Display the \$HOME/labs/Load/tab1.dat file. It contains five records.

```
$ more tab1.dat
30, Shirt
40, Socks
50, Cap
60, Gloves
70, Tie
$
```

4. Load the five records into the HR.TAB1 table using SQL\*Loader in Express Mode.

```
$ sqlldr hr TABLE=hr.tab1
Password: *****
SQL*Loader: Release 12.1.0.2.0 - Production on Thu May 22
09:45:01 2014

Copyright (c) 1982, 2014, Oracle and/or its affiliates. All
rights reserved.

Express Mode Load, Table: HR.TAB1
Path used:          External Table, DEGREE_OF_PARALLELISM=AUTO

Table HR.TAB1:
  5 Rows successfully loaded.

Check the log files:
  hr.log
  hr_%p.log_xt
for more information about the load.
$
```

5. Verify the existence of the log files.

```
$ ls -l hr*
-rw-r--r-- 1 oracle oinstall 1036 May  2 05:06 hr_17957.log_xt
-rw-r--r-- 1 oracle oinstall 2240 May  2 05:06 hr.log
$
```

6. View the `hr.log` file.

```
$ more hr.log

SQL*Loader: Release 12.1.0.2.0 - Production on Fri May 2
05:06:28 2014

Copyright (c) 1982, 2014, Oracle and/or its affiliates. All
rights reserved.

Express Mode Load, Table: HR.TAB1
Data File:      tab1.dat
  Bad File:      tab1.bad
  Discard File:  none specified

(Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Continuation:    none specified
Path used:      External Table

Table HR.TAB1, loaded from every logical record.
Insert option in effect for this table: APPEND

      Column Name      Position   Len    Term Encl Datatype
-----
ID                      FIRST      *      ,      CHARACTER
PROD_NAME                NEXT      *      ,      CHARACTER

Generated control file for possible reuse:
OPTIONS(EXTERNAL_TABLE=EXECUTE, TRIM=LRTRIM)
LOAD DATA
INFILE '(null)'
APPEND
INTO TABLE HR.TAB1
FIELDS TERMINATED BY ","
(
  ID,
  PROD_NAME
)
End of generated control file for possible reuse.
```

```
created temporary directory object SYS_SQLLDR_XT_TMPDIR_00000
for path /home/oracle/labs/Load
```

```
enable parallel DML: ALTER SESSION ENABLE PARALLEL DML
```

```
creating external table "SYS_SQLLDR_X_EXT_TAB1"
```

```
CREATE TABLE "SYS_SQLLDR_X_EXT_TAB1"
```

```
(
  "ID" NUMBER,
  "PROD_NAME" VARCHAR2(10)
)
```

```
ORGANIZATION external
```

```
(
  TYPE oracle_loader
  DEFAULT DIRECTORY SYS_SQLLDR_XT_TMPDIR_00000
  ACCESS PARAMETERS
  (
    RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    BADFILE 'SYS_SQLLDR_XT_TMPDIR_00000':'tab1.bad'
    LOGFILE 'hr_%p.log_xt'
    READSIZE 1048576
    FIELDS TERMINATED BY "," LRTRIM
    REJECT ROWS WITH ALL NULL FIELDS
  (
    "ID" CHAR(255),
    "PROD_NAME" CHAR(255)
  )
)
```

```
location
```

```
(
  'tab1.dat'
)
)REJECT LIMIT UNLIMITED
```

```
executing INSERT statement to load database table HR.TAB1
```

```
INSERT /*+ append parallel(auto) */ INTO HR.TAB1
```

```
(
  ID,
  PROD_NAME
)
```



```

SELECT
  "ID",
  "PROD_NAME"
FROM "SYS_SQLLDR_X_EXT_TAB1"

dropping external table "SYS_SQLLDR_X_EXT_TAB1"

Table HR.TAB1:
  5 Rows successfully loaded.

Run began on Thu May 22 09:45:01 2014
Run ended on Thu May 22 09:45:06 2014

Elapsed time was:      00:00:04.68
CPU time was:         00:00:00.01
$

```

7. View the hr\_17957.log\_xt file.

```

$ more hr_17957.log_xt

LOG file opened at 05/22/14 09:45:06

Field Definitions for table SYS_SQLLDR_X_EXT_TAB1
Record format DELIMITED BY NEWLINE
Data in file has same endianness as the platform
Reject rows with all null fields

Fields in Data Source:

      ID                                CHAR (255)
      Terminated by ","
      Trim whitespace from left and right
      PROD_NAME                        CHAR (255)
      Terminated by ","
      Trim whitespace from left and right

LOG file opened at 05/22/14 09:45:06

KUP-05004:  Warning: Intra source concurrency disabled because
parallel select was not requested.

Field Definitions for table SYS_SQLLDR_X_EXT_TAB1

```

```
Record format DELIMITED BY NEWLINE
Data in file has same endianness as the platform
Reject rows with all null fields
```

Fields in Data Source:

```

ID                                CHAR (255)
  Terminated by ","
  Trim whitespace from left and right
PROD_NAME                        CHAR (255)
  Terminated by ","
  Trim whitespace from left and right
```

\$

8. Verify that the records have been inserted into the HR.TAB1 table.

```
$ sqlplus / as sysdba
```

Connected to:

Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -  
64bit Production

With the Partitioning, OLAP, Advanced Analytics, Real  
Application Testing and Unified Auditing options

```
SQL> SELECT * FROM hr.tab1;
```

```

      ID PROD_NAME
-----
      10 Skirt
      20 Trousers
      30  Shirt
      40  Socks
      50   Cap
      60  Gloves
      70   Tie
```

```
SQL>
```

9. Drop the HR.TAB1 table.

```
SQL> DROP TABLE hr.tab1 PURGE;
```

Table dropped.

```
SQL> EXIT
```

\$

## **Practices for Lesson 23: Partitioning Enhancements**

### **Chapter 23**

## Practices for Lesson 23: Overview

---

### Practices Overview

In this practice, you will familiarize yourself with using partial local and global indexes.

## Practice 23-1: Local and Global Partial Indexing on Partitioned Tables

### Overview

In this practice, you will create a partitioned table with five partitions: A local partitioned index indexing only two partitions of the table and therefore composed of three index partitions, and a global index indexing the rows of only two partitions of the table.

### Tasks

1. Make sure you are in the ~/labs/Part directory.

```
$ cd ~/labs/Part
$
```

2. Create the partitioned table HR.TAB\_PART1 with five partitions and only three local index partitions.

- a. Connect to the source database orcl.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL>
```

- b. To be sure that the segments, tables, partitions and indexes are created without inserting rows, set the deferred\_segment\_creation parameter to FALSE.

```
SQL> ALTER SYSTEM SET deferred_segment_creation=FALSE;

System altered.

SQL>
```

- c. Execute the \$HOME/labs/Part/tab\_part1.sql script to create the table.

```
SQL> @tab_part1
DROP TABLE hr.tab_part1 PURGE
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

```
Table created.
```

```
SQL>
```

3. Check the default indexing value of the table created.

```
SQL> SELECT def_indexing FROM dba_part_tables
       WHERE table_name='TAB_PART1';
```

```
2
```

```
DEF
```

```
---
```

```
OFF
```

```
SQL>
```

4. Create a partial global index as follows.

```
SQL> CREATE INDEX hr.tab_part1_gidx_ordermode
       ON hr.tab_part1 (order_mode)
       GLOBAL INDEXING PARTIAL;
```

```
2      3
```

```
Index created.
```

```
SQL>
```

5. Create a partial local partitioned index as follows.

```
SQL> CREATE INDEX hr.tab_part1_lidx_orderdate
       ON hr.tab_part1 (order_date)
       LOCAL INDEXING PARTIAL;
```

```
2      3
```

```
Index created.
```

```
SQL>
```

6. Check the indexing type and status of the indexes. Use the \$HOME/labs/Part/view.sql script to display the indexes attributes.

```
SQL> @view
```

```
INDEXING INDEX_NAME                                STATUS
```

```
-----
```

```
PARTIAL TAB_PART1_LIDX_ORDERDATE                   N/A
```

```
PARTIAL TAB_PART1_GIDX_ORDERMODE                   VALID
```

```
SQL>
```

7. Use the \$HOME/labs/Part/status.sql script to check the status of the index partitions of the partial local index.

```
SQL> @status
```

INDEX_NAME	PARTITION_NAME	STATUS
TAB_PART1_LIDX_ORDERDATE	ORD_P5	UNUSABLE
TAB_PART1_LIDX_ORDERDATE	ORD_P4	UNUSABLE
TAB_PART1_LIDX_ORDERDATE	ORD_P3	<b>USABLE</b>
TAB_PART1_LIDX_ORDERDATE	ORD_P2	UNUSABLE
TAB_PART1_LIDX_ORDERDATE	ORD_P1	<b>USABLE</b>

```
SQL>
```

8. Insert rows into the table. Execute the \$HOME/labs/Part/ins.sql script

```
SQL> @ins
```

1 row created.

1 row created.

1 row created.

1 row created.

Commit complete.

```
SQL>
```

9. Check that the partial global index TAB\_PART1\_GIDX\_ORDERMODE is used while performing queries on the highly selective column ORDER\_MODE to access a single value such as direct. The optimizer has to rely on a full scan of the non-indexed partitions for the value such as online stored in rows in partitions that are not indexed.

- a. Collect statistics for the table.

```
SQL> EXEC dbms_stats.gather_table_stats ('HR','TAB_PART1')
```

PL/SQL procedure successfully completed.

```
SQL>
```

- b. Generate a plan for a query accessing the value direct of the ORDER\_MODE in the table where there are few rows.

```
SQL> EXPLAIN PLAN FOR
      SELECT order_mode, order_status FROM hr.tab_part1
```

```

        WHERE order_mode='direct';

2      3
Explained.

SQL> select * from table(dbms_xplan.display) ;

PLAN_TABLE_OUTPUT
-----
Plan hash value: 3230465927
-----

| Id | Operation                                | Name | | | | |
|---|---|---|---|---|---|---|
| Rows | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
|-----|-----|-----|
| 0 | SELECT STATEMENT                                |      |
| 2 |      18 |      30  (0)| 00:00:01 |      |      |
| 1 | VIEW                                            | VW_TE_2 |
| 3 |      57 |      30  (0)| 00:00:01 |      |      |
| 2 | UNION-ALL                                         |      |
|   |   |   |   |   |   |   |
| * 3 | TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED | TAB_PART1 |
| 1 |      17 |      3  (0)| 00:00:01 | ROWID | ROWID |
| * 4 | INDEX RANGE SCAN                                |      |
TAB_PART1_GIDX_ORDERMODE
| 2 |      1 |      1  (0)| 00:00:01 |      |      |
| 5 | PARTITION RANGE OR                                |      |
| 2 |      34 |      27  (0)| 00:00:01 | KEY(OR) | KEY(OR) |
| * 6 | TABLE ACCESS FULL                                | TAB_PART1 |
| 2 |      34 |      27  (0)| 00:00:01 | KEY(OR) | KEY(OR) |
-----

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

3 - filter("TAB_PART1"."ORDER_DATE"<TO_DATE(' 1999-10-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
        "TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-07-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') OR
        "TAB_PART1"."ORDER_DATE"<TO_DATE('
        1999-03-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
4 - access("ORDER_MODE"='direct')
6 - filter("ORDER_MODE"='direct' AND
        ("TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-10-01 00:00:00',
        'syyyy-mm-dd hh24:mi:ss') AND

```



```

        "TAB_PART1"."ORDER_DATE"<TO_DATE(' 2010-03-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') OR
"TAB_PART1"."ORDER_DATE"<TO_DATE('
        1999-07-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
AND "TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-03-01 00:00:00',
'syyyy-mm-ddhh24:mi:ss'))

25 rows selected.

SQL>

```

The partial global index is used to access the two partitions of the table where the rows containing the direct value in the ORDER\_MODE column are stored.

10. Load the ord\_p1 partition. This partition is indexed. So both the partial global and local indexes will be updated. Execute the \$HOME/labs/Part/load.sql script.

```

SQL> @load

1 row created.

2 rows created.

4 rows created.

8 rows created.

16 rows created.

32 rows created.

64 rows created.

128 rows created.

256 rows created.

512 rows created.

1024 rows created.

Commit complete.

SQL>

```

11. Generate the plan for a query by using the key of the partial global index.

```
SQL> EXEC dbms_stats.set_table_prefs('HR','TAB_PART1','CASCADE',
'DBMS_STATS.AUTO_CASCADE')
```

PL/SQL procedure successfully completed.

```
SQL> EXEC dbms_stats.gather_table_stats ('HR','TAB_PART1')
```

PL/SQL procedure successfully completed.

```
SQL>
```

```
SQL> EXPLAIN PLAN FOR
```

```
      SELECT order_mode, order_status FROM hr.tab_part1 WHERE
order_mode='direct';
```

2

Explained.

```
SQL>
```

12. You see that the partial local index is used to access the ord\_p3 partition.

```
SQL> select * from table(dbms_xplan.display) ;
```

PLAN\_TABLE\_OUTPUT

-----  
Plan hash value: 1639651856  
-----

Id	Operation	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	Name
0	SELECT STATEMENT							
2050		18450		43 (0)	00:00:01			
1	VIEW							VW_TE_2
3600		68400		43 (0)	00:00:01			
2	UNION-ALL							
3	CONCATENATION							
4	PARTITION RANGE SINGLE							
2048		34816		14 (0)	00:00:01	1	1	
* 5	TABLE ACCESS FULL							
TAB_PART1					2048	34816	14 (0)	00:00:01
1		1						

```

| 6 | PARTITION RANGE SINGLE | | | | |
| 1 | 17 | 2 (0) | 00:00:01 | 3 | 3 |
| * 7 | TABLE ACCESS BY LOCAL INDEX ROWID BATCHED |
TAB_PART1 | 1 | 17 | 2 (0) | 00:00:01
| 3 | 3 |
| * 8 | INDEX RANGE SCAN |
TAB_PART1_LIDX_ORDERDATE | 1 | 1 (0) | 00:00:01
| 3 | 3 | | | | |
| 9 | PARTITION RANGE OR |
| 1551 | 26367 | 27 (0) | 00:00:01 | KEY(OR) | KEY(OR) |
| * 10 | TABLE ACCESS FULL |
TAB_PART1 | 1551 | 26367 | 27 (0) | 00:00:01
| KEY(OR) | KEY(OR) |
-----

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

      5 - filter("ORDER_MODE"='direct' AND
"TAB_PART1"."ORDER_DATE"<TO_DATE(' 1999-03-01 00:00:00', 'syyy-
mm-dd hh24:mi:ss'))
      7 - filter("ORDER_MODE"='direct')
      8 - access("TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-07-01
00:00:00', 'syyy-mm-dd hh24:mi:ss') AND
          "TAB_PART1"."ORDER_DATE"<TO_DATE(' 1999-10-01
00:00:00', 'syyy-mm-dd hh24:mi:ss'))
          filter(LNNVL("TAB_PART1"."ORDER_DATE"<TO_DATE(' 1999-03-
01 00:00:00', 'syyy-mm-dd hh24:mi:ss'))))
     10 - filter(("TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-10-01
00:00:00', 'syyy-mm-dd hh24:mi:ss') AND
          "TAB_PART1"."ORDER_DATE"<TO_DATE(' 2010-03-01
00:00:00', 'syyy-mm-dd hh24:mi:ss') OR
"TAB_PART1"."ORDER_DATE"<TO_DATE('
          1999-07-01 00:00:00', 'syyy-mm-dd hh24:mi:ss')
AND "TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-03-01 00:00:00',
'syyy-mm-dd
          hh24:mi:ss')) AND "ORDER_MODE"='direct')

30 rows selected.

SQL>

```

13. Generate the plan for a query on the non-indexed partition `ord_p2`. A table access full is performed on the partition.

```
SQL> EXPLAIN PLAN FOR
      SELECT order_mode, order_status FROM hr.tab_part1
      WHERE order_date >=TO_DATE('1999-03-01 00:00:00','yyyy-mm-
dd hh24:mi:ss')
      AND   order_date <TO_DATE('1999-07-01 00:00:00','yyyy-mm-
dd hh24:mi:ss');
      2      3      4
Explained.

SQL> select * from table(dbms_xplan.display) ;

PLAN_TABLE_OUTPUT
-----
|  0 | SELECT STATEMENT |          | 1 | 17 | 14 |
| (0) | 00:00:01 |          |   |   |   |
|  1 | VIEW             | VW_TE_2  | 2 | 38 | 14 |
| (0) | 00:00:01 |          |   |   |   |
|  2 | UNION-ALL        |          |   |   |   |
|   |   |          |   |   |   |
| * 3 | FILTER           |          |   |   |   |
|   |   |          |   |   |   |
|  4 | PARTITION RANGE SINGLE |          | 1 | 17 |   |
14 (0) | 00:00:01 | 2 | 2 |   |
|  5 | TABLE ACCESS FULL | TAB_PART1 | 1 | 17 |   |
14 (0) | 00:00:01 | 2 | 2 |   |
|  6 | PARTITION RANGE SINGLE |          | 1 | 17 |   |
14 (0) | 00:00:01 | 2 | 2 |   |
|  7 | TABLE ACCESS FULL | TAB_PART1 | 1 | 17 |   |
14 (0) | 00:00:01 | 2 | 2 |   |
-----
-
Predicate Information (identified by operation id):
-----

      3 - filter(NULL IS NOT NULL)

19 rows selected.

SQL>
```

14. Generate the plan for a query on the indexed partition `ord_p1` and `ord_p3`. A full scan is performed on the large partition `ord_p1` and the partial local index is used to access the `ord_p3` partition rows.

```
SQL> EXPLAIN PLAN FOR
      SELECT order_mode, order_status FROM hr.tab_part1
      WHERE order_date <TO_DATE('1999-03-01 00:00:00','syyy-MM-dd hh24:mi:ss')
      OR (order_date between TO_DATE('1999-07-01
00:00:00','syyy-MM-dd hh24:mi:ss') AND TO_DATE('1999-10-01
00:00:00','syyy-MM-dd hh24:mi:ss'));
      2      3      4
Explained.

SQL> select * from table(dbms_xplan.display) ;

PLAN_TABLE_OUTPUT
-----
Plan hash value: 2109685904
-----
```

Id	Operation	Name				
Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop	
0	SELECT STATEMENT					
3337	56729	30 (0)	00:00:01			
1	VIEW	VW_TE_2				
6621	122K	30 (0)	00:00:01			
2	UNION-ALL					
3	CONCATENATION					
4	PARTITION RANGE SINGLE					
1	17	2 (0)	00:00:01	3	3	
5	TABLE ACCESS BY LOCAL INDEX ROWID BATCHED					
TAB_PART1			1	17	2 (0)	00:00:01
3	3					
* 6	INDEX RANGE SCAN					
TAB_PART1_LIDX_ORDERDATE			1		1 (0)	00:00:01
3	3					
7	PARTITION RANGE SINGLE					
4095	69615	14 (0)	00:00:01	1	1	
* 8	TABLE ACCESS FULL					
TAB_PART1			4095	69615	14 (0)	00:00:01
1	1					
9	PARTITION RANGE OR					
2525	42925	14 (0)	00:00:01	KEY(OR)	KEY(OR)	

```

|* 10 |      TABLE ACCESS FULL
TAB_PART1 |      2525 | 42925 |      14  (0) | 00:00:01
|KEY(OR) |KEY(OR) |
-----
Predicate Information (identified by operation id):
-----

      6 - access("TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-07-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
          "TAB_PART1"."ORDER_DATE"<TO_DATE(' 1999-10-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
          filter("ORDER_DATE"<TO_DATE(' 1999-03-01 00:00:00',
'syyyy-mm-dd hh24:mi:ss') OR "ORDER_DATE">=TO_DATE(' 1999-07-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
"ORDER_DATE"<=TO_DATE(' 1999-10-01 00:00:00', 'syyyy-mm-dd
hh24:mi:ss'))
      8 - filter("TAB_PART1"."ORDER_DATE"<TO_DATE(' 1999-03-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND ("ORDER_DATE"<TO_DATE('
1999-03-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') OR
"ORDER_DATE">=TO_DATE(' 1999-07-01 00:00:00', 'syyyy-mm-dd
hh24:mi:ss') AND
          "ORDER_DATE"<=TO_DATE(' 1999-10-01 00:00:00',
'syyyy-mm-dd hh24:mi:ss')) AND
(LNNVL("TAB_PART1"."ORDER_DATE">=TO_DATE('
1999-07-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
OR LNNVL("TAB_PART1"."ORDER_DATE"<TO_DATE(' 1999-10-01
00:00:00', 'syyyy-mm-dd
hh24:mi:ss'))))
     10 - filter(("TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-10-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
          "TAB_PART1"."ORDER_DATE"<TO_DATE(' 2010-03-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') OR
"TAB_PART1"."ORDER_DATE"<TO_DATE('
1999-07-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
AND "TAB_PART1"."ORDER_DATE">=TO_DATE(' 1999-03-01 00:00:00',
'syyyy-mm-dd
hh24:mi:ss')) AND ("ORDER_DATE">=TO_DATE(' 1999-
07-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND
"ORDER_DATE"<=TO_DATE(' 1999-10-01
00:00:00', 'syyyy-mm-dd hh24:mi:ss') OR
"ORDER_DATE"<TO_DATE(' 1999-03-01 00:00:00', 'syyyy-mm-dd
hh24:mi:ss'))))

35 rows selected.

SQL> EXIT
$

```

# **Practices for Lesson 24: SQL Enhancements and Migration Assistant for Unicode**

## **Chapter 24**

## Practices for Lesson 24: Overview

---

### Practices Overview

In the practice for this lesson, you use the extended data type column to create columns of 32767 bytes long and the row-limiting clause to limit the rows resulting from queries.



## Practice 24-1: Using 32K VARCHAR2 Data Type

### Overview

In this practice, you create a new table with a column of data type VARCHAR2 (32767).

### Tasks

1. Connect to the source database orcl.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL>
```

2. Create a table LONG\_VARCHAR with a column VARCHAR2 (32767).

```
SQL> CREATE TABLE long_varchar(id NUMBER,vc VARCHAR2(32767));
CREATE TABLE long_varchar(id NUMBER,vc VARCHAR2(32767))
*
ERROR at line 1:
ORA-00910: specified length too long for its datatype

SQL>
```

3. Set the instance parameter MAX\_STRING\_SIZE to EXTENDED.

```
SQL> alter system set MAX_STRING_SIZE =EXTENDED;
alter system set MAX_STRING_SIZE =EXTENDED
*
ERROR at line 1:
ORA-02097: parameter cannot be modified because specified value
is invalid
ORA-14694: database must in UPGRADE mode to begin
MAX_STRING_SIZE migration

SQL>
```

4. Configure the database to be compatible with extended data type columns.
  - a. Restart the database instance.

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

```
SQL> startup upgrade
ORACLE instance started.

Total System Global Area  3355443200 bytes
Fixed Size                  2921000 bytes
Variable Size               973082072 bytes
Database Buffers            2365587456 bytes
Redo Buffers                 13852672 bytes
Database mounted.
Database opened.
SQL>
```

- b. Set the instance parameter MAX\_STRING\_SIZE to the EXTENDED value.

```
SQL> ALTER SYSTEM SET MAX_STRING_SIZE = EXTENDED;

System altered.

SQL>
```

- c. Execute the \$ORACLE\_HOME/rdbms/admin/utl32k.sql script as SYSDBA.

```
SQL> @$ORACLE_HOME/rdbms/admin/utl32k.sql

Session altered.

DOC>#####
#
DOC>#####
#
DOC>  The following statement will cause an "ORA-01722: invalid
number"
DOC>  error if the database has not been opened for UPGRADE.
DOC>
DOC>  Perform a "SHUTDOWN ABORT" and
DOC>  restart using UPGRADE.
DOC>#####
DOC>#####
```

```

DOC>#

no rows selected

DOC>#####
DOC>#####
DOC>  The following statement will cause an "ORA-01722: invalid
number"
DOC>  error if the database does not have compatible >= 12.0.0
DOC>
DOC>  Set compatible >= 12.0.0 and retry.
DOC>#####
DOC>#####
DOC>#

PL/SQL procedure successfully completed.

Session altered.

2 rows updated.

Commit complete.

System altered.

PL/SQL procedure successfully completed.

Commit complete.

System altered.

Session altered.

PL/SQL procedure successfully completed.

No errors.

Session altered.

PL/SQL procedure successfully completed.

Commit complete.

Package altered.

```

Package altered.

SQL>

- d. Restart the database in normal mode.

SQL> **shutdown immediate**

Database closed.

Database dismounted.

ORACLE instance shut down.

SQL> **startup**

ORACLE instance started.

Total System Global Area 3355443200 bytes

Fixed Size 2921000 bytes

Variable Size 973082072 bytes

Database Buffers 2365587456 bytes

Redo Buffers 13852672 bytes

Database mounted.

Database opened.

SQL>

- e. Verify that the MAX\_STRING\_SIZE is set to EXTENDED.

SQL> **show parameter MAX\_STRING\_SIZE**

NAME	TYPE	VALUE
-----	-----	-----
max_string_size	string	<b>EXTENDED</b>

SQL>

5. Create a table with an extended data type column of 32767 bytes.

SQL> **CREATE TABLE long\_varchar(id NUMBER,vc VARCHAR2(32767));**

Table created.

SQL> **DESC long\_varchar**

Name	Null?	Type
-----	-----	-----
ID		NUMBER
VC		VARCHAR2 (32767)

SQL>

## Practice 24-2: Querying a Table Using a SQL Row-Limiting Clause (Optional)

### Overview

In this practice, you limit the number of rows returned by a query that orders data.

### Tasks

1. Create a new `HR.TEST` table and count the number of rows in the `HR.EMPLOYEES` table.

```
SQL> CREATE TABLE hr.test AS SELECT * FROM hr.employees;

Table created.

SQL> select count(*) from hr.test;

COUNT(*)
-----
          83

SQL>
```

2. Select the `EMPLOYEE_ID` and `LAST_NAME` of the first 10 employees ordered by their hire date.

```
SQL> SELECT employee_id, last_name, hire_date FROM hr.test
       ORDER BY hire_date
       FETCH FIRST 10 ROWS ONLY;

 2      3
EMPLOYEE_ID LAST_NAME                HIRE_DATE
-----
          102 De Haan                13-JAN-01
          203 Mavris                 07-JUN-02
          206 Gietz                  07-JUN-02
          205 Higgins                07-JUN-02
          204 Baer                   07-JUN-02
          109 Favier                 16-AUG-02
          108 Greenberg              17-AUG-02
          114 Raphaely               07-DEC-02
          122 Kaufling               01-MAY-03
          115 Khoo                   18-MAY-03

10 rows selected.

SQL>
```

You see the first 10 employees ordered by their hire date.

3. Select the `EMPLOYEE_ID`, `LAST_NAME` and `HIRE_DATE` of the next 5 employees ordered by their hire date.

```
SQL> SELECT employee_id,last_name, hire_date FROM hr.test
      ORDER BY hire_date
      OFFSET 10 ROWS FETCH NEXT 5 ROWS ONLY;
```

EMPLOYEE_ID	LAST_NAME	HIRE_DATE
100	King	17-JUN-03
137	Ladwig	14-JUL-03
200	Whalen	17-SEP-03
141	Rajs	17-OCT-03
184	Sarchand	27-JAN-04

```
SQL>
```

You see the next 5 employees ordered by their hire date.

4. Now, you work from two distinct sessions.
- In the first session, you retrieved the first 10 employees ordered by their hire date in task 2. In task 3, you retrieved the next five employees still ordered by their hire date. In another terminal session, delete an employee that belongs to the first 10 selected rows in the first session and first selection.

```
$ . oraenv
ORACLE_SID = [orcl2] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / as sysdba

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real
Application Testing and Unified Auditing options

SQL> delete hr.test where employee_id=122;

1 row deleted.

SQL> commit;

Commit complete.

SQL>
```

- b. Back to the first session, select the `EMPLOYEE_ID`, `LAST_NAME` and `HIRE_DATE` of the next 5 employees ordered by their hire date.

```
SQL> SELECT employee_id,last_name, hire_date FROM hr.test
       ORDER BY hire_date
       OFFSET 10 ROWS FETCH NEXT 5 ROWS ONLY;
```

EMPLOYEE_ID	LAST_NAME	HIRE_DATE
137	Ladwig	14-JUL-03
200	Whalen	17-SEP-03
141	Rajs	17-OCT-03
184	Sarchand	27-JAN-04
156	King	30-JAN-04

```
SQL> EXIT
$
```

Now, the employee 100 is in position 10. So the next 5 rows do not show the employee 100 anymore. It belongs to the first 10 employees.





# **Practices for Appendix C: Schema and Data Changes Management**

## **Chapter 25**

## Practices for Appendix C: Overview

---

### Practices Overview

In the practice for this lesson, you will use the Schema Change Plans demonstration to understand and view the steps required during schema change plan usage between two databases to synchronize two databases together.

## Appendix C-1: Using Schema Change Plans

---

### Overview

In this practice you use a browser to execute the Schema Change Plans demonstration.

### Tasks

1. Launch a browser and enter: file:  
`////home/oracle/demos/Schema_Change_Plans/Schema_change_plan.html.`

