# Oracle Database 11*g*: OCM Exam Preparation Workshop

**Activity Guide • Volume III**

D69748GC20

Edition 2.0

June 2013

D82219

**Author**

Setsuko Fujitani

**Technical Contributors and Reviewers**

Sharath Bhujani

Joel Goodman

Setsuko Fujitani

Lakshmi Narapareddi

**Editors**

Vijayalakshmi Narasimhan

Rashmi Rajagopal

Malavika Jinka

**Publishers**

Sujatha Nagendra

Joseph Fernandez

# Contents

**Appendix B: Solutions**

# Appendix C: Manual Solutions

## Manual Solutions for Practice 1-1: Configuring the Initial Oracle Network Environment

### Your Tasks

1. First, stop the default listener, and then create a `LISTENER` listener. Use the following information:

| **Object** | **Setting** |
|---|---|
| Listener name | `LISTENER` |
| Host | `<fully qualified host name of your odd PC>` |
| Protocol | TCP/IP |
| Port | `1521` |

a) Create the `listener.ora` file in the `$ORACLE_HOME/network/admin` directory. The `listener.ora` file includes the following:

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = <fully qualified host
name of your odd PC>)(PORT = 1521))
    )
  )
```

b) Stop the default listener by using the `lsnrctl` command. Start the `LISTENER` listener.

```
$ lsnrctl stop
$ lsnrctl start
```

2. Configure local naming methods for the new `orcl` database and the `PROD1` database (already exists). The `orcl` database will be created in the practice titled “Practice 1-2: Creating an Oracle Database.” Use the following information:

| **Object** | **Setting** |
|---|---|
| Service Name | `orcl.oracle.com` |
| Protocol | TCP/IP |
| Port | `1521` |
| Host | `IP address or fully qualified host name of your odd PC` |
| Net Service Name | `orcl` |

| **Object** | **Setting** |
|---|---|
| Service Name | `PROD1.us.oracle.com` |

| | |
|---|---|
| Protocol | TCP/IP |
| Port | 1521 |
| Host | IP address or fully qualified host name of your odd PC |
| Net Service Name | PROD1 |

a) Create the tnsnames.ora file in the $ORACLE_HOME/network/admin directory. The tnsnames.ora file includes the following:

```
orcl =
  (DESCRIPTION =
    (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = <fully qualified host name of your odd PC>)
        (PORT = 1521)
    )
    (CONNECT_DATA =
      (SERVICE_NAME = orcl.oracle.com)
    )
  )

PROD1 =
  (DESCRIPTION =
    (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = <fully qualified host name of your odd PC>)
        (PORT = 1521)
    )
    (CONNECT_DATA =
      (SERVICE_NAME = PROD1.us.oracle.com)
    )
  )
```

## Manual Solutions for Practice 1-2: Creating an Oracle Database

**Background:** You are about to begin creating your first Oracle database. You anticipate that several similar databases will be needed in the near future. Therefore, you decide to create your `orcl` database, as well as a database template and the database creation scripts. Locate the scripts in the `/home/oracle/labs` directory (which is the directory that you use most often throughout this course).

**Note:** Completing the database creation is critical for all following practice sessions.

### Your Tasks

1. Create a database with the following settings:

| Object | Setting |
|---|---|
| Global Database Name | `orcl.oracle.com` |
| SID | `orcl` |
| Password for All Accounts | `oracle` |
| Character Set | `AL32UTF8` |
| National Character Set | `AL16UTF16` |
| Database File Directory | `/u01/app/oracle/oradata/orcl` |
| Block Size | 8 KB |
| Fast Recovery Area | `/u01/app/oracle/fast_recovery_area` |
| Fast Recovery Area Size | 10 GB |
| Undo Management | AUTO |
| Memory Size | Automatic Memory Management 800 MB |
| Controlfile name | `control0*.ctl` |
| Default tablespace | `USERS` |
| UNDO tablespace | `UNDOTBS1` |
| Temporary tablespace | `TEMP` |

   a) Create the initialization parameter file `initorcl.ora` in the `$ORACLE_HOME/dbs` directory. An example is as follows:

```
db_name=orcl
control_files = (/u01/app/oracle/oradata/orcl/control01.ctl,
                 /u01/app/oracle/oradata/orcl/control02.ctl)
memory_target=800M
processes = 150
db_block_size=8192
db_domain=oracle.com
db_recovery_file_dest='/u01/app/oracle/fast_recovery_area'
db_recovery_file_dest_size=10G
diagnostic_dest='/u01/app/oracle'
open_cursors=300
remote_login_passwordfile='EXCLUSIVE'
undo_tablespace='UNDOTBS1'
compatible ='11.2.0'
```

b) Create the `cre_db.sql` file in the `/home/oracle` directory with the following `CREATE DATABASE` statement. An example is as follows:

```
CREATE DATABASE orcl
   USER SYS IDENTIFIED BY oracle
   USER SYSTEM IDENTIFIED BY oracle
   LOGFILE GROUP 1 ('/u01/app/oracle/oradata/orcl/redo01.log')
             SIZE 100M BLOCKSIZE 512,
           GROUP 2 ('/u01/app/oracle/oradata/orcl/redo02.log')
             SIZE 100M BLOCKSIZE 512,
           GROUP 3 ('/u01/app/oracle/oradata/orcl/redo03.log')
             SIZE 100M BLOCKSIZE 512
   MAXLOGFILES 5
   MAXLOGMEMBERS 5
   MAXLOGHISTORY 1
   MAXDATAFILES 100
   MAXINSTANCES 1
   CHARACTER SET AL32UTF8
   NATIONAL CHARACTER SET AL16UTF16
   EXTENT MANAGEMENT LOCAL
   DATAFILE '/u01/app/oracle/oradata/orcl/system01.dbf'
     SIZE 50M REUSE AUTOEXTEND ON
   SYSAUX DATAFILE '/u01/app/oracle/oradata/orcl/sysaux01.dbf'
     SIZE 50M REUSE AUTOEXTEND ON
   DEFAULT TABLESPACE USERS
     DATAFILE '/u01/app/oracle/oradata/orcl/users01.dbf'
     SIZE 50M REUSE AUTOEXTEND ON
   DEFAULT TEMPORARY TABLESPACE TEMP
     TEMPFILE '/u01/app/oracle/oradata/orcl/temp01.dbf'
     SIZE 20M REUSE AUTOEXTEND ON
   UNDO TABLESPACE UNDOTBS1
     DATAFILE '/u01/app/oracle/oradata/orcl/undotbs01.dbf'
     SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

c) Create a password file in the `$ORACLE_HOME/dbs` directory.

```
$ cd $ORACLE_HOME/dbs
$ orapwd file=orapworcl password=oracle
```

d) Create necessary directories.

```
$ mkdir -p /u01/app/oracle/oradata/orcl
```

e) Run the preceding `CREATE DATABASE` statement by using SQL*Plus.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

SQL> startup nomount
ORACLE instance started.

Total System Global Area  835104768 bytes
```

```
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes

SQL> @/home/oracle/cre_db.sql

Database created.
```

f) Create dictionary views and standard PL/SQL packages by using the `catalog.sql` and `catproc.sql` scripts.

```
SQL> spool cat.log
SQL> @?/rdbms/admin/catalog.sql
SQL> @?/rdbms/admin/catproc.sql
```

g) Create the SQL*Plus `PRODUCT_USER_PROFILE` tables.

```
SQL> connect system/oracle
SQL> spool pupbld.log
SQL> @?/sqlplus/admin/pupbld.sql
SQL> spool off
```

2. Configure Enterprise Manager Database Control using Enterprise Manager Configuration Assistant (EMCA). The repository should already exist in the database in the `sysaux` tablespace.

   a) If you created the database manually, the repository does not exist in your database. Create the repository.

```
$ emca -config dbcontrol db –repos create
STARTED EMCA at Mar 07, 2013 11:18:20 AM
EM Configuration Assistant, Version 11.2.0.0.3 Production
Copyright (c) 2003, 20011, Oracle.  All rights reserved.

Enter the following information:
Database SID: orcl
Listener port number: 1521
Listener ORACLE_HOME
[ /u01/app/oracle/product/11.2.0/dbhome_1 ]:
Password for SYS user:oracle
Password for DBSNMP user:oracle
Mar 7, 2013 11:18:31 AM oracle.sysman.emcp.util.GeneralUtil
initSQLEngineLoacly
WARNING: ORA-28000: the account is locked
Password for SYSMAN user:oracle
Email address for notifications (optional):
Outgoing Mail (SMTP) server for notifications (optional):
----------------------------------------------------------------
-
```

```
You have specified the following settings

Database ORACLE_HOME ................
/u01/app/oracle/product/11.2.0/dbhome_1

Local hostname ................ edaor80p1.us.oracle.com
Listener ORACLE_HOME ................
/u01/app/oracle/product/11.2.0/dbhome_1
Listener port number ................ 1521
Database SID ................ orcl
Email address for notifications ...............
Outgoing Mail (SMTP) server for notifications ...............

-----------------------------------------------------------------
-
Do you wish to continue? [yes(Y)/no(N)]: Y
Mar 07, 2013  11:20:48 AM oracle.sysman.emcp.EMConfig perform
INFO: This operation is being logged at
/u01/app/oracle/cfgtoollogs/emca/orcl/emca_2013_03_07_16_56_31.l
og.
…………
Mar 7, 2013 11:25:23 AM oracle.sysman.emcp.EMDBPostConfig
performConfiguration
INFO: >>>>>>>>>>> The Database Control URL is
https://edaor80p1.us.oracle.com:1158/em <<<<<<<<<<<
Mar 7, 2013 11:25:31 AM oracle.sysman.emcp.EMDBPostConfig invoke
WARNING:
************************  WARNING  ************************

Management Repository has been placed in secure mode wherein
Enterprise Manager data will be encrypted.  The encryption key
has been placed in the file:
/u01/app/oracle/product/11.2.0/dbhome_1/edaor80p1.us.oracle.com_
orcl/sysman/config/emkey.ora. Ensure this file is backed up as
the encrypted data will become unusable if this file is lost.

***********************************************************
Enterprise Manager configuration completed successfully
FINISHED EMCA at Mar 7, 2013 11:25:31 AM
```

## Manual Solutions for Practice 1-3: Managing the Oracle Instance

**Background:** You have just installed the Oracle software and created a database. You want to ensure that you can start and stop the database and see the application data.

**Your Tasks**

1. View the initialization parameters of the `orcl` database. Set the `JOB_QUEUE_PROCESSES` parameter to `30`.

```
SQL> show parameter job

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
---------------
job_queue_processes                  integer     1000
SQL> alter system set job_queue_processes = 30;

System altered.
```

If you want to set the parameter permanently, you should set the parameter to your initialization parameter file.

```
job_queue_processes = 30
```

2. Shut down the database instance.

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

3. When the Status of the instance is Down, use SQL*Plus to verify that you are **not** able to connect as the `SYSTEM` user.

   a. Enter the following to attempt to log in to the database:

```
$ sqlplus system/oracle

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

ERROR:
ORA-01034: ORACLE not available
ORA-27101: shared memory realm does not exist
Linux Error: 2: No such file or directory
Process ID: 0
Session ID: 0 Serial number: 0


Enter user-name:
```

4. Restart the database instance by connecting the user SYS with SYSDBA.

```
$ sqlplus / as sysdba
SQL> startup
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
Database opened.
```

5. In the alert log, view the phases that the database went through during startup.

   a) Confirm the directory of diagnostic_dest using SQL*Plus.

```
SQL> show parameter diagnostic

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
---------------
diagnostic_dest                      string      /u01/app/oracle
```

   b) Change directory to diagnostic_dest. See the alert log by using any editor or viewer.

```
$ cd /u01/app/oracle/diag/rdbms/orcl/orcl/trace/
$ tail -n 20 alert_orcl.log
SMON: enabling cache recovery
Successfully onlined Undo Tablespace 2.
Verifying file header compatibility for 11g tablespace
encryption..
Verifying 11g file header compatibility for tablespace
encryption completed
SMON: enabling tx recovery
Database Characterset is AL32UTF8
No Resource Manager plan active
replication_dependency_tracking turned off (no async multimaster
replication found)
Starting background process QMNC
Thu Mar 07 12:19:12 2013
QMNC started with pid=18, OS id=12591
Completed: ALTER DATABASE OPEN
...
```

## Manual Solutions for Practice 1-4: Managing Undo Data

**Background:** In the orcl database, a new version of your application will include several reports based on very long-running queries. Configure your system to support these reports.

**Your Tasks**

1. Use the Undo Advisor to calculate the amount of undo space required to support a report that takes 60 minutes to run, on the basis of an analysis period of the last seven days.

   **See the GUI solution.**

2. Resize the undo tablespace to support the retention period required by the new reports (or 400 MB, whichever is smaller). Do this by increasing the size of the existing data file.

   a) Confirm the current size of the undo tablespace.

```
SQL> select file_name,bytes / 1024 / 1024 MB
  2  from dba_data_files;

FILE_NAME                                                  MB
-------------------------------------------------- ----------
/u01/app/oracle/oradata/orcl/system01.dbf                 325
/u01/app/oracle/oradata/orcl/undotbs01.dbf                200
/u01/app/oracle/oradata/orcl/sysaux01.dbf                 325
/u01/app/oracle/oradata/orcl/users01.dbf                    5
```

   b) If you increase the size of the tablespace to 400 MB, enter the following SQL by using SQL*Plus:

```
SQL> alter database datafile
  2  '/u01/app/oracle/oradata/orcl/undotbs01.dbf'
  3  resize 400M;

Database altered.
```

## Manual Solutions for Practice 1-5: Managing Database Storage Structures

**Background:** You need to create a new tablespace for the `INVENTORY` application in the `orcl` database. All scripts are located in the `/home/oracle/labs` directory.

**Your Tasks**

1. Create a new, locally managed tablespace called `INVENTORY`. Use the following specifications:

| **Object** | **Setting** |
|---|---|
| Tablespace name | `INVENTORY` |
| Extent Management | Locally Managed |
| Type | Permanent |
| Status | Read Write |
| Use Bigfile tablespace | *Deselected* |
| Data File Name | `inventory01.dbf` |
| File Size | `5 MB` |
| Autoextend | *Deselected* |
| Extent Allocation | Automatic |
| Segment Space Management | Automatic |
| Enable Logging | *Selected* |

   a) Enter the following SQL to create the `INVENTORY` tablespace by using SQL*Plus:

```
$ cd /home/oracle/labs
$ export ORACLE_SID=orcl
$ sqlplus system/oracle

SQL> create tablespace INVENTORY
  2    datafile '/u01/app/oracle/oradata/orcl/inventory01.dbf'
  3    size 5M autoextend off
  4    extent management local autoallocate
  5    segment space management auto
  6    logging;

Tablespace created.
```

2. Run the `lab_01_05_02.sql` script to create and populate a table (called `X`) in the `INVENTORY` tablespace. What error do you eventually see?

   a) Note that there is eventually an error ORA-01653 stating that the table cannot be extended. There is not enough space to accommodate all of the rows to be inserted.

```
SQL> @lab_01_05_02.sql
SQL> create table x (a char(1000)) tablespace inventory
  2  /
Table created.
```

```
SQL> insert into x values('a')
  2  /
1 row created.

SQL> insert into x select * from x
  2  /
1 row created.
…

SQL> insert into x select * from x
  2  /
insert into x select * from x
*
ERROR at line 1:
ORA-01653: unable to extend table SYS.X by 128 in tablespace
INVENTORY


SQL> commit
  2  /

Commit complete.
```

3. Define space for 50 MB in the tablespace instead of 5 MB, while keeping the same single data file in the tablespace. What is the `ALTER` statement that is executed to make this change?

   Enter the following SQL to resize the `INVENTORY` tablespace by using SQL*Plus:

```
SQL> alter database datafile
  2    '/u01/app/oracle/oradata/orcl/inventory01.dbf'
  3    resize 50M;

Database altered.
```

4. Run the `lab_01_05_04.sql` script that drops the table and re-executes the original script that previously returned the space error.

   a) Note that the same number of row inserts are attempted, and there is no error because of the increased size of the tablespace.

```
SQL> @lab_01_05_04.sql
SQL> set echo on
SQL> create table x (a char(1000)) tablespace inventory
  2  /

Table created.

SQL> insert into x values('a')
  2  /
1 row created.
```

```
SQL> insert into x select * from x
  2  /
1 row created.

…

SQL> insert into x select * from x
  2  /
1024 rows created.

SQL> insert into x select * from x
  2  /

2048 rows created.

SQL> commit
  2  /

Commit complete.
```

5. Create a new, bigfile tablespace called `BIGTBS`. Use the following specifications:

| **Object** | **Setting** |
|---|---|
| Tablespace name | `BIGTBS` |
| Extent Management | Locally Managed |
| Status | Read Write |
| Data File Name | `bigtbs.dbf` |
| File Size | `5 MB` |

   a) Enter the following SQL commands:

```
SQL> create bigfile tablespace BIGTBS
  2  datafile '/u01/app/oracle/oradata/orcl/bigtbs.dbf'
  3  size 5M;
```

6. Drop the `INVENTORY` tablespace and the `BIGTBS` tablespace.

   a) Enter the following SQL commands:

```
SQL> drop tablespace bigtbs
  2  including contents and datafiles;

Tablespace dropped.
SQL> drop tablespace inventory
  2  including contents and datafiles;

Tablespace dropped.
```

## Manual Solutions for Practice 1-6: Configuring the Oracle Network Environment

**Background:** You need to connect to the instructor's `orcl` database instance. Work with your instructor to enable connections by using different methods. Ensure that you can use connect-time failover to take advantage of a backup listener.

### Your Tasks

1. Make a copy of your `tnsnames.ora` file. It is in the `$ORACLE_HOME/network/admin` directory.

   a) Change directory and copy the `tnsnames.ora` file.

```
$ cd /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/
$ cp tnsnames.ora tnsnames.ora.org
```

2. Modify your local Names Resolution file so that you can connect to your instructor's `orcl` database instance. Name this connection **`testorcl`**.

| Object | Setting |
|---|---|
| Connection name | `testorcl` |
| DB ID Method | `SID` |
| SID value | `orcl` |
| Protocol | TCP/IP |
| Port | `1521` |
| Host | IP or fully qualified host name of your instructor's PC |

   Add the following entry to the `tnsnames.ora` file:

```
testorcl =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = <fully qualified host
name of your instructor's PC>)(PORT = 1521))
    (CONNECT_DATA =
      (SID = orcl)
    )
  )
```

3. Test your changes to the network configuration by using SQL*Plus. Use `system` as the username, `oracle` as the password, and `testorcl` as the connect string. To see information related to the instructor, select the `instance_name` and `host_name` columns from the `v$instance` table. You should see the instructor's host name.

   a) Access your partner's `orcl` database by using `testorcl`.

```
$ sqlplus system/oracle@testorcl

SQL> select instance_name,host_name from v$instance;
```

```
INSTANCE_NAME
----------------
HOST_NAME
----------------------------------------------------------------
orcl
edaor80p0.us.oracle.com
```

4. Create a `LISTENER2` listener to support connect-time failover. Use port 1561 for this listener. Use the **Static Database Registration** tab to connect the listener to your database. Use the following information:

| **Object** | **Setting** |
|---|---|
| Listener name | `LISTENER2` |
| Host | `<fully qualified host name of your odd PC>` |
| Service name | `orcl.oracle.com` |
| Protocol | TCP/IP |
| Port | `1561` |
| SID | `orcl` |
| Oracle Home Directory | `/u01/app/oracle/product/11.2.0/dbhome_1` |

a) Add the following entry to the `listener.ora` file:

```
LISTENER2 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = <fully qualified host
name of your odd PC>)(PORT = 1561))
    )
  )

SID_LIST_LISTENER2 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = orcl)
      (GLOBAL_DBNAME = orcl.oracle.com)
      (ORACLE_HOME = /u01/app/oracle/product/11.2.0/dbhome_1)
    )
  )
```

5. Start the `LISTENER2` listener.

```
$ lsnrctl start LISTENER2

LSNRCTL for Linux: Version 11.2.0.3.0 - Production on 07-MAR-
2013 18:12:40
Copyright (c) 1991, 2011, Oracle.  All rights reserved.
```

```
Starting /u01/app/oracle/product/11.2.0/dbhome_1/bin/tnslsnr:
please wait...

TNSLSNR for Linux: Version 11.2.0.3.0 - Production
System parameter file is
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/listener.o
ra
Listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=edaor80p1.us.oracle.co
m)(PORT=1561)))

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=edaor80p1.us.oracle.co
m)(PORT=1561)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER2
Version                   TNSLSNR for Linux: Version 11.2.0.3.0
- Production
Start Date                07-MAR-2013 18:12:40
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/listener.o
ra
Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=edaor80p1.us.oracle.co
m)(PORT=1561)))
Services Summary...
Service "orcl.oracle.com" has 1 instance(s).
  Instance "orcl", status UNKNOWN, has 1 handler(s) for this
service...
The command completed successfully
```

## Manual Solutions for Practice 1-7: Oracle Shared Server

You notice that your system is performing poorly during peak load times. After investigating, you find that user sessions are consuming so much memory that your system is swapping excessively. Configure your system to reduce the amount of memory that is consumed by user sessions.

**Tasks**

- Investigate the impact of dedicated server connections on your system.
- Configure your system to use shared servers.
- Investigate the impact of shared-server connections on your system.

1. Investigate the impact of dedicated server connections on your system.
   a) Open four terminal sessions on your server.
   b) Check to see how many Oracle processes are running. (Your answer will vary.)

```
$ ps -ef|grep oracle|wc -l
72
```

   c) Now start SQL*Plus sessions in your remaining three terminal sessions. Verify that none of your three SQL*Plus sessions are using shared servers.

```
$ sqlplus system/oracle@orcl
SQL> select username,server from v$session where username =
'SYSTEM';

USERNAME                       SERVER
------------------------------ ---------
SYSTEM                         DEDICATED
SYSTEM                         DEDICATED
SYSTEM                         DEDICATED
```

   d) Your default service name is configured to use dedicated server processes. Notice that two new processes are created with each SQL*Plus session that is started. Remember that each of your SQL*Plus sessions starts *two* resource-consuming processes: one is the SQL*Plus session itself (which will normally consume resources on the client or middle tier), and the other the dedicated server process (which consumes resources on the server).

```
$ ps -ef|grep oracle|wc -l

78
```

2. Configure your system to use shared servers with a minimum of two shared servers and the dispatchers attribute for TCP/IP should be set to a minimum of two dispatchers.

a) Confirm the current setting of `shared_servers` and `dispatchers` initialized parameters.

```
SQL> show parameter shared_servers

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
---------------
max_shared_servers                   integer
shared_servers                       integer     0

SQL> show parameter dispatchers

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
---------------
dispatchers                          string
max_dispatchers                      integer
```

b) Add the following parameters to the initialized parameter file for your `orcl` instance.

```
shared_servers = 2
dispatchers = '(PROTOCOL=TCP)(DISPATCHERS=2)'
```

c) Restart the `orcl` database instance on your odd PC.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

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

Total System Global Area  285212672 bytes
Fixed Size                  1260420 bytes
Variable Size              92275836 bytes
Database Buffers          188743680 bytes
Redo Buffers                2932736 bytes
Database mounted.
Database opened.
SQL> show parameter shared_servers

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
---------------
max_shared_servers                   integer
shared_servers                       integer     2
SQL> show parameter dispatchers
```

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
---------------
dispatchers                          string
(PROTOCOL=TCP)(DISPATCHERS=2)
max_dispatchers                      integer
```

3. Investigate the impact of shared server connections on your system.
   a) Exit your three SQL*Plus sessions.
   b) Check to see how many Oracle processes are running. (Your answer will vary.)

```
$ ps -ef|grep oracle|wc -l
70
```

   c) Reconnect to your instance in the three terminal windows.

```
$ sqlplus system/oracle@<hostname>:1521/orcl.oracle.com
SQL*Plus: Release 11.2.0.3.0 Production on Fri Mar 1 13:45:12
2013
Copyright (c) 1982, 2009, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 -
ProductionWith the Partitioning, Oracle Label Security, OLAP,
Data Mining
and Real Application Testing options
SQL>
```

   d) The connection shown in the preceding step uses EZConnect to connect to the shared server.
   e) Verify that you are connected using a shared server:

```
SQL> select username,server from v$session where username =
'SYSTEM';

USERNAME                       SERVER
------------------------------ ---------
SYSTEM                         NONE
SYSTEM                         NONE
SYSTEM                         SHARED
```

   f) Count the number of Oracle processes. Notice that your three SQL*Plus sessions started only three new processes. (These are the three SQL*Plus processes.) All three of these would normally have been located on the client machine or application server, which means that these three new sessions would not have added *any* new processes on the server.

```
$ ps -ef|grep oracle|wc –l
73
```

4. Configure two additional local naming methods for the `orcl` database. The `shared_orcl` alias always uses a shared server connection and the `orcl` alias always uses a dedicated server connection.

   a) Update the `tnsnames.ora` file in the `$ORACLE_HOME/network/admin` directory. The `tnsnames.ora` file includes the following:

```
orcl =
  (DESCRIPTION =
    (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = <fully qualified host name of your odd PC>)
        (PORT = 1521)
    )
    (CONNECT_DATA =
      (SERVER = dedicated)
      (SERVICE_NAME = orcl.oracle.com)
    )
  )

sharedorcl =
  (DESCRIPTION =
    (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = <fully qualified host name of your odd PC>)
        (PORT = 1521)
    )
    (CONNECT_DATA =
      (SERVER = shared)
      (SERVICE_NAME = orcl.oracle.com)
    )
  )
```

## Manual Solutions for Practice 1-8: Using the Password Security Feature

You decide to increase the security of your passwords by enforcing case sensitivity for the password for privileged users.

If there is no HR schema on the orcl database, run the HR.sh script that is located at /home/oracle/labs. The script will create the HR schema.

```
$ cd /home/oracle/labs
$ ./HR.sh
```

1. Confirm the Password Case Sensitivity settings for the instance for database users. View the value of the SEC_CASE_SENSITIVE_LOGON parameter. Connect to the orcl instance as HR with hr as the password. Attempt to connect with the username and password combinations: HR/hr, hr/hr, and hr/HR.
   a) Connect to SQL*Plus as sysdba and check the SEC_CASE_SENSITIVE_LOGON parameter:

```
$ sqlplus / as sysdba

SQL> show parameter sec_case_sensitive_logon

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------
sec_case_sensitive_logon             boolean     TRUE
SQL> exit
$
```

   b) Start SQL*Plus with the /nolog option. Attempt to connect with the various combinations of username and password. Observe which combinations allow a connection.
   **Note:** Be sure to use a valid connect string before you make five failed attempts, or the account will be locked for five minutes.

```
$ sqlplus /nolog

SQL> connect HR/HR
ERROR:
ORA-01017: invalid username/password; logon denied

SQL> connect HR/hr
Connected.
SQL> connect hr/hr
Connected.
SQL> connect hr/HR
ERROR:
ORA-01017: invalid username/password; logon denied

Warning: You are no longer connected to ORACLE.
```

```
SQL> exit
$
```

2. Make the password file non-case-sensitive.

```
$ rm $ORACLE_HOME/dbs/orapworcl
$ orapwd file=$ORACLE_HOME/dbs/orapworcl entries=5 password=oracle
ignorecase=Y force=Y
$
```

3. Confirm the Password Case Sensitivity settings for the instance for privileged users. Attempt to connect to the `orcl` instance as the `SYSDBA` user using a net service name and:
   - A lowercase password
   - An uppercase password
   - A mixed-case password

```
$ sqlplus /nolog

SQL> connect sys/oracle@orcl AS SYSDBA
Connected.
SQL> connect sys/ORACLE@orcl AS SYSDBA
Connected.
SQL> connect sys/Oracle@orcl AS SYSDBA
Connected.
SQL> exit
$
```

4. Make the password file case-sensitive to match the database password usage.

```
$ rm $ORACLE_HOME/dbs/orapworcl
$ orapwd file=$ORACLE_HOME/dbs/orapworcl entries=5 password=oracle
ignorecase=N force=Y
$
```

## Manual Solutions for Practice 2-1: Creating the Recovery Catalog and Registering the Database

1. Create a tablespace for the recovery catalog and a recovery catalog owner in your `PROD1` database. The tablespace name is `RCTS`. The username is `RCUSER` with `oracle` as password. The user is granted a privilege of recovery catalog owner.

```
$ export ORACLE_SID=PROD1
$ sqlplus / as sysdba

SQL> create tablespace RCTS
  2  datafile '/u01/app/oracle/oradata/PROD1/rcts01.dbf'
  3  size 10M autoextend on;

Tablespace created.

SQL> create user RCUSER identified by oracle
  2  default tablespace RCTS
  3  quota unlimited on RCTS;

User created.

SQL> grant RECOVERY_CATALOG_OWNER to RCUSER;

Grand succeeded.
```

2. Connect to the recovery catalog database (`PROD1`) with the appropriate recovery catalog owner name (`RCUSER`) using RMAN. Create the recovery catalog in the `RCTS` tablespace.

```
$ rman catalog rcuser/oracle@PROD1

Recovery Manager: Release 11.2.0.3.0 - Production on Thu Mar 7
14:07:56 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights
reserved.
connected to recovery catalog database

RMAN> create catalog;

recovery catalog created
```

3. Using RMAN, connect to your target database `orcl` and the recovery catalog database `PROD1`.

   **Answer:**

```
$ export ORACLE_SID=orcl
$ rman target / catalog rcuser/oracle@PROD1
```

```
Recovery Manager: Release 11.2.0.3.0 - Production on Thu Mar 7
14:09:23 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights
reserved.

connected to target database: ORCL (DBID=1262929399)
connected to recovery catalog database
```

4. Using RMAN, execute the `resync catalog` command to resynchronize the control file and the recovery catalog. What happens? Why?

   **Answer: The target database is not yet registered in the recovery catalog, so the `resync` command fails.**

```
RMAN> resync catalog;

RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
===============
RMAN-00571:
===========================================================
RMAN-03002: failure of resync command at 11/18/2010 11:10:34
RMAN-06004: ORACLE error from recovery catalog database: RMAN-20001:
target database not found in recovery catalog
```

5. Register the target database in the recovery catalog.

```
RMAN> register database;

database registered in recovery catalog
starting full resync of recovery catalog
full resync complete

RMAN> list incarnation;
List of Database Incarnations
DB Key  Inc Key DB Name  DB ID            STATUS  Reset SCN  Reset Time
------- ------- -------- -------------- --- ---------- ----------
2       21      ORCL     1262929399     PARENT  1          15-AUG-09
2       4       ORCL     1262929399     CURRENT 945184     15-NOV-10
```

6. You can use Enterprise Manager Grid Control to create the recovery catalog and register the database. (If you manually create the recovery catalog and manually register the database, skip this step.)
   Select **Recovery Catalog Settings** on the Maintenance page. Select **Add Recovery Catalog** on the Recovery Catalog Settings page.

   **See the GUI solution.**

7. Create an RMAN script named `whole_backup` to make a whole database backup. **Do not execute the `whole_backup` script at this time.**

   **Answer:**

   a) Using RMAN, connect to your target database and the recovery catalog.

```
$ rman target / catalog rcuser/oracle@PROD1


connected to target database: ORCL (DBID=1262929399)
connected to recovery catalog database
```

   b) Create the `whole_backup` script by using the `CREATE SCRIPT` command.

```
RMAN> create script whole_backup {
2> backup database;
3> }

created script whole_backup
```

8. Use the `PRINT` command to query the recovery catalog and verify the creation of your `whole_backup` script.

```
RMAN> list script names;

List of Stored Scripts in Recovery Catalog


    Scripts of Target Database ORCL

       Script Name
       Description
       ----------------------------------------------------------------------
---------
       whole_backup

RMAN> print script whole_backup;

printing stored script: whole_backup
 {backup database;
}
```

## Manual Solutions for Practice 2-2: Configuring Your Database

1. Configure your database (`ORCL`) in `ARCHIVELOG` mode.

   **Answer:**

```
SQL> archive log list
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     9
Current log sequence           11
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL> alter database archivelog;

Database altered.

SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     9
Next log sequence to archive   11
Current log sequence           11

SQL> alter database open;

Database altered.
```

2. Verify that the Fast Recovery Area has been configured for your database and increase the Fast Recovery Area size to 12 GB.

   **Answer:**

```
SQL> show parameter recovery_file_dest
NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------------
----------
db_recovery_file_dest                string
/u01/app/oracle/fast_recovery_area
```

```
db_recovery_file_dest_size           big integer 10G
SQL> alter system set db_recovery_file_dest_size = 12G;
System altered.
```

3. Set Preferred Credentials in Enterprise Manager.

   **See the GUI solution.**

4. Use Recovery Manager (RMAN) to connect to your target database. Make a note of the database identifier (DBID) of your database.

   Database Identifier: ___________________________

   **Answer:**

   a) Open a terminal window and log in as `oracle/oracle`.

   b) Start RMAN and connect to the target database by entering the following command at the operating system prompt: `rman target /`

```
$ export ORACLE_SID=orcl
$ rman target /

Recovery Manager: Release 11.2.0.3.0 - Production on Thu Mar 7
14:23:45 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights
reserved.

connected to target database: ORCL (DBID=1262929399)
RMAN>
```

   Make a note of the database identifier (DBID) of your database.

5. Use the RMAN `SHOW ALL` command to view the configuration settings in your database, and then exit from your RMAN session.

   **Answer:**

```
RMAN> show all;

using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; #
default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; #
default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE
FOR LOAD TRUE ; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/u01/app/oracle/product/11.2.0/dbhome_1/dbs/snapcf_orcl.f'; # default


RMAN> exit
Recovery Manager complete.
```

## Manual Solutions for Practice 2-3: Using RMAN to Create and Manage Backups

1. For the `orcl` database, configure autobackup of the control file and the server parameter file.

```
$ rman target / catalog rcuser/oracle@PROD1

RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;

new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete
```

2. Configure backup optimization and enable block change tracking. Specify **`/u01/app/oracle/oradata/orcl/chg_track.f`** for the name of the block change tracking file.

   **Answer:**

```
RMAN> CONFIGURE BACKUP OPTIMIZATION ON;

new RMAN configuration parameters:
CONFIGURE BACKUP OPTIMIZATION ON;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete

RMAN> exit
Recovery Manager complete.

$ sqlplus / as sysdba

SQL> alter database enable block change tracking using file
'/u01/app/oracle/oradata/orcl/chg_track.f';

Database altered.
```

3. Create a whole database backup as the base backup of incremental backup using the Oracle-suggested backup strategy.

```
$ rman target / catalog rcuser/oracle@PROD1

connected to target database: ORCL (DBID=1262929399)
connected to recovery catalog database

RMAN> backup incremental level 0 database;

Starting backup at 18-NOV-10
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=23 device type=DISK
channel ORA_DISK_1: starting incremental level 0 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001
name=/u01/app/oracle/oradata/orcl/system01.dbf
input datafile file number=00002
name=/u01/app/oracle/oradata/orcl/sysaux01.dbf
input datafile file number=00003
name=/u01/app/oracle/oradata/orcl/undotbs01.dbf
input datafile file number=00005
name=/u01/app/oracle/oradata/orcl/example01.dbf
input datafile file number=00004
name=/u01/app/oracle/oradata/orcl/users01.dbf
channel ORA_DISK_1: starting piece 1 at 18-NOV-10
channel ORA_DISK_1: finished piece 1 at 18-NOV-10
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/backupset/2010_11_18/o
1_mf_nnnd0_TAG20101118T194128_6gc0c93h_.bkp tag=TAG20101118T194128
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:25
Finished backup at 18-NOV-10

Starting Control File and SPFILE Autobackup at 18-NOV-10
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/autobackup/2010_11_18/
o1_mf_s_735421314_6gc0d2sf_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 18-NOV-10
```

4. View information about your backups.

```
RMAN> list backup;

List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
490     Incr 0  1.19G      DISK        00:00:16     18-NOV-10
        BP Key: 491   Status: AVAILABLE  Compressed: NO  Tag:
TAG20101118T194324
        Piece Name:
/u01/app/oracle/fast_recovery_area/ORCL/backupset/2010_11_18/o1_mf_nn
nd0_TAG20101118T194324_6gc0gw71_.bkp
  List of Datafiles in backup set 490
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1    0  Incr 1449171    18-NOV-10
/u01/app/oracle/oradata/orcl/system01.dbf
  2    0  Incr 1449171    18-NOV-10
/u01/app/oracle/oradata/orcl/sysaux01.dbf
  3    0  Incr 1449171    18-NOV-10
/u01/app/oracle/oradata/orcl/undotbs01.dbf
  4    0  Incr 1449171    18-NOV-10
/u01/app/oracle/oradata/orcl/users01.dbf
```

```
  5    0  Incr 1449171    18-NOV-10
/u01/app/oracle/oradata/orcl/example01.dbf

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
510     Full    9.36M      DISK        00:00:01     18-NOV-10
        BP Key: 516   Status: AVAILABLE  Compressed: NO  Tag:
TAG20101118T194349
        Piece Name:
/u01/app/oracle/fast_recovery_area/ORCL/autobackup/2010_11_18/o1_mf_s
_735421429_6gc0howl_.bkp
  SPFILE Included: Modification time: 18-NOV-10
  SPFILE db_unique_name: ORCL
  Control File Included: Ckp SCN: 1449204      Ckp time: 18-NOV-10
```

5. Create an archival backup of the `orcl` database for Long-Term storage using the tag (long01) into the `/home/oracle/my-files/backup` directory.

   **Answer:**

   a) Create a directory and start RMAN.

```
$ mkdir /home/oracle/my-files/backup
$ rman target / catalog rcuser/oracle@PROD1
```

   b) Configure channel for archival backup and create an archival backup.

```
RMAN> CONFIGURE CHANNEL DEVICE TYPE DISK format '/home/oracle/my-
files/backup/%U';
RMAN> backup database keep forever tag long01;

Starting backup at 07-MAR-13
current log archived

allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=141 device type=DISK
backup will never be obsolete
archived logs required to recover from this backup will be backed up
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00003
name=/u01/app/oracle/oradata/orcl/undotbs01.dbf
input datafile file number=00001 name=/u01/app/oracle input datafile
file number=00002
name=/u01/app/oracle/oradata/orcl/sysaux01.dbf/oradata/orcl/system01.
dbf
input datafile file number=00004
name=/u01/app/oracle/oradata/orcl/users01.dbf
channel ORA_DISK_1: starting piece 1 at 07-MAR-13
channel ORA_DISK_1: finished piece 1 at 07-MAR-13
piece handle=/home/oracle/backup/05o3ucol_1_1 tag=TAG20130307T144332
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting incremental level 0 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
channel ORA_DISK_1: starting piece 1 at 07-MAR-13
channel ORA_DISK_1: finished piece 1 at 07-MAR-13
piece handle=/home/oracle/my-files/backup/06o3ucp4_1_1
tag=TAG20130307T144332 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 07-MAR-13
```

c) Use the RMAN `LIST BACKUP SUMMARY` command to view a list of backup files. You can find "LONG01" for Tag name. Delete the current settings for the channel.

```
RMAN> list backup summary;

List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies
Compressed Tag
------- -- -- - ----------- --------------- ------- ------- ---------
- ---
51      B  F  A DISK        07-MAR-13       1       1       NO
LONG01
65      B  A  A DISK        07-MAR-13       1       1       NO
LONG01
80      B  F  A DISK        07-MAR-13       1       1       NO
LONG01
RMAN> CONFIGURE CHANNEL DEVICE TYPE DISK clear;

old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT   '/home/oracle/my-
files/backup/%U';
old RMAN configuration parameters are successfully deleted
released channel: ORA_DISK_1
released channel: ORA_DISK_1
full resync complete

RMAN> exit

Recovery Manager complete.
```

## Manual Solutions for Practice 2-4: Using RMAN to Recover a Data File

In this practice, you use RMAN to recover a lost data file for `orcl`.

1. Use SQL*Plus to query the `HR.REGIONS` table. Make a note of the number of rows in the `HR.REGIONS` table.

   **Answer:**

   a) Open a terminal window and log in to SQL*Plus and connect as the `HR` user with `hr` as the password.

   b) Query the `HR.REGIONS` table and record the number of rows.

   Number of rows: ___________

```
$ sqlplus hr/hr

SQL> select * from regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
SQL> exit
```

   c) Exit from your SQL*Plus session.

2. At the operating system prompt, execute the `lab_02_04_02.sh` script located in `/home/oracle/labs` to simulate a failure in your database. This script deletes the `EXAMPLE` tablespace data file.

   **Answer:**

   a) At the operating system prompt, change to the `labs` directory and execute the `lab_02_04_02.sh` script.

```
$ cd /home/oracle/labs
$ ./lab_02_04_02.sh

EXAMPLE tablespace file deleted
```

3. Use SQL*Plus to query the `HR.JOBS` table.

   **Answer:**

   a) Log in to SQL*Plus and connect as the `HR` user with `hr` as the password.

   b) Query the `HR.JOBS` table.

```
SQL> select * from jobs;
select * from jobs
```

```
               *
ERROR at line 1:
ORA-01116: error in opening database file 5
ORA-01110: data file 5: '/u01/app/oracle/oradata/orcl/example01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

4. Perform database recovery of the `EXAMPLE` tablespace data file.
   a) Use Recovery Manager to manually restore and recover the data file.

```
$ rman target / catalog rcuser/oracle@PROD1

connected to target database: ORCL (DBID=1262929399)
connected to recovery catalog database
RMAN> list failure;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
202        HIGH     OPEN      18-NOV-10     One or more non-system
datafiles are missing

RMAN> advise failure;

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
202        HIGH     OPEN      18-NOV-10     One or more non-system
datafiles are missing

analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=25 device type=DISK
analyzing automatic repair options complete

Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
=======================
1. If file /u01/app/oracle/oradata/orcl/example01.dbf was
unintentionally renamed or moved, restore it
Automated Repair Options
========================
Option Repair Description
------ ------------------
```

```
1      Restore and recover datafile 5
  Strategy: The repair includes complete media recovery with no data
loss
  Repair script:
/u01/app/oracle/diag/rdbms/orcl/orcl/hm/reco_854265368.hm

RMAN> repair failure;

Strategy: The repair includes complete media recovery with no data
loss
Repair script:
/u01/app/oracle/diag/rdbms/orcl/orcl/hm/reco_854265368.hm

contents of repair script:
   # restore and recover datafile
   sql 'alter database datafile 5 offline';
   restore datafile 5;
   recover datafile 5;
   sql 'alter database datafile 5 online';

Do you really want to execute the above repair (enter YES or NO)? YES
executing repair script

sql statement: alter database datafile 5 offline

Starting restore at 18-NOV-10
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00005 to
/u01/app/oracle/oradata/orcl/example01.dbf
channel ORA_DISK_1: reading from backup piece
/u01/app/oracle/fast_recovery_area/ORCL/backupset/2010_11_18/o1_mf_nn
nd0_TAG20101118T194324_6gc0gw71_.bkp
channel ORA_DISK_1: piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/backupset/2010_11_18/o
1_mf_nnnd0_TAG20101118T194324_6gc0gw71_.bkp tag=TAG20101118T194324
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
Finished restore at 18-NOV-10

Starting recover at 18-NOV-10
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:00

Finished recover at 18-NOV-10

sql statement: alter database datafile 5 online
repair failure complete
```

5. Return to your SQL*Plus session and again attempt to query the `HR.JOBS` table.

**Answer:**

a) Query the HR.JOBS table.

```
SQL> select * from hr.jobs;

JOB_ID     JOB_TITLE                           MIN_SALARY MAX_SALARY
---------- ----------------------------------- ---------- ----------
AD_PRES    President                                20000      40000
AD_VP      Administration Vice President            15000      30000
AD_ASST    Administration Assistant                  3000       6000
FI_MGR     Finance Manager                           8200      16000
…
MK_REP     Marketing Representative                  4000       9000
HR_REP     Human Resources Representative            4000       9000
PR_REP     Public Relations Representative           4500      10500

19 rows selected.
```

## Manual Solutions for Practice 2-5: Recovering Control Files

In this practice, you recover your control file by using autobackup.

1. Use SQL*Plus to view information about the control files in the `orcl` database. Query `V$CONTROLFILE`.

   **Answer:**

   a) Log in to SQL*Plus as `SYSDBA`.

   b) Query the `NAME` column in `V$CONTROLFILE`. Exit from SQL*Plus.

```
SQL> SELECT name FROM v$controlfile;

NAME
---------------------------------------------
/u01/app/oracle/oradata/orcl/control01.ctl
/u01/app/oracle/oradata/orcl/control02.ctl
/u01/app/oracle/oradata/orcl/control03.ctl
```

2. Simulate a failure in your environment by executing the `lab_02_05_02.sh` script that is located in `/home/oracle/labs` to delete all your control files.

   **Answer:**

   a) In your terminal window session, change to the `labs` directory and execute the `lab_02_05_02.sh` script.

```
$ cd /home/oracle/labs
$ ./lab_02_05_02.sh

Control files deleted
```

3. You need some more information about your control files. Query `V$CONTROLFILE_RECORD_SECTION` to learn more about the contents of your control file.

   **Answer:**

   a) Log in to SQL*Plus as `SYSDBA`.

   b) Query the `V$CONTROLFILE_RECORD_SECTION` view.

```
SQL> SELECT * FROM v$controlfile_record_section;
SELECT * FROM v$controlfile_record_section
              *
ERROR at line 1:
ORA-00210: cannot open the specified control file
ORA-00202: control file: '/u01/app/oracle/oradata/orcl/control01.ctl'
ORA-27041: unable to open file
Linux Error: 2: No such file or directory
Additional information: 3
```

4. You have lost all your control files and will need to recover them from the control file autobackup. Use Recovery Manager to recover the control files.

   **Answer:**

   a) Use SQL*Plus to shut down your instance. Exit from your SQL*Plus session.

```
$ sqlplus / as sysdba

SQL> shutdown abort
ORACLE instance shut down.
SQL> exit
```

   b) Use RMAN to connect to your target database.

```
$ rman target /
connected to target database (not started)

RMAN>
```

   c) Restart the instance in `NOMOUNT` mode.

```
RMAN> startup nomount

Oracle instance started

Total System Global Area     835104768 bytes

Fixed Size                     2217952 bytes
Variable Size                671090720 bytes
Database Buffers             155189248 bytes
Redo Buffers                   6606848 bytes
```

   d) Restore the control file from the autobackup.

```
RMAN> restore controlfile from autobackup;

Starting restore at 18-NOV-10
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=134 device type=DISK

recovery area destination: /u01/app/oracle/fast_recovery_area
database name (or database unique name) used for search: ORCL
channel ORA_DISK_1: AUTOBACKUP
/u01/app/oracle/fast_recovery_area/ORCL/autobackup/2010_11_18/o1_mf_s
_735400224_6gbcs068_.bkp found in the recovery area
AUTOBACKUP search with format "%F" not attempted because DBID was not
set
channel ORA_DISK_1: restoring control file from AUTOBACKUP
/u01/app/oracle/fast_recovery_area/ORCL/autobackup/2010_11_18/o1_mf_
s_735400224_6gbcs068_.bkp
channel ORA_DISK_1: control file restore from AUTOBACKUP complete
output file name=/u01/app/oracle/oradata/orcl/control01.ctl
output file
name=/u01/app/oracle/fast_recovery_area/orcl/control02.ctl
Finished restore at 18-NOV-10
```

e) Mount the database.

```
RMAN> alter database mount;

database mounted
released channel: ORA_DISK_1
```

f) Recover the database by issuing the `RECOVER DATABASE` command.

```
RMAN> recover database;

Starting recover at 18-NOV-10
Starting implicit crosscheck backup at 18-NOV-10
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=134 device type=DISK
Crosschecked 4 objects
Finished implicit crosscheck backup at 18-NOV-10

Starting implicit crosscheck copy at 18-NOV-10
using channel ORA_DISK_1
Crosschecked 5 objects
Finished implicit crosscheck copy at 18-NOV-10

searching for all files in the recovery area
cataloging files...
cataloging done

List of Cataloged Files
=======================
File Name:
/u01/app/oracle/fast_recovery_area/ORCL/autobackup/2010_11_18/o1_mf_s
_735400224_6gbcs068_.bkp

using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 21 is already on disk as file
/u01/app/oracle/oradata/orcl/redo03.log
archived log file name=/u01/app/oracle/oradata/orcl/redo03.log
thread=1 sequence=21
media recovery complete, elapsed time: 00:00:00
Finished recover at 18-NOV-10
```

g) Open the database with the `RESETLOGS` option.

```
RMAN> alter database open resetlogs;

database opened
```

h) Back up the current online redo log file and back up all the archived redo log files.

```
RMAN> SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';

sql statement: ALTER SYSTEM ARCHIVE LOG CURRENT
RMAN> backup archivelog all;

Starting backup at 18-NOV-10
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=19 RECID=4 STAMP=735402403
input archived log thread=1 sequence=20 RECID=5 STAMP=735402404
input archived log thread=1 sequence=21 RECID=6 STAMP=735402404
channel ORA_DISK_1: starting piece 1 at 18-NOV-10
channel ORA_DISK_1: finished piece 1 at 18-NOV-10
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/backupset/2010_11_18/o
1_mf_annnn_TAG20101118T142732_6gbfynf3_.bkp tag=TAG20101118T142732
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=1 RECID=7 STAMP=735402421
input archived log thread=1 sequence=2 RECID=8 STAMP=735402452
channel ORA_DISK_1: starting piece 1 at 18-NOV-10
channel ORA_DISK_1: finished piece 1 at 18-NOV-10
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/backupset/2010_11_18/o
1_mf_annnn_TAG20101118T142732_6gbfyqkn_.bkp tag=TAG20101118T142732
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 18-NOV-10

Starting Control File and SPFILE Autobackup at 18-NOV-10
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/autobackup/2010_11_18/
o1_mf_s_735402456_6gbfyrq3_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 18-NOV-10
```

i) Create a whole database backup.

```
RMAN> backup database;
Starting backup at 18-NOV-10
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001
name=/u01/app/oracle/oradata/orcl/system01.dbf
input datafile file number=00002
name=/u01/app/oracle/oradata/orcl/sysaux01.dbf
```

```
input datafile file number=00003
name=/u01/app/oracle/oradata/orcl/undotbs01.dbf
input datafile file number=00005
name=/u01/app/oracle/oradata/orcl/example01.dbf
input datafile file number=00004
name=/u01/app/oracle/oradata/orcl/users01.dbf
channel ORA_DISK_1: starting piece 1 at 18-NOV-10
channel ORA_DISK_1: finished piece 1 at 18-NOV-10
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/backupset/2010_11_18/o
1_mf_nnndf_TAG20101118T142811_6gbfzvwq_.bkp tag=TAG20101118T142811
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:25
Finished backup at 18-NOV-10

Starting Control File and SPFILE Autobackup at 18-NOV-10
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/autobackup/2010_11_18/
o1_mf_s_735402516_6gbg0nwt_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 18-NOV-10
```

## Manual Solutions for Practice 2-6: Enabling Flashback Database

1. Enable Flashback Database for `orcl`.

   **Answer:**

   a) Use SQL*Plus to manually enable Flashback Database.

```
SQL> select FLASHBACK_ON from v$database;

FLASHBACK_ON
------------------
NO

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             671090720 bytes
Database Buffers          155189248 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL> alter database flashback on;

Database altered.

SQL> select FLASHBACK_ON from v$database;

FLASHBACK_ON
------------------
YES

SQL> alter database open;

Database altered.
```

2. Use the `ALTER DATABASE` command to enable supplemental logging.

   **Answer:**

   a) In a terminal window, log in to SQL*Plus as `SYSDBA`.

```
$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.3.0 Production on Thu Mar 7 15:49:25 2013

Copyright (c) 1982, 2011, Oracle.  All rights reserved.
```

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
```

b) Execute the `ALTER DATABASE` command to enable supplemental logging.

```

SQL> ALTER DATABASE add supplemental log data;

Database altered.
```

## Manual Solutions for Practice 2-7: Using Flashback Database

1. In orcl, verify that Flashback Database is enabled.

   **Answer:**

   a) Use SQL*Plus to manually verify that Flashback Database is enabled.

```
SQL> select FLASHBACK_ON from v$database;

FLASHBACK_ON
------------------
YES
```

2. Create a Guaranteed Restore Point named BEFORE_TRUNCATE.

   **Answer:**

   a) Use SQL*Plus to manually create a Guaranteed Restore Point.

```
SQL> create restore point BEFORE_TRUNCATE guarantee flashback
database;

Restore point created.

SQL> SELECT NAME, GUARANTEE_FLASHBACK_DATABASE FROM V$RESTORE_POINT;

NAME                           GUA
------------------------------ ---
BEFORE_TRUNCATE                YES
```

3. Determine the number of rows in the HR.JOB_HISTORY table. Record the number of rows: __________

   **Answer:**

   a) Determine the number of rows in the HR.JOB_HISTORY table.

```
SQL> SELECT count(*)
  2  FROM hr.job_history
  3  /

  COUNT(*)
----------
        10
```

4. Truncate the HR.JOB_HISTORY table.

   **Answer:**

   a) Enter the following SQL.

```
SQL> truncate table hr.job_history;

Table truncated.
```

5. Determine the number of rows in the HR.JOB_HISTORY table.

   **Answer:**

   a) Determine the number of rows in the HR.JOB_HISTORY table.

```
SQL> SELECT count(*) FROM hr.job_history;

  COUNT(*)
----------
         0
```

6. Use Flashback Database to restore the HR.JOB_HISTORY table rows.

   **Answer:**

   a) Use Flashback Database to restore the HR.JOB_HISTORY table rows by using SQL*Plus.

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             671090720 bytes
Database Buffers          155189248 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL> flashback database to restore point BEFORE_TRUNCATE;

Flashback complete.

SQL> alter database open resetlogs;

Database altered.
```

7. Return to your SQL*Plus session. Query the HR.JOB_HISTORY table again to be sure that the data has been restored. For cleanup, drop the restore point BEFORE_TRUNCATE.

   **Answer:**

   a) Invoke SQL*Plus and log in as SYS/oracle as SYSDBA.

   b) Query the HR.JOB_HISTORY table. At last, drop the restore point.

```
SQL> SELECT count(*)
  2  FROM hr.job_history
  3  /
  COUNT(*)
----------
        10
```

```
SQL> drop restore point BEFORE_TRUNCATE;

Restore point dropped.
```

## Manual Solutions for Practice 3-1: Using External Tables

If there is no SH schema on the orcl database, run the SH.sh script that is located in /home/oracle/labs. The script will create the SH schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. This practice will demonstrate the use of external tables to load data into a data warehouse. The table will be called sales_delta_XT and it will use data contained in the salesDec01.dat file, which is located in the /home/oracle/labs directory. Before you can create an external table, you will need to create a directory object in the database that points to the directory on the file system where the data files reside. Optionally, you can separate the location for the logfile, badfile, and discarded files from the location of the data files. Create the directory objects and the grant access privilege for the directories to the SH user.

```
SQL> connect system/oracle
Connected.
SQL> CREATE DIRECTORY data_dir AS '/home/oracle/labs';

Directory created.

SQL> CREATE DIRECTORY log_dir AS '/tmp';

Directory created.

SQL> GRANT ALL ON DIRECTORY data_dir TO sh;

Grant succeeded.

SQL> GRANT ALL ON DIRECTORY log_dir TO sh;

Grant succeeded.
```

2. When creating an external table, you are defining two parts of information:
   - The metadata information for the table representation inside the database
   - The HOW access parameter definition to extract the data from the external file

   After the creation of this meta information, the external data can be accessed from within the database, without the necessity of an initial load.

Execute the following statements as the SH user to create the external table:

```
CREATE TABLE sales_delta_XT (
   PROD_ID NUMBER,
   CUST_ID NUMBER,
   TIME_ID DATE,
   CHANNEL_ID CHAR(2),
   PROMO_ID NUMBER,
   QUANTITY_SOLD NUMBER(3),
   AMOUNT_SOLD NUMBER(10,2)
   )
   ORGANIZATION external (
   TYPE oracle_loader
     DEFAULT DIRECTORY data_dir
     ACCESS PARAMETERS  (
       RECORDS DELIMITED BY NEWLINE
       BADFILE log_dir:'sh_sales.bad'
       LOGFILE log_dir:'sh_sales.log_xt'
       FIELDS TERMINATED BY "|" (
          prod_id, cust_id,
          time_id CHAR(11) DATE_FORMAT DATE MASK "DD-MON-YYYY",
          channel_id, promo_id, quantity_sold, amount_sold)
       )
       location('salesDec01.dat')
   );
```

```
SQL> connect sh/sh
Connected.
SQL> CREATE TABLE sales_delta_XT (
  2     PROD_ID NUMBER,
  3     CUST_ID NUMBER,
  4     TIME_ID DATE,
  5     CHANNEL_ID CHAR(2),
  6     PROMO_ID NUMBER,
  7     QUANTITY_SOLD NUMBER(3),
  8     AMOUNT_SOLD NUMBER(10,2)
  9     )
 10     ORGANIZATION external (
 11       DEFAULT DIRECTORY data_dir
 12       ACCESS PARAMETERS  (
 13         RECORDS DELIMITED BY NEWLINE
 14         BADFILE log_dir:'sh_sales.bad'
 15         LOGFILE log_dir:'sh_sales.log_xt'
 16         FIELDS TERMINATED BY "|" (
 17            prod_id, cust_id,
 18            time_id CHAR(11) DATE_FORMAT DATE MASK "DD-MON-YYYY",
 19            channel_id, promo_id, quantity_sold, amount_sold)
 20         )
 21         location('salesDec01.dat')
 22     );

Table created.
```

3. The data in the external file can now be accessed without any further action. To demonstrate this capability, execute the following SQL commands:

```
SELECT COUNT(*) FROM sales_delta_xt;
SELECT MAX(time_id) FROM sales_delta_xt;
```

```
SQL> SELECT COUNT(*) FROM sales_delta_xt;

  COUNT(*)
----------
     79130

SQL> SELECT MAX(time_id) FROM sales_delta_xt;

MAX(TIME_
---------
31-DEC-01
```

4. Load the data serially from the `sales_delta_xt` external table into the `SALES` fact table. Roll back the operation when you have finished. Execute the following SQL statements:

```
INSERT /*+ APPEND */ INTO sales (
  PROD_ID, CUST_ID, TIME_ID, CHANNEL_ID,
  PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD
  )
  SELECT PROD_ID, CUST_ID, TIME_ID,
    case CHANNEL_ID
      when 'S' then 3
      when 'T' then 9
      when 'C' then 5
      when 'I' then 4
      else 2
    end,
    PROMO_ID, sum(QUANTITY_SOLD), sum(AMOUNT_SOLD)
    FROM SALES_DELTA_XT
    GROUP BY prod_id,time_id,cust_id,channel_id,promo_id;
```

```
SQL> INSERT /*+ APPEND */ INTO sales (
  2    PROD_ID, CUST_ID, TIME_ID, CHANNEL_ID,
  3    PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD
  4    )
  5    SELECT PROD_ID, CUST_ID, TIME_ID,
  6      case CHANNEL_ID
  7        when 'S' then 3
  8        when 'T' then 9
  9        when 'C' then 5
 10        when 'I' then 4
 11        else 2
 12      end,
 13      PROMO_ID, sum(QUANTITY_SOLD), sum(AMOUNT_SOLD)
 14      FROM SALES_DELTA_XT
 15      GROUP BY prod_id,time_id,cust_id,channel_id,promo_id;

75985 rows created.

SQL> rollback;

Rollback complete.
```

5. Using the Data Pump driver, create an external table called `sales_ch` from the `SALES` table that contains all the records that have a `channel_id` of 4. Use the same `data_dir` directory that you created in step 1. Name the Data Pump output file that will be created as `'sales_ch.exp'`. Execute the following SQL statements to create the external table:

```
CREATE TABLE sales_ch
  ORGANIZATION external (
    TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY data_dir
    LOCATION ('sales_ch.exp')
  )
  AS SELECT * from sales
       WHERE channel_id = '4';
```

Make sure that you can access the `sales_ch` table. Look at the `sales_ch.exp` file that is being used by the new table (and created in the process).

```
SQL> CREATE TABLE sales_ch
  2    ORGANIZATION external (
  3      TYPE ORACLE_DATAPUMP
  4      DEFAULT DIRECTORY data_dir
  5      LOCATION ('sales_ch.exp')
  6    )
  7    AS SELECT * from sales
  8         WHERE channel_id = '4';

Table created.

SQL> select count(*) from sales_ch;

  COUNT(*)
----------
    118416

SQL> !ls -al /home/oracle/labs/sales_ch.exp

-rw-r-----  1 oracle oinstall 4018176 Jan 22 11:40
/home/oracle/labs/sales_ch.exp
```

6. Drop the `data_dir` and `log_dir` directories.

```
SQL> connect system/oracle
Connected.
SQL> drop directory data_dir;

Directory dropped.

SQL> drop directory log_dir;

Directory dropped.
```

## Manual Solutions for Practice 3-2: Moving Data

If there is no HR schema on the orcl database, run the HR.sh script that is located in /home/oracle/labs. The script will create the HR schema.

```
$ cd /home/oracle/labs
$ ./HR.sh
```

**Background:** In the recent past, you received a number of questions about the HR schema. To analyze them, without interfering with the daily activities, you decide to use the Data Pump Wizard to export the HR schema to file. When you perform the export, you are not sure into which database you will be importing this schema.

In the end, you find out that the only database for which the management approves an import is the orcl database. So you perform the import with the Data Pump Wizard, remapping the HR schema to a newly created HR_TEST schema in the HR_TEST tablespace.

1. Create an HR_TEST tablespace and an HR_TEST user by using SQL*Plus.

```
HR_TEST tablespace:
  DATAFILE : /u01/app/oracle/oradata/orcl/hr_test01.dbf
  SIZE : 10MB

HR_TEST user:
  DEFAULT TABLESPACE : HR_TEST
  QUOTA : UNLIMITED on HR_TEST
  PRIVIREGE : CREATE SESSION
```

```
$ sqlplus system/oracle

SQL> create tablespace hr_test
  2  datafile '/u01/app/oracle/oradata/orcl/hr_test01.dbf' size 10M;

Tablespace created.

SQL> create user hr_test identified by oracle
  2    default tablespace hr_test
  3    quota unlimited on hr_test;

User created.

SQL> grant create session to hr_test;

Grant succeeded.
```

2. Export the HR schema from the orcl database.

   a) Export the HR schema by using the Datapump Export utility.

```
$ expdp system/oracle schemas=HR directory=DATA_PUMP_DIR
dumpfile=HR.dmp

Export: Release 11.2.0.3.0 - Production on Thu Nov 25 13:45:01 2012

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights
reserved.

Connected to: Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 -  Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
Starting "SYSTEM"."SYS_EXPORT_SCHEMA_01":  system/******** schemas=HR
directory=DATA_PUMP_DIR dumpfile=HR.dmp
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 448 KB
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/PROCEDURE/PROCEDURE
Processing object type SCHEMA_EXPORT/PROCEDURE/ALTER_PROCEDURE
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/TRIGGER
Processing object type
SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
. . exported "HR"."COUNTRIES"                            6.367 KB
25 rows
. . exported "HR"."DEPARTMENTS"                          7.007 KB
27 rows
. . exported "HR"."EMPLOYEES"                            16.80 KB
107 rows
. . exported "HR"."JOBS"                                 6.984 KB
19 rows
. . exported "HR"."JOB_HISTORY"                          7.054 KB
10 rows
. . exported "HR"."LOCATIONS"                            8.273 KB
23 rows
. . exported "HR"."REGIONS"                              5.476 KB
4 rows
Master table "SYSTEM"."SYS_EXPORT_SCHEMA_01" successfully
loaded/unloaded
```

```
**********************************************************************
*********
Dump file set for SYSTEM.SYS_EXPORT_SCHEMA_01 is:
  /u01/app/oracle/product/11.2.0/dbhome_1/rdbms/log/HR.dmp
Job "SYSTEM"."SYS_EXPORT_SCHEMA_01" successfully completed at
13:45:18
```

b) Check the Export dump file.

```
SQL> select DIRECTORY_PATH from dba_directories where DIRECTORY_NAME
= 'DATA_PUMP_DIR';

DIRECTORY_PATH
--------------------------------------------------------------------
/u01/app/oracle/product/11.2.0/dbhome_1/rdbms/log/

SQL> !ls -l /u01/app/oracle/product/11.2.0/dbhome_1/rdbms/log
total 436
-rw-r--r-- 1 oracle oinstall   2452 Nov 25 13:45 export.log
-rw-r----- 1 oracle oinstall 499712 Nov 25 13:45 HR.dmp
```

3. As the `SYSTEM` user, import the exported `HR` schema back into the `orcl` database, remapping it to the previously created `HR_TEST` schema and `HR_TEST` tablespace.

   a) Confirm the name of the tablespace in which the objects of the `HR` schema are stored. Import the exported `HR` schema by using the Datapump Import utility.

```
$ sqlplus / as sysdba

SQL> select distinct(TABLESPACE_NAME)from dba_segments where
owner='HR';

TABLESPACE_NAME
--------------------------------------------------------------
EXAMPLE

SQL> exit

$ impdp system/oracle remap_schema=HR:HR_TEST
remap_tablespace=example:hr_test directory=DATA_PUMP_DIR
dumpfile=HR.dmp

Import: Release 11.2.0.3.0 - Production on Thu Mar 7 18:23:04 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights
reserved.

Connected to: Oracle Database 11g Enterprise Edition Release
11.2.0.3.0  Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
```

```
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully
loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01":  system/********
remap_schema=HR:HR_TEST directory=DATA_PUMP_DIR dumpfile=HR.dmp
Processing object type SCHEMA_EXPORT/USER
ORA-31684: Object type USER:"HR_TEST" already exists
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
. . imported "HR_TEST"."COUNTRIES"                       6.367 KB
25 rows
. . imported "HR_TEST"."DEPARTMENTS"                     7.007 KB
27 rows
. . imported "HR_TEST"."EMPLOYEES"                       16.80 KB
107 rows
. . imported "HR_TEST"."JOBS"                            6.984 KB
19 rows
. . imported "HR_TEST"."JOB_HISTORY"                     7.054 KB
10 rows
. . imported "HR_TEST"."LOCATIONS"                       8.273 KB
23 rows
. . imported "HR_TEST"."REGIONS"                         5.476 KB
4 rows
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/PROCEDURE/PROCEDURE
Processing object type SCHEMA_EXPORT/PROCEDURE/ALTER_PROCEDURE
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/TRIGGER
Processing object type
SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Job "SYSTEM"."SYS_IMPORT_FULL_01" completed with 1 error(s) at
14:05:50
```

b) Using SQL*Plus, connect to the database as `system` and confirm the tablespace name of objects owned by `HR_TEST` user.

```
$ sqlplus system/oracle

SQL> select distinct(TABLESPACE_NAME) from dba_segments where
owner='HR_TEST';

TABLESPACE_NAME
```

```
-------------------------------------------------------
HR_TEST
```

c) Connect as the `HR_TEST` user, and select data from tables in the `hr_test` schema, for verification of the import.

```
SQL> connect hr_test/oracle

SQL> select * from jobs;
```

## Manual Solutions for Practice 3-3: Materialized Views

If there is no `SH` schema on the `orcl` database, run the `SH.sh` script that is located in `/home/oracle/labs`. The script will create the `SH` schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. On the `orcl` database, connect as `SH` with the password `sh` and estimate the number of rows that a materialized view corresponding to the following query would contain:

```
SELECT c.cust_id, SUM(amount_sold) AS dollar_sales,
COUNT(amount_sold) AS cnt_dollars, COUNT(*) AS cnt
FROM   sales s, customers c
WHERE  s.cust_id= c.cust_id
GROUP  BY c.cust_id;
```

```
SQL> connect sh/sh
Connected.
SQL>
SQL> set serveroutput on;
SQL>
SQL> DECLARE
  2   no_of_rows NUMBER;
  3   mv_size NUMBER;
  4  BEGIN
  5   DBMS_MVIEW.estimate_mview_size('MV1',
  6         'SELECT c.cust_id, SUM(amount_sold) AS dollar_sales,
  7               COUNT(amount_sold) AS cnt_dollars, COUNT(*) AS cnt
  8          FROM sales s, customers c
  9          WHERE s.cust_id= c.cust_id
 10          GROUP BY c.cust_id' , no_of_rows, mv_size);
 11   DBMS_OUTPUT.put_line ( '');
 12   DBMS_OUTPUT.put_line ( 'Complete MV');
 13   DBMS_OUTPUT.put_line ( 'No of Rows: ' || no_of_rows );
 14   DBMS_OUTPUT.put_line ( 'MV Size: ' || mv_size);
 15   DBMS_OUTPUT.put_line ( '');
 16  END;
 17  /
Complete MV
No of Rows: 7059
MV Size: 621192

PL/SQL procedure successfully completed.

SQL>
```

2. Execute the following SQL to create a table called CUST_ID_SALES_AGGR. Compare the estimated size that you obtained in step 1 with the actual number of rows in the existing CUST_ID_SALES_AGGR table. This table contains the data corresponding to the query in step 1. What is your conclusion?

```
CREATE TABLE cust_id_sales_aggr
AS
SELECT c.cust_id,
       SUM(amount_sold) AS dollar_sales,
       COUNT(amount_sold) AS cnt_dollars,
       COUNT(*) AS cnt
FROM sales s, customers c
WHERE s.cust_id= c.cust_id
GROUP BY c.cust_id
/
```

**Answer:** The estimated size from the ESTIMATE_MVIEW_SIZE procedure is accurate, provided that you have up-to-date statistics on the relevant objects.

```
SQL> connect sh/sh
Connected.
SQL> CREATE TABLE cust_id_sales_aggr
  2  AS
  3  SELECT c.cust_id,
  4         SUM(amount_sold) AS dollar_sales,
  5         COUNT(amount_sold) AS cnt_dollars,
  6         COUNT(*) AS cnt
  7  FROM sales s, customers c
  8  WHERE s.cust_id= c.cust_id
  9  GROUP BY c.cust_id
 10  /
Table created.

SQL> SELECT COUNT(*)
  2  FROM cust_id_sales_aggr;
COUNT(*)
----------
      7059
```

3. Determine the number of objects in the SH schema.

```
SQL> SELECT COUNT(*)
  2  FROM user_objects;

  COUNT(*)
----------
       298
```

4. Assume that the CUST_ID_SALES_AGGR table indeed corresponds with the result of the query in step 1. Use this knowledge to create a materialized view on the prebuilt CUST_ID_SALES_AGGR table. Make sure that this materialized view can be used for future query rewrites and is fast refreshable on demand.

```
SQL> DROP MATERIALIZED VIEW cust_id_sales_aggr;
DROP MATERIALIZED VIEW cust_id_sales_aggr
*
ERROR at line 1:
ORA-12003: materialized view "SH"."CUST_ID_SALES_AGGR" does not exist
SQL> CREATE MATERIALIZED VIEW cust_id_sales_aggr
  2  ON PREBUILT TABLE
  3  REFRESH FORCE
  4  ENABLE QUERY REWRITE
  5  AS
  6  SELECT c.cust_id, SUM(amount_sold) AS dollar_sales,
  7         COUNT(amount_sold) AS cnt_dollars, COUNT(*) AS cnt
  8  FROM sales s, customers c
  9  WHERE s.cust_id= c.cust_id
 10  GROUP BY c.cust_id;

Materialized view created.
```

5. Count the number of objects in the SH schema again, to check for new objects. From the data dictionary, identify the objects having CUST_ID_SALES_AGGR as name. What type of objects are these? What is your conclusion?

**Answer:** One additional object is created for the materialized view. Because the materialized view is defined on a prebuilt table, a container table is not needed. Note that both objects have the same name.

```
SQL> SELECT COUNT(*) FROM user_objects;

  COUNT(*)
----------
       299


SQL> SELECT object_type
  2  FROM user_objects
  3  WHERE object_name='CUST_ID_SALES_AGGR';

OBJECT_TYPE
-------------------
TABLE
MATERIALIZED VIEW

SQL>
```

6. Create a new materialized view named `MV1`. This materialized view should be defined exactly the same way as `CUST_ID_SALES_AGGR` (created in the previous step), except that `MV1` should not be defined on the prebuilt `CUST_ID_SALES_AGGR` table, and its `SELECT` list should not contain the following two expressions: `COUNT(amount_sold)` and `COUNT(*)`. Count the number of objects in the `SH` schema again. What new objects have been created? What is your conclusion?

   **Answer:** The new materialized view is not defined on a prebuilt table. A container table and an index are created to enhance fast refresh performance.

```
SQL> DROP MATERIALIZED VIEW mv1;
DROP MATERIALIZED VIEW mv1
*
ERROR at line 1:
ORA-12003: materialized view "SH"."MV1" does not exist
SQL> CREATE MATERIALIZED VIEW mv1
  2  REFRESH FORCE
  3  ENABLE QUERY REWRITE
  4  AS
  5  SELECT c.cust_id, SUM(amount_sold) AS dollar_sales
  6  FROM sales s, customers c
  7  WHERE s.cust_id= c.cust_id
  8  GROUP BY c.cust_id;

Materialized view created.
SQL>
SQL> SELECT COUNT(*) FROM user_objects;

  COUNT(*)
----------
       302

SQL> SELECT SUBSTR(object_name,1,30) Name, object_type
  2  FROM user_objects
  3  WHERE object_name LIKE '%MV1%';

NAME                           OBJECT_TYPE
------------------------------ -------------------
I_SNAP$_MV1                    INDEX
MV1                            TABLE
MV1                            MATERIALIZED VIEW
SQL>
```

7. Compare the staleness status of the materialized views that you have created. What is your conclusion?

   **Answer:** When you create a materialized view on a prebuilt table, the contents of the table are not validated. Oracle does not guarantee that the contents are fresh.

```
SQL> SELECT SUBSTR(mview_name,1,30) mviewname, staleness
  2  FROM user_mviews
  3  WHERE mview_name IN ('MV1','CUST_ID_SALES_AGGR');

MVIEWNAME                      STALENESS
------------------------------ -------------------
CUST_ID_SALES_AGGR             UNKNOWN
MV1                            FRESH
```

8. Execute the `lab_03_03_08.sql` script. This script adds one row to the `CUSTOMERS` table. Check the staleness status and the compile status of both the materialized views.

   **Answer:** The `COMPILE_STATE` of both the materialized views is `NEEDS_COMPILE`. This status is also reflected in the `STALENESS` column.

```
SQL> @lab_03_03_08.sql
SQL> INSERT INTO customers VALUES
  2  (5000000,'John','Vance','M',1944,'married','77 Bradford Avenue',
  3  '52773','Santos',52293,'Sao Paulo',52735,52775,'129-379-
7148','G: 1,
  4  3000,'JohnVance@company.com','Customer total',52772,
0,sysdate,'' ,;
1 row created.

SQL>
SQL> COMMIT;

Commit complete.
SQL>
SQL> SELECT SUBSTR(mview_name,1,30), staleness, compile_state
  2  FROM user_mviews
  3  WHERE mview_name IN ('MV1','CUST_ID_SALES_AGGR');

SUBSTR(MVIEW_NAME,1,30)        STALENESS           COMPILE_STATE
------------------------------ ------------------- -----------------
CUST_ID_SALES_AGGR             NEEDS_COMPILE       NEEDS_COMPILE
MV1                            NEEDS_COMPILE       NEEDS_COMPILE
```

9. How would you refresh the staleness status of both the materialized views to reflect their current status? When refreshed, check the status. What happens, and why?

   **Answer:** You must recompile both views to see an accurate `STALENESS` status.

```
SQL> ALTER MATERIALIZED VIEW mv1 COMPILE;

Materialized view altered.

SQL>
SQL> ALTER MATERIALIZED VIEW cust_id_sales_aggr COMPILE;

Materialized view altered.
SQL> SELECT SUBSTR(mview_name,1,30), staleness, compile_state
  2  FROM user_mviews
  3  WHERE mview_name IN ('MV1','CUST_ID_SALES_AGGR');

SUBSTR(MVIEW_NAME,1,30)        STALENESS           COMPILE_STATE
------------------------------ ------------------- -------------------
CUST_ID_SALES_AGGR             STALE               VALID
MV1                            STALE               VALID
```

## Manual Solutions for Practice 3-4: Using SQL*Loader

If there is no SH schema on the orcl database, run the SH.sh script that is located in /home/oracle/labs. The script will create the SH schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

The salesDec01.dat file that is located in /home/oracle/labs has been extracted from an OLTP database and needs to be loaded into the data warehouse. You need a transformation, because the data is not in the right format. The QUANTITY_SOLD and AMOUNT_SOLD columns must be summed, grouping on all other columns, before loading the data into the target table.

Because SQL*Loader cannot sum values while loading, the data must first be loaded into a staging table, and then transformed while being inserted into the target table.

1. Connect to SQL*Plus as the SH user and create a staging table SALES_DEC01 to load the data into. The table must have the same structure as the SALES table.
   **Hint:** Use the clause WHERE 1=0.
   Modify the channel_id column to accept CHAR data because the initial data that you will load is not in exactly the same format as the SALES fact table.

```
CREATE TABLE sales_dec01 AS
SELECT *
FROM   sales
WHERE  1=0;

ALTER TABLE sales_dec01 MODIFY (channel_id CHAR(2) null);
```

```
SQL> connect sh/sh
Connected.
SQL> CREATE TABLE sales_dec01 AS
2     SELECT *
3     FROM   sales
4     WHERE  1=0;
Table created.
SQL> select count(*) from sales_dec01;
        0
SQL> ALTER TABLE sales_dec01 MODIFY (channel_id CHAR(2) null);
Table altered.
```

2. Load data from the salesDec01.dat file into the SALES_DEC01 staging table, using the sales_dec01.ctl control file that is located in /home/oracle/labs. Verify the information in the control file before loading.

```
$ cd /home/oracle/labs
$ sqlldr sh/sh control=sales_dec01.ctl direct=true
log=sales_dec01.log

SQL*Loader: Release 11.2.0.3.0 - Production on Thu Mar 7 19:02:38
2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights
reserved.

Load completed - logical record count 79130.
```

3. Load data from the `SALES_DEC01` staging table into the `SALES` target table. The data needs to be transformed; the `QUANTITY_SOLD` and `AMOUNT_SOLD` columns must be summed, grouping on all other columns. In addition, the `CHAR` values for `channel_id` must be converted to `NUMBER` as is the same column in the `SALES` table. As the `SH` user, execute the following SQL statements to perform this task:

```
INSERT /*+ APPEND */ INTO sales
  ( PROD_ID, CUST_ID, TIME_ID, CHANNEL_ID,
    PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD )
  SELECT
    PROD_ID, CUST_ID, TIME_ID,
    case CHANNEL_ID
    when 'S' then 3
    when 'T' then 9
    when 'C' then 5
    when 'I' then 4
    else 2
    end,
    PROMO_ID,
    sum(QUANTITY_SOLD),
    sum(AMOUNT_SOLD)
  FROM sales_dec01
  GROUP BY prod_id,time_id,cust_id,channel_id,promo_id;
```

```
$ sqlplus sh/sh

SQL> INSERT /*+ APPEND */ INTO sales
  2    ( PROD_ID, CUST_ID, TIME_ID, CHANNEL_ID,
  3      PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD )
  4    SELECT
  5      PROD_ID, CUST_ID, TIME_ID,
  6      case CHANNEL_ID
  7      when 'S' then 3
  8      when 'T' then 9
  9      when 'C' then 5
 10      when 'I' then 4
 11      else 2
 12      end,
 13      PROMO_ID,
 14      sum(QUANTITY_SOLD),
 15      sum(AMOUNT_SOLD)
 16    FROM sales_dec01
 17    GROUP BY prod_id,time_id,cust_id,channel_id,promo_id;

75985 rows created.
```

4. When you have successfully loaded the data into the `SALES` table, drop the `sales_dec01` staging table.

```
SQL> drop table sales_dec01;
```

## Manual Solutions for Practice 3-5: Loading Data from Transportable Tablespaces

If there is no SH schema on the orcl database, run the SH.sh script that is located in /home/oracle/labs. The script will create the SH schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. Assume that you want to move the January 2000 sales data from your data warehouse to the SALES table in your data mart. The data must be placed into a separate tablespace in order to be transported. So, create a tablespace called tt_temp_sales to hold the January sales data. After creating the tablespace, create a table called temp_jan_sales by using the CREATE TABLE ... AS SELECT statement. Execute the following CREATE TABLESPACE statements as the SYSTEM user and execute the following CREATE TABLE statements as the SH user on the orcl database:

```
CREATE TABLESPACE tt_temp_sales DATAFILE '/tmp/tt_temp_sales.dbf'
SIZE 30M REUSE autoextend on;

CREATE TABLE temp_jan_sales NOLOGGING TABLESPACE tt_temp_sales
AS SELECT * FROM sales
WHERE time_id BETWEEN '31-DEC-1999' AND '01-FEB-2000';
```

```
$ export ORACLE_SID=orcl
$ sqlplus system/oracle

SQL> CREATE TABLESPACE tt_temp_sales DATAFILE
'/tmp/tt_temp_sales.dbf' SIZE 30M REUSE autoextend on;

Tablespace created.

SQL> connect sh/sh
Connected.
SQL> CREATE TABLE temp_jan_sales NOLOGGING TABLESPACE tt_temp_sales
  2  AS SELECT * FROM sales
  3  WHERE time_id BETWEEN '31-DEC-1999' AND '01-FEB-2000';

Table created.
```

2. To prevent any changes to the tt_temp_sales tablespace, set it to read-only by executing the following SQL statement:

```
ALTER TABLESPACE tt_temp_sales READ ONLY;
```

```
SQL> connect system/oracle
SQL> ALTER TABLESPACE tt_temp_sales READ ONLY;
Tablespace altered.
SQL> exit
```

3. Create a directory called `/home/oracle/orcl` and a directory object called `orcl_dir` that points to `/home/oracle/orcl` where the log and dump files for the Data Pump export will be written. Next, use the `expdp` utility as shown as follows to export the `tt_transfer` tablespace metadata:

```
CREATE DIRECTORY orcl_dir as '/home/oracle/orcl';

expdp system/oracle DIRECTORY=orcl_dir
DUMPFILE=meta_tt_temp_sales.dmp TRANSPORT_TABLESPACES=tt_temp_sales
```

```
$ mkdir /home/oracle/orcl
$ sqlplus system/oracle

SQL> CREATE DIRECTORY orcl_dir as '/home/oracle/orcl';

Directory created.

SQL> exit;

$ expdp system/oracle DIRECTORY=orcl_dir
DUMPFILE=meta_tt_temp_sales.dmp TRANSPORT_TABLESPACES=tt_temp_sales

Connected to: Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 -  Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
Starting "SYSTEM"."SYS_EXPORT_TRANSPORTABLE_01":  system/********
DIRECTORY=orcl_dir DUMPFILE=meta_tt_temp_sales.dmp
TRANSPORT_TABLESPACES=tt_temp_sales
Processing object type TRANSPORTABLE_EXPORT/PLUGTS_BLK
Processing object type TRANSPORTABLE_EXPORT/TABLE
Processing object type TRANSPORTABLE_EXPORT/POST_INSTANCE/PLUGTS_BLK
Master table "SYSTEM"."SYS_EXPORT_TRANSPORTABLE_01" successfully
loaded/unloaded
*********************************************************************
*********
Dump file set for SYSTEM.SYS_EXPORT_TRANSPORTABLE_01 is:
  /home/oracle/orcl/meta_tt_temp_sales.dmp
*********************************************************************
*********
Datafiles required for transportable tablespace TT_TEMP_SALES:
  /tmp/tt_temp_sales.dbf
Job "SYSTEM"."SYS_EXPORT_TRANSPORTABLE_01" successfully completed at
16:25:33
```

4. You will transport the tablespace to the `PROD1` database. Create the `/home/oracle/PROD1` directory and copy the data file and the dump file to the directory. Verify that the files have been successfully copied.

```
$ mkdir /home/oracle/PROD1
$ cp /tmp/tt_temp_sales.dbf /home/oracle/PROD1
$ cp /home/oracle/orcl/meta_tt_temp_sales.dmp /home/oracle/PROD1

$ ls -l /home/oracle/PROD1
total 30860
-rw-r----- 1 oracle oinstall    94208 Nov 25 16:29
meta_tt_temp_sales.dmp
-rw-r----- 1 oracle oinstall 31465472 Nov 25 16:28 tt_temp_sales.dbf
```

5. Set the `tt_temp_sales` tablespace to read-write by executing the following SQL statement on the `orcl` database:

```
ALTER TABLESPACE tt_temp_sales READ WRITE;
```

```
$ sqlplus system/oracle

SQL> ALTER TABLESPACE tt_temp_sales READ WRITE;

Tablespace altered.

SQL> exit
```

6. On the `PROD1` database, create a directory called `PROD1_dir` that points to `/home/oracle/PROD1` that will be used by the Data Pump import, and create a user named `SH` by using the following SQL statement:

```
CREATE USER sh IDENTIFIED BY sh;
GRANT create session TO sh;
```

When this is done, use the `impdp` utility as shown as follows to make the `tt_temp_sales` tablespace accessible to the `PROD1` database.

```
impdp system/oracle DIRECTORY=PROD1_dir
DUMPFILE=meta_tt_temp_sales.dmp logfile=imp_tt.log
transport_datafiles=/home/oracle/PROD1/tt_temp_sales.dbf
```

```
$ export ORACLE_SID=PROD1
$ sqlplus system/oracle

SQL> create directory PROD1_dir as '/home/oracle/PROD1';

Directory created.

SQL> CREATE USER sh IDENTIFIED BY sh;
```

```
User created.

SQL> GRANT create session TO sh;

Grant succeeded.
SQL> exit;

$ impdp system/oracle DIRECTORY=PROD1_dir
DUMPFILE=meta_tt_temp_sales.dmp logfile=imp_tt.log
transport_datafiles=/home/oracle/PROD1/tt_temp_sales.dbf

Import: Release 11.2.0.3.0 - Production on Thu Mar 7 19:07:00 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights
reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
Master table "SYSTEM"."SYS_IMPORT_TRANSPORTABLE_01" successfully
loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_TRANSPORTABLE_01":  system/********
directory=PROD1_dir dumpfile=meta_tt_temp_sales.dmp logfile=imp_tt.log
transport_datafiles=/home/oracle/PROD1/tt_temp_sales.dbf
Processing object type TRANSPORTABLE_EXPORT/PLUGTS_BLK
Processing object type TRANSPORTABLE_EXPORT/TABLE
Processing object type TRANSPORTABLE_EXPORT/POST_INSTANCE/PLUGTS_BLK
Job "SYSTEM"."SYS_IMPORT_TRANSPORTABLE_01" successfully completed at
16:37:37
```

7. When the import is finished, verify that the `temp_jan_sales` table is accessible:

```
$ export ORACLE_SID=PROD1
$ sqlplus sh/sh

SQL> select count(*) from temp_jan_sales;

  COUNT(*)
----------
     22594
```

8. Drop the `tt_temp_sales` tablespace and the `orcl_dir` directory from the `orcl` database.

```
$ export ORACLE_SID=orcl
$ sqlplus system/oracle

SQL> drop tablespace tt_temp_sales including contents and datafiles;

Tablespace dropped.
```

```
SQL> drop DIRECTORY orcl_dir;

Directory dropped.
```

9. Drop the `tt_temp_sales` tablespace and the `PROD1_dir` directory from the `PROD1` database.

```
$ export ORACLE_SID=PROD1
$ sqlplus system/oracle

SQL> drop tablespace tt_temp_sales including contents and datafiles;

Tablespace dropped.

SQL> drop DIRECTORY PROD1_dir;

Directory dropped.
```

## Manual Solution for Practice 3-6: Automatic Parallelism Feature

1. In the `orcl` database, verify the current value of the initialization parameters that are used for parallel execution.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

SQL> set pagesize 9999
SQL> col type for a10
SQL> col value for a10
SQL> show parameter parallel

NAME                                 TYPE       VALUE
------------------------------------ ---------- --------------------
fast_start_parallel_rollback         string     LOW

parallel_adaptive_multi_user         boolean    TRUE
parallel_automatic_tuning            boolean    FALSE
parallel_degree_limit                string     CPU
parallel_degree_policy               string     MANUAL
parallel_execution_message_size      integer    16384
parallel_force_local                 boolean    FALSE
parallel_instance_group              string
parallel_io_cap_enabled              boolean    FALSE
parallel_max_servers                 integer    80
parallel_min_percent                 integer    0
parallel_min_servers                 integer    0
parallel_min_time_threshold          string     AUTO
parallel_server                      boolean    FALSE
parallel_server_instances            integer    1
parallel_servers_target              integer    32
parallel_threads_per_cpu             integer    2
recovery_parallelism                 integer    0
```

2. Enable automatic parallelism for this session.

```
SQL> alter session set parallel_degree_policy=auto;

Session altered.
```

3. Tune the parameter that is to be executed in serial if the computed elapsed time is below 20 seconds in this session.

```
SQL> alter session set parallel_min_time_threshold=20;

Session altered.
```

4. Verify the current value of the initialization parameters that are used for parallel execution.

```
SQL> show parameter parallel_degree_policy

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------------
----------
parallel_degree_policy               string      AUTO

SQL> show parameter parallel_min_time_threshold

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------------
----------
parallel_min_time_threshold          string      20
```

## Manual Solutions for Practice 3-7: Enabling Parallel DML

1. In the `orcl` database, connect as the `SH` user and create the partitioned table `LITTLE_SALES` by executing the following SQL statements:

```
CREATE TABLE little_sales
PARTITION BY HASH (time_id)
(PARTITION LS1,PARTITION LS2)
PARALLEL
AS
SELECT * FROM sales WHERE 1=2;
```

Note that the `LITTLE_SALES` table has only two partitions and the dictionary degree of parallelism (DOP) is set to `DEFAULT`. Execute the following SQL statements to insert some rows into the `LITTLE_SALES` table:

```
INSERT INTO little_sales
SELECT *
FROM sales
WHERE ROWNUM<5000;
commit;
```

Is the `INSERT` statement executed in parallel? You can confirm it by using the `V$PQ_SESSTAT` view.

**Answer:** A DML statement can be parallelized only if you specifically issue an `ALTER SESSION` statement to enable parallel DML:

```
SQL> connect sh/sh
SQL> CREATE TABLE little_sales
  2  PARTITION BY HASH (time_id)
  3  (PARTITION LS1,PARTITION LS2)
  4  PARALLEL
  5  AS
  6  SELECT * FROM sales WHERE 1=2;

Table created.

SQL> INSERT INTO little_sales
  2  SELECT *
  3  FROM sales
  4  WHERE ROWNUM<5000;

4999 rows created.

SQL> commit;

Commit complete.
```

```
SQL> select * from v$pq_sesstat where STATISTIC = 'Allocation
Height';

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Allocation Height                       0             0
```

2. Enable parallel DML in your session. Execute the INSERT command again and confirm output of selecting the v$pq_sesstat view.

```
SQL> ALTER SESSION ENABLE PARALLEL DML;

Session altered.

SQL> INSERT INTO little_sales
  2  SELECT *
  3  FROM sales
  4  WHERE ROWNUM<5000;

4999 rows created.

SQL> commit;

Commit complete.

SQL> select * from v$pq_sesstat where STATISTIC = 'Allocation
Height';

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Allocation Height                       4             0

SQL> commit;

Commit complete.
```

3. Drop the LITTLE_SALES table.

```
SQL> DROP  TABLE little_sales;
Table dropped.
```

## Manual Solutions for Practice 3-8: Managing Processes for Parallel SQL Execution

1. In the `orcl` database, connect as `SYSDBA` and query the current values of the initialization parameters that are used for parallel execution and that are of interest to you. Execute the following SQL statements:

```
SELECT name, value FROM v$parameter
WHERE UPPER(name) IN ('LARGE_POOL_SIZE','PROCESSES','SESSIONS',
'PARALLEL_EXECUTION_MESSAGE_SIZE','PARALLEL_MIN_SERVERS',
'PARALLEL_MAX_SERVERS','PARALLEL_ADAPTIVE_MULTI_USER',
'PARALLEL_MIN_PERCENT');
```

```
SQL> connect / as sysdba
Connected.
SQL> COL name FORMAT A35
SQL> COL value FORMAT A15
SQL> SELECT name, value FROM v$parameter
  2  WHERE UPPER(name) IN ('LARGE_POOL_SIZE','PROCESSES','SESSIONS',
  3  'PARALLEL_EXECUTION_MESSAGE_SIZE','PARALLEL_MIN_SERVERS',
  4  'PARALLEL_MAX_SERVERS','PARALLEL_ADAPTIVE_MULTI_USER',
  5  'PARALLEL_MIN_PERCENT');

NAME                                VALUE
----------------------------------- ---------------
processes                           150
sessions                            248
large_pool_size                     0
parallel_min_percent                0
parallel_min_servers                0
parallel_max_servers                80
parallel_execution_message_size     16384
parallel_adaptive_multi_user        TRUE

8 rows selected.
```

2. Connect as the `SH` user and execute the following statements to create the `TEMP_CHANNELS` table.

```
CREATE TABLE temp_channels
PARALLEL 5 AS
SELECT *
FROM sh.channels
WHERE channel_id IN ('2','3','4');
```

```
SQL> connect sh/sh
SQL> CREATE TABLE temp_channels
  2  PARALLEL 5 AS
  3  SELECT *
  4  FROM sh.channels
  5  WHERE channel_id IN ('2','3','4');
Table created.
```

3. Connect as SYSDBA and change the PARALLEL_MAX_SERVERS parameter to 5 and the PARALLEL_ADAPTIVE_MULTI_USER to FALSE. Verify that the change is in effect.

```
SQL> connect / as sysdba
SQL> alter system set parallel_max_servers = 5;

System altered.

SQL> alter system set parallel_adaptive_multi_user = false;

System altered.

SQL> COL name FORMAT A35
SQL> COL value FORMAT A15
SQL> SELECT name, value FROM v$parameter
  2  WHERE UPPER(name) IN ('LARGE_POOL_SIZE','PROCESSES','SESSIONS',
  3  'PARALLEL_EXECUTION_MESSAGE_SIZE','PARALLEL_MIN_SERVERS',
  4  'PARALLEL_MAX_SERVERS','PARALLEL_ADAPTIVE_MULTI_USER',
  5  'PARALLEL_MIN_PERCENT');

NAME                                VALUE
----------------------------------- ---------------
processes                           150
sessions                            248
large_pool_size                     0
parallel_min_percent                0
parallel_min_servers                0
parallel_max_servers                5
parallel_execution_message_size     16384
parallel_adaptive_multi_user        FALSE

8 rows selected.
```

4. In a single statement, create a TEMP_SALES table as the SH user with minimal logging and with a DOP set to 5. This table should be a subset of the SALES table; it should contain only rows with CHANNEL_ID values '2' or '3'. Execute the following SQL statements to create the table. Examine V$PQ_SESSTAT to determine the DOP for the operation.

```
CREATE TABLE temp_sales NOLOGGING PARALLEL 5 AS
   SELECT *
   FROM sh.sales
   WHERE channel_id IN ('2','3');
```

```
SQL> connect sh/sh

SQL> CREATE TABLE temp_sales NOLOGGING PARALLEL 5 AS
  2     SELECT *
  3     FROM sh.sales
  4     WHERE channel_id IN ('2','3');

Table created.

SQL> select * from v$pq_sesstat;
```

```
STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Queries Parallelized                    0             0
DML Parallelized                        0             0
DDL Parallelized                        1             1
DFO Trees                               1             1
Server Threads                          5             0
Allocation Height                       5             0
Allocation Width                        1             0
Local Msgs Sent                       151           151
Distr Msgs Sent                         0             0
Local Msgs Recv'd                     151           151
Distr Msgs Recv'd                       0             0
```

5. Execute the following SQL statements. This script joins the `TEMP_SALES` and `TEMP_CHANNELS` tables using a DOP of 5. Examine `V$PQ_SESSTAT` to determine the DOP for the operation.

```
SELECT  count(*)
FROM temp_sales s, temp_channels c
WHERE (s.channel_id) = (c.channel_id);
```

```
SQL> connect sh/sh
Connected.
SQL> SELECT  count(*)
  2  FROM temp_sales s, temp_channels c
  3  WHERE (s.channel_id) = (c.channel_id);

  COUNT(*)
----------
    837918

SQL> select * from v$pq_sesstat;

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Queries Parallelized                    1             1
DML Parallelized                        0             0
DDL Parallelized                        0             1
DFO Trees                               1             2
Server Threads                          4             0
Allocation Height                       2             0
Allocation Width                        1             0
Local Msgs Sent                        94           245
Distr Msgs Sent                         0             0
Local Msgs Recv'd                      94           245
Distr Msgs Recv'd                       0             0

11 rows selected.
```

**Answer:** You can see from the query of V$PQ_SESSTAT that the number of server threads is 4 and the allocation height is 2. This means that the statement executed with a DOP of 2 with 2 sets of slaves even though you specified a DOP of 5, and that the database should have been able to accommodate this number.

6. Connect as SYSDBA and restore the value of PARALLEL_MAX_SERVERS to its original value. Reconnect as the SH user, rerun the query in step 5, and inspect V$PQ_SESSTAT when the query has finished. Compare the results with those in step 5. What are your conclusions?

```
SQL> connect / as sysdba
Connected.
SQL> alter system set PARALLEL_MAX_SERVERS = 20;

System altered.

SQL> connect sh/sh
Connected.
SQL> SELECT  count(*)
  2  FROM temp_sales s, temp_channels c
  3  WHERE (s.channel_id) = (c.channel_id);

  COUNT(*)
----------
    837918

SQL> select * from v$pq_sesstat;

STATISTIC                      LAST_QUERY SESSION_TOTAL
------------------------------ ---------- -------------
Queries Parallelized                    1             1
DML Parallelized                        0             0
DDL Parallelized                        0             0
DFO Trees                               1             1
Server Threads                         10             0
Allocation Height                       5             0
Allocation Width                        1             0
Local Msgs Sent                       229           229
Distr Msgs Sent                         0             0
Local Msgs Recv'd                     229           229
Distr Msgs Recv'd                       0             0

11 rows selected.
SQL>
```

**Answer:** You can see from the query of V$PQ_SESSTAT that the number of server threads is 10 and the allocation height is 5. This means that the statement executed with a DOP of 5 with 2 sets of slaves. When the same statement is run in an environment where the value of PARALLEL_MAX_SERVERS is equal to that of the requested DOP, as was the case in step 5, the DOP is reduced by the adaptive multiuser algorithm to ensure there are enough parallel resources for other user requests.

7. After you have finished, drop the `TEMP_CHANNELS` and `TEMP_SALES` tables.

```
SQL> drop table temp_channels;
Table dropped.

SQL> drop table temp_sales;
Table dropped.
```

## Manual Solutions for Practice 3-9: Star Schema Tuning

In this practice, you optimize a query to use star transformation and access the benefits of using this optimizer technique.

1. From a terminal session, connected as the `oracle` user, execute the `setup_star_schema_lab.sh` script that is located in your `/home/oracle/labs` directory.

```
$ ./setup_star_schema_lab.sh
SQL*Plus: Release 11.2.0.3.0 Production on Thu Mar 7 19:33:15 2013

Copyright (c) 1982, 2011, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL>
Grant succeeded.

SQL> SQL>
User altered.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
```

2. From the same terminal window, start a SQL*Plus session connected as the `SH` user and do not disconnect from it until this practice finishes. Before executing the following SQL statement, ensure that you flush both the shared pool and the buffer cache to avoid caching issues as much as possible. After this, analyze the execution of the following query (you can use the `first_run.sql` script):

```
SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
                SUM(s.amount_sold) sales_amount
FROM sales s, times t, customers c, channels ch
WHERE s.time_id = t.time_id                    AND
      s.cust_id = c.cust_id                    AND
      s.channel_id = ch.channel_id             AND
      c.cust_state_province = 'CA'             AND
      ch.channel_desc in ('Internet','Catalog') AND
      t.calendar_quarter_desc IN ('1999-01','1999-02','2000-03','2000-04')
GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc;
```

What are your conclusions?

a) As you can see in the output of the execution plan, this query seems to use a large number of bytes to access the SALES table. Basically, the optimizer performs a full scan of this table. This might not be the best way to handle it.

```
$ sqlplus sh/sh
SQL> @first_run
SQL> connect / as sysdba
Connected.
SQL> grant dba to sh;

Grant succeeded.

SQL> connect sh/sh
Connected.
SQL> set echo on
SQL>
SQL> alter system flush shared_pool;

System altered.

SQL> alter system flush buffer_cache;

System altered.

SQL>
SQL> set pagesize 200
SQL> set linesize 250
SQL> set timing on
SQL> set autotrace on
SQL>
SQL> SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
  2     SUM(s.amount_sold) sales_amount
  3  FROM sales s, times t, customers c, channels ch
  4  WHERE s.time_id = t.time_id
  5  AND   s.cust_id = c.cust_id
  6  AND   s.channel_id = ch.channel_id
  7  AND   c.cust_state_province = 'CA'
  8  AND   ch.channel_desc in ('Internet','Catalog')
  9  AND   t.calendar_quarter_desc IN ('1999-01','1999-02','2000-03','2000-
04')
 10  GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc;

CHANNEL_CLASS        CUST_CITY                      CALENDA SALES_AMOUNT
-------------------- ------------------------------ ------- ------------
Indirect             Quartzhill                     1999-01        987.3
Indirect             Arbuckle                       1999-02        241.2
Indirect             Pala                           2000-03      4694.03
Indirect             San Francisco                  2000-04     13227.99
Indirect             Montara                        2000-04         1319
Indirect             Cloverdale                     2000-04         7.27
Indirect             Legrand                        1999-02        18.66
Indirect             San Francisco                  2000-03      4536.43
Indirect             Pala                           2000-04      1728.06
Indirect             Pescadero                      2000-04       278.93
Indirect             Los Angeles                    1999-01      1886.19
Indirect             Pescadero                      1999-02       298.44
Indirect             Cloverdale                     1999-02       266.28
Indirect             Montara                        2000-03       591.31
Indirect             Quartzhill                     2000-03       492.84
Indirect             Pescadero                      2000-03       659.33
Indirect             San Mateo                      2000-04     38794.78
Indirect             Legrand                        1999-01        26.32
Indirect             Pescadero                      1999-01        26.32
Indirect             El Sobrante                    2000-03      7160.35
Indirect             El Sobrante                    2000-04      7644.07
Indirect             Quartzhill                     2000-04       557.26
Indirect             San Mateo                      1999-01      8754.59
Indirect             Montara                        1999-01       289.07
Indirect             Arbuckle                       1999-01       270.08
Indirect             El Sobrante                    1999-02      3744.03
Indirect             Los Angeles                    1999-02      2128.59
Indirect             Pala                           1999-02       936.62
Indirect             San Mateo                      2000-03      17480.9
Indirect             Legrand                        2000-03         9.33
Indirect             Los Angeles                    2000-04      1984.65
Indirect             El Sobrante                    1999-01      5392.34
Indirect             Cloverdale                     1999-01        52.64
Indirect             San Francisco                  1999-02        11257
Indirect             Los Angeles                    2000-03      1660.31
Indirect             Arbuckle                       2000-03       565.49
Indirect             Pala                           1999-01      3263.93
Indirect             San Francisco                  1999-01      3058.27
Indirect             San Mateo                      1999-02     21399.42
Indirect             Montara                        1999-02      1618.01
Indirect             Quartzhill                     1999-02       412.83

41 rows selected.

Elapsed: 00:00:00.14

Execution Plan
----------------------------------------------------------
Plan hash value: 593420798

-----------------------------------------------------------------------------------------------
--------------
| Id  | Operation                       | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
Pstart| Pstop |
-----------------------------------------------------------------------------------------------
--------------
|   0 | SELECT STATEMENT                |           |  1144 | 96096 |   927   (3)| 00:00:12 |
|       |
|   1 |  HASH GROUP BY                  |           |  1144 | 96096 |   927   (3)| 00:00:12 |
|       |
|*  2 |   HASH JOIN                     |           |  6231 |   511K|   926   (3)| 00:00:12 |
|       |
|*  3 |    TABLE ACCESS FULL            | CHANNELS  |     2 |    42 |     3   (0)| 00:00:01 |
|       |
|*  4 |    HASH JOIN                    |           | 12462 |   766K|   923   (3)| 00:00:12 |
|       |
|   5 |     PART JOIN FILTER CREATE     | :BF0000   |   365 |  5840 |    18   (0)| 00:00:01 |
|       |
|*  6 |      TABLE ACCESS FULL          | TIMES     |   365 |  5840 |    18   (0)| 00:00:01 |
|       |
|*  7 |     HASH JOIN                   |           | 49822 |  2286K|   904   (3)| 00:00:11 |
|       |
|*  8 |      TABLE ACCESS FULL          | CUSTOMERS |   383 |  9958 |   406   (1)| 00:00:05 |
|       |
|   9 |      PARTITION RANGE JOIN-FILTER|           |   918K|    18M|   493   (3)| 00:00:06
|:BF0000|:BF0000|
|  10 |       TABLE ACCESS FULL         | SALES     |   918K|    18M|   493   (3)| 00:00:06
|:BF0000|:BF0000|
-----------------------------------------------------------------------------------------------
--------------

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

   2 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
   3 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
   4 - access("S"."TIME_ID"="T"."TIME_ID")
   6 - filter("T"."CALENDAR_QUARTER_DESC"='1999-01' OR "T"."CALENDAR_QUARTER_DESC"='1999-02' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-03' OR "T"."CALENDAR_QUARTER_DESC"='2000-04')
   7 - access("S"."CUST_ID"="C"."CUST_ID")
   8 - filter("C"."CUST_STATE_PROVINCE"='CA')


Statistics
----------------------------------------------------------
      11253  recursive calls
          0  db block gets
       4586  consistent gets
       2011  physical reads
        476  redo size
       2168  bytes sent via SQL*Net to client
        546  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
        158  sorts (memory)
          0  sorts (disk)
         41  rows processed

SQL>
SQL>
```

3. Without modifying the `SH` schema, how can you improve the execution plan for the query mentioned in step 2? Verify your solution and explain why it is probably a better solution. You can use `second_run.sql`.
   a) Enable star transformation in your session. In this step, you do not want to use a temporary table for the star transformation. Looking at the previous execution plan, the optimizer estimates the data that is to be manipulated in megabytes. Using the star transformation as follows, the estimation is now expressed in kilobytes. That is why this new execution plan is probably a much better alternative. However, note that this time the `CUSTOMERS` table is accessed using full scan twice. If the table is larger, the impact is significant.

```
SQL> @second_run
SQL> set echo on
SQL>
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.02
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.11
SQL>
SQL> set pagesize 200
SQL> set linesize 250
SQL> set timing on
SQL> set autotrace on
SQL>
SQL>
SQL> ALTER SESSION SET star_transformation_enabled=TEMP_DISABLE;

Session altered.
```

```
Elapsed: 00:00:00.00
SQL>
SQL>
SQL> SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
  2     SUM(s.amount_sold) sales_amount
  3  FROM sales s, times t, customers c, channels ch
  4  WHERE s.time_id = t.time_id
  5  AND   s.cust_id = c.cust_id
  6  AND   s.channel_id = ch.channel_id
  7  AND   c.cust_state_province = 'CA'
  8  AND   ch.channel_desc in ('Internet','Catalog')
  9  AND   t.calendar_quarter_desc IN ('1999-01','1999-02','2000-03','2000-
04')
 10  GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc;

CHANNEL_CLASS        CUST_CITY                      CALENDA SALES_AMOUNT
-------------------- ------------------------------ ------- ------------
Indirect             Quartzhill                     1999-01        987.3
Indirect             Arbuckle                       1999-02        241.2
Indirect             Pala                           2000-03      4694.03
Indirect             San Francisco                  2000-04     13227.99
Indirect             Montara                        2000-04         1319
Indirect             Cloverdale                     2000-04         7.27
Indirect             Legrand                        1999-02        18.66
Indirect             San Francisco                  2000-03      4536.43
Indirect             Pala                           2000-04      1728.06
Indirect             Pescadero                      2000-04       278.93
Indirect             Los Angeles                    1999-01      1886.19
Indirect             Pescadero                      1999-02       298.44
Indirect             Cloverdale                     1999-02       266.28
Indirect             Montara                        2000-03       591.31
Indirect             Quartzhill                     2000-03       492.84
Indirect             Pescadero                      2000-03       659.33
Indirect             San Mateo                      2000-04     38794.78
Indirect             Legrand                        1999-01        26.32
Indirect             Pescadero                      1999-01        26.32
Indirect             El Sobrante                    2000-03      7160.35
Indirect             El Sobrante                    2000-04      7644.07
Indirect             Quartzhill                     2000-04       557.26
Indirect             San Mateo                      1999-01      8754.59
Indirect             Montara                        1999-01       289.07
Indirect             Arbuckle                       1999-01       270.08
Indirect             El Sobrante                    1999-02      3744.03
Indirect             Los Angeles                    1999-02      2128.59
Indirect             Pala                           1999-02       936.62
Indirect             San Mateo                      2000-03      17480.9
Indirect             Legrand                        2000-03         9.33
Indirect             Los Angeles                    2000-04      1984.65
Indirect             El Sobrante                    1999-01      5392.34
Indirect             Cloverdale                     1999-01        52.64
Indirect             San Francisco                  1999-02        11257
Indirect             Los Angeles                    2000-03      1660.31
Indirect             Arbuckle                       2000-03       565.49
Indirect             Pala                           1999-01      3263.93
Indirect             San Francisco                  1999-01      3058.27
Indirect             San Mateo                      1999-02     21399.42
Indirect             Montara                        1999-02      1618.01
Indirect             Quartzhill                     1999-02       412.83

41 rows selected.

Elapsed: 00:00:00.17

Execution Plan
----------------------------------------------------------
Plan hash value: 1039573286
```

```
-----------------------------------------------------------------------------------------------
-----------------------------
| Id  | Operation                                | Name               | Rows  | Bytes | Cost
(%CPU)| Time       | Pstart| Pstop |
-----------------------------------------------------------------------------------------------
-----------------------------
|   0 | SELECT STATEMENT                         |                    |   506 | 42504 |  1408
(1)| 00:00:17 |         |       |
|   1 |  HASH GROUP BY                           |                    |   506 | 42504 |  1408
(1)| 00:00:17 |         |       |
|*  2 |   HASH JOIN                              |                    |   506 | 42504 |   980
(1)| 00:00:12 |         |       |
|*  3 |    TABLE ACCESS FULL                     | CUSTOMERS          |   383 |  9958 |   406
(1)| 00:00:05 |         |       |
|*  4 |    HASH JOIN                             |                    |   506 | 29348 |   574
(1)| 00:00:07 |         |       |
|*  5 |     TABLE ACCESS FULL                    | CHANNELS           |     2 |    42 |     3
(0)| 00:00:01 |         |       |
|*  6 |     HASH JOIN                            |                    |   506 | 18722 |   571
(1)| 00:00:07 |         |       |
|*  7 |      TABLE ACCESS FULL                   | TIMES              |   365 |  5840 |    18
(0)| 00:00:01 |         |       |
|   8 |      PARTITION RANGE SUBQUERY            |                    |   507 | 10647 |   552
(1)| 00:00:07 |KEY(SQ)|KEY(SQ)|
|   9 |       TABLE ACCESS BY LOCAL INDEX ROWID| SALES              |   507 | 10647 |   552
(1)| 00:00:07 |KEY(SQ)|KEY(SQ)|
|  10 |        BITMAP CONVERSION TO ROWIDS      |                    |       |       |      |
|       |       |
|  11 |         BITMAP AND                       |                    |       |       |      |
|       |       |
|  12 |          BITMAP MERGE                    |                    |       |       |      |
|       |       |
|  13 |           BITMAP KEY ITERATION           |                    |       |       |      |
|       |       |
|  14 |            BUFFER SORT                   |                    |       |       |      |
|       |       |
|* 15 |             TABLE ACCESS FULL            | CHANNELS           |     2 |    26 |     3
(0)| 00:00:01 |         |       |
|* 16 |            BITMAP INDEX RANGE SCAN       | SALES_CHANNEL_BIX |       |       |      |
|KEY(SQ)|KEY(SQ)|
|  17 |          BITMAP MERGE                    |                    |       |       |      |
|       |       |
|  18 |           BITMAP KEY ITERATION           |                    |       |       |      |
|       |       |
|  19 |            BUFFER SORT                   |                    |       |       |      |
|       |       |
|* 20 |             TABLE ACCESS FULL            | TIMES              |   365 |  5840 |    18
(0)| 00:00:01 |         |       |
|* 21 |            BITMAP INDEX RANGE SCAN       | SALES_TIME_BIX     |       |       |      |
|KEY(SQ)|KEY(SQ)|
|  22 |          BITMAP MERGE                    |                    |       |       |      |
|       |       |
|  23 |           BITMAP KEY ITERATION           |                    |       |       |      |
|       |       |
|  24 |            BUFFER SORT                   |                    |       |       |      |
|       |       |
|* 25 |             TABLE ACCESS FULL            | CUSTOMERS          |   383 |  6128 |   406
(1)| 00:00:05 |         |       |
|* 26 |            BITMAP INDEX RANGE SCAN       | SALES_CUST_BIX     |       |       |      |
|KEY(SQ)|KEY(SQ)|
-----------------------------------------------------------------------------------------------
-----------------------------

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

   2 - access("S"."CUST_ID"="C"."CUST_ID")
   3 - filter("C"."CUST_STATE_PROVINCE"='CA')
   4 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
   5 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
   6 - access("S"."TIME_ID"="T"."TIME_ID")
   7 - filter("T"."CALENDAR_QUARTER_DESC"='1999-01' OR "T"."CALENDAR_QUARTER_DESC"='1999-02' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-03' OR "T"."CALENDAR_QUARTER_DESC"='2000-04')
  15 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
  16 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
  20 - filter("T"."CALENDAR_QUARTER_DESC"='1999-01' OR "T"."CALENDAR_QUARTER_DESC"='1999-02' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-03' OR "T"."CALENDAR_QUARTER_DESC"='2000-04')
  21 - access("S"."TIME_ID"="T"."TIME_ID")
  25 - filter("C"."CUST_STATE_PROVINCE"='CA')
  26 - access("S"."CUST_ID"="C"."CUST_ID")

Note
-----
   - star transformation used for this statement


Statistics
----------------------------------------------------------
      11545  recursive calls
         20  db block gets
      21259  consistent gets
       2126  physical reads
        476  redo size
       2168  bytes sent via SQL*Net to client
        546  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
        167  sorts (memory)
          0  sorts (disk)
         41  rows processed

SQL>
```

4. How would you enhance the previous optimization without changing the SH schema? You can use third_run.sql.
   a) Let the optimizer decide if it is better to use a temporary table. You can try to set the STAR_TRANSFORMATION_ENABLED parameter to TRUE.

```
SQL> @third_run
SQL> @third_runwn command beginning "
SQL> set echo on
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.02
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.04
SQL>
SQL> set pagesize 200
SQL> set linesize 250
SQL> set timing on
SQL> set autotrace on
SQL>
SQL> ALTER SESSION SET star_transformation_enabled=TRUE;

Session altered.
```

```
Elapsed: 00:00:00.00
SQL>
SQL> SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
  2     SUM(s.amount_sold) sales_amount
  3  FROM sales s, times t, customers c, channels ch
  4  WHERE s.time_id = t.time_id
  5  AND   s.cust_id = c.cust_id
  6  AND   s.channel_id = ch.channel_id
  7  AND   c.cust_state_province = 'CA'
  8  AND   ch.channel_desc in ('Internet','Catalog')
  9  AND   t.calendar_quarter_desc IN ('1999-01','1999-02','2000-03','2000-
04')
 10  GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc;

CHANNEL_CLASS        CUST_CITY                      CALENDA SALES_AMOUNT
-------------------- ------------------------------ ------- ------------
Indirect             Quartzhill                     1999-01        987.3
Indirect             Arbuckle                       1999-02        241.2
Indirect             Pala                           2000-03      4694.03
Indirect             San Francisco                  2000-04     13227.99
Indirect             Montara                        2000-04         1319
Indirect             Cloverdale                     2000-04         7.27
Indirect             Legrand                        1999-02        18.66
Indirect             San Francisco                  2000-03      4536.43
Indirect             Pala                           2000-04      1728.06
Indirect             Pescadero                      2000-04       278.93
Indirect             Los Angeles                    1999-01      1886.19
Indirect             Pescadero                      1999-02       298.44
Indirect             Cloverdale                     1999-02       266.28
Indirect             Montara                        2000-03       591.31
Indirect             Quartzhill                     2000-03       492.84
Indirect             Pescadero                      2000-03       659.33
Indirect             San Mateo                      2000-04     38794.78
Indirect             Legrand                        1999-01        26.32
Indirect             Pescadero                      1999-01        26.32
Indirect             El Sobrante                    2000-03      7160.35
Indirect             El Sobrante                    2000-04      7644.07
Indirect             Quartzhill                     2000-04       557.26
Indirect             San Mateo                      1999-01      8754.59
Indirect             Montara                        1999-01       289.07
Indirect             Arbuckle                       1999-01       270.08
Indirect             El Sobrante                    1999-02      3744.03
Indirect             Los Angeles                    1999-02      2128.59
Indirect             Pala                           1999-02       936.62
Indirect             San Mateo                      2000-03      17480.9
Indirect             Legrand                        2000-03         9.33
Indirect             Los Angeles                    2000-04      1984.65
Indirect             El Sobrante                    1999-01      5392.34
Indirect             Cloverdale                     1999-01        52.64
Indirect             San Francisco                  1999-02        11257
Indirect             Los Angeles                    2000-03      1660.31
Indirect             Arbuckle                       2000-03       565.49
Indirect             Pala                           1999-01      3263.93
Indirect             San Francisco                  1999-01      3058.27
Indirect             San Mateo                      1999-02     21399.42
Indirect             Montara                        1999-02      1618.01
Indirect             Quartzhill                     1999-02       412.83

41 rows selected.

Elapsed: 00:00:00.20

Execution Plan
----------------------------------------------------------
Plan hash value: 3672241944
```

```
--------------------------------------------------------------------------------------------------
------------------------------------
| Id  | Operation                         | Name                      | Rows  | Bytes |
Cost (%CPU)| Time      | Pstart| Pstop |
--------------------------------------------------------------------------------------------------
------------------------------------
|   0 | SELECT STATEMENT                  |                           |   506 | 36938 |
603   (1)| 00:00:08 |       |       |
|   1 |  TEMP TABLE TRANSFORMATION        |                           |       |       |
|           |       |       |
|   2 |   LOAD AS SELECT                  | SYS_TEMP_0FD9D6620_4A69 |       |       |
|           |       |       |
|*  3 |    TABLE ACCESS FULL              | CUSTOMERS                 |   383 |  9958 |
406   (1)| 00:00:05 |       |       |
|   4 |   HASH GROUP BY                   |                           |   506 | 36938 |
197   (2)| 00:00:03 |       |       |
|*  5 |    HASH JOIN                      |                           |   506 | 36938 |
173   (2)| 00:00:03 |       |       |
|   6 |     TABLE ACCESS FULL             | SYS_TEMP_0FD9D6620_4A69 |   383 |  5745 |
2   (0)| 00:00:01 |       |       |
|*  7 |     HASH JOIN                     |                           |   506 | 29348 |
171   (2)| 00:00:03 |       |       |
|*  8 |      TABLE ACCESS FULL            | CHANNELS                  |     2 |    42 |
3   (0)| 00:00:01 |       |       |
|*  9 |      HASH JOIN                    |                           |   506 | 18722 |
167   (1)| 00:00:03 |       |       |
|* 10 |       TABLE ACCESS FULL           | TIMES                     |   365 |  5840 |
18   (0)| 00:00:01 |       |       |
|  11 |       PARTITION RANGE SUBQUERY    |                           |   507 | 10647 |
149   (1)| 00:00:02 |KEY(SQ)|KEY(SQ)|
|  12 |        TABLE ACCESS BY LOCAL INDEX ROWID| SALES               |   507 | 10647 |
149   (1)| 00:00:02 |KEY(SQ)|KEY(SQ)|
|  13 |         BITMAP CONVERSION TO ROWIDS |                         |       |       |
|           |       |       |
|  14 |          BITMAP AND               |                           |       |       |
|           |       |       |
|  15 |           BITMAP MERGE            |                           |       |       |
|           |       |       |
|  16 |            BITMAP KEY ITERATION   |                           |       |       |
|           |       |       |
|  17 |             BUFFER SORT           |                           |       |       |
|           |       |       |
|* 18 |              TABLE ACCESS FULL    | CHANNELS                  |     2 |    26 |
3   (0)| 00:00:01 |       |       |
|* 19 |             BITMAP INDEX RANGE SCAN | SALES_CHANNEL_BIX       |       |       |
|           |KEY(SQ)|KEY(SQ)|
|  20 |           BITMAP MERGE            |                           |       |       |
|           |       |       |
|  21 |            BITMAP KEY ITERATION   |                           |       |       |
|           |       |       |
|  22 |             BUFFER SORT           |                           |       |       |
|           |       |       |
|* 23 |              TABLE ACCESS FULL    | TIMES                     |   365 |  5840 |
18   (0)| 00:00:01 |       |       |
|* 24 |             BITMAP INDEX RANGE SCAN | SALES_TIME_BIX          |       |       |
|           |KEY(SQ)|KEY(SQ)|
|  25 |           BITMAP MERGE            |                           |       |       |
|           |       |       |
|  26 |            BITMAP KEY ITERATION   |                           |       |       |
|           |       |       |
|  27 |             BUFFER SORT           |                           |       |       |
|           |       |       |
|  28 |              TABLE ACCESS FULL    | SYS_TEMP_0FD9D6620_4A69 |   383 |  1915 |
2   (0)| 00:00:01 |       |       |
|* 29 |             BITMAP INDEX RANGE SCAN | SALES_CUST_BIX          |       |       |
|           |KEY(SQ)|KEY(SQ)|
--------------------------------------------------------------------------------------------------
------------------------------------

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

```
   3 - filter("C"."CUST_STATE_PROVINCE"='CA')
   5 - access("S"."CUST_ID"="C0")
   7 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
   8 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
   9 - access("S"."TIME_ID"="T"."TIME_ID")
  10 - filter("T"."CALENDAR_QUARTER_DESC"='1999-01' OR "T"."CALENDAR_QUARTER_DESC"='1999-02' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-03' OR "T"."CALENDAR_QUARTER_DESC"='2000-04')
  18 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
  19 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
  23 - filter("T"."CALENDAR_QUARTER_DESC"='1999-01' OR "T"."CALENDAR_QUARTER_DESC"='1999-02' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-03' OR "T"."CALENDAR_QUARTER_DESC"='2000-04')
  24 - access("S"."TIME_ID"="T"."TIME_ID")
  29 - access("S"."CUST_ID"="C0")

Note
-----
   - star transformation used for this statement

Statistics
----------------------------------------------------------
      12105  recursive calls
         38  db block gets
      19880  consistent gets
       2158  physical reads
        672  redo size
       2168  bytes sent via SQL*Net to client
        546  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
        168  sorts (memory)
          0  sorts (disk)
         41  rows processed

SQL>
```

5. How do you eliminate one access on the CUSTOMERS table from the previous execution plan for the same SELECT statement seen in step 3?
   a) Create a bitmap join index between the SALES and CUSTOMERS tables.

6. Try to apply your finding. What happens and why?
   a) Because the CUSTOMERS_PK primary key constraint is not enforced, it is not possible to create a bitmap join index between the SALES and CUSTOMERS tables.

```
SQL> CREATE BITMAP INDEX sales_c_state_bjix ON
  2  sales(customers.cust_state_province)
  3  FROM sales, customers
  4  WHERE sales.cust_id=customers.cust_id
  5  LOCAL NOLOGGING COMPUTE STATISTICS;
FROM sales, customers
            *
ERROR at line 3:
ORA-25954: missing primary key or unique constraint on dimension
SQL>
```

7. Fix the issue you found and apply your solution from step 5 again.

```
SQL>  alter table customers enable constraint customers_pk;

Table altered.

SQL> CREATE BITMAP INDEX sales_c_state_bjix ON
```

```
  2  sales(customers.cust_state_province)
  3  FROM sales, customers
  4  WHERE sales.cust_id=customers.cust_id
  5  LOCAL NOLOGGING COMPUTE STATISTICS;

Index created.
SQL>
```

8. Verify that you solved the problem from step 5. You can use `fourth_run.sql`.

```
SQL> @fourth_run
SQL> set echo on
SQL>
SQL> ALTER SESSION SET star_transformation_enabled=TRUE;

Session altered.

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL> alter system flush buffer_cache;

System altered.

SQL>
SQL> set pagesize 200
SQL> set linesize 250
SQL> set timing on
SQL> set autotrace on
SQL>
SQL> SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
  2     SUM(s.amount_sold) sales_amount
  3  FROM sales s, times t, customers c, channels ch
  4  WHERE s.time_id = t.time_id
  5  AND   s.cust_id = c.cust_id
  6  AND   s.channel_id = ch.channel_id
  7  AND   c.cust_state_province = 'CA'
  8  AND   ch.channel_desc in ('Internet','Catalog')
  9  AND   t.calendar_quarter_desc IN ('1999-01','1999-02','2000-03','2000-
04')
 10  GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc;

CHANNEL_CLASS        CUST_CITY                      CALENDA SALES_AMOUNT
-------------------- ------------------------------ ------- ------------
Indirect             Quartzhill                     1999-01        987.3
Indirect             Arbuckle                       1999-02        241.2
Indirect             Pala                           2000-03      4694.03
Indirect             San Francisco                  2000-04     13227.99
Indirect             Montara                        2000-04         1319
Indirect             Cloverdale                     2000-04         7.27
Indirect             Legrand                        1999-02        18.66
Indirect             San Francisco                  2000-03      4536.43
Indirect             Pala                           2000-04      1728.06
Indirect             Pescadero                      2000-04       278.93
Indirect             Los Angeles                    1999-01      1886.19
Indirect             Pescadero                      1999-02       298.44
Indirect             Cloverdale                     1999-02       266.28
Indirect             Montara                        2000-03       591.31
Indirect             Quartzhill                     2000-03       492.84
Indirect             Pescadero                      2000-03       659.33
Indirect             San Mateo                      2000-04     38794.78
Indirect             Legrand                        1999-01        26.32
Indirect             Pescadero                      1999-01        26.32
Indirect             El Sobrante                    2000-03      7160.35
Indirect             El Sobrante                    2000-04      7644.07
Indirect             Quartzhill                     2000-04       557.26
Indirect             San Mateo                      1999-01      8754.59
Indirect             Montara                        1999-01       289.07
Indirect             Arbuckle                       1999-01       270.08
Indirect             El Sobrante                    1999-02      3744.03
Indirect             Los Angeles                    1999-02      2128.59
Indirect             Pala                           1999-02       936.62
Indirect             San Mateo                      2000-03      17480.9
Indirect             Legrand                        2000-03         9.33
Indirect             Los Angeles                    2000-04      1984.65
Indirect             El Sobrante                    1999-01      5392.34
Indirect             Cloverdale                     1999-01        52.64
Indirect             San Francisco                  1999-02        11257
Indirect             Los Angeles                    2000-03      1660.31
Indirect             Arbuckle                       2000-03       565.49
Indirect             Pala                           1999-01      3263.93
Indirect             San Francisco                  1999-01      3058.27
Indirect             San Mateo                      1999-02     21399.42
Indirect             Montara                        1999-02      1618.01
Indirect             Quartzhill                     1999-02       412.83

41 rows selected.

Elapsed: 00:00:00.15

Execution Plan
----------------------------------------------------------
Plan hash value: 632695221

-----------------------------------------------------------------------------------------------
------------------------------
| Id  | Operation                             | Name                | Rows  | Bytes | Cost
(%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------
------------------------------
|   0 | SELECT STATEMENT                      |                     |  1144 | 96096 |   577
(2)| 00:00:07 |       |       |
|   1 |  HASH GROUP BY                        |                     |  1144 | 96096 |   577
(2)| 00:00:07 |       |       |
|*  2 |   HASH JOIN                           |                     |  3975 |   326K|   555
(1)| 00:00:07 |       |       |
|*  3 |    TABLE ACCESS FULL                  | CHANNELS            |     2 |    42 |     3
(0)| 00:00:01 |       |       |
|*  4 |    HASH JOIN                          |                     |  3975 |   244K|   551
(1)| 00:00:07 |       |       |
|*  5 |     TABLE ACCESS FULL                 | TIMES               |   365 |  5840 |    18
(0)| 00:00:01 |       |       |
|*  6 |     HASH JOIN                         |                     |  3984 |   182K|   532
(1)| 00:00:07 |       |       |
|*  7 |      TABLE ACCESS FULL                | CUSTOMERS           |   383 |  9958 |   406
(1)| 00:00:05 |       |       |
|   8 |      PARTITION RANGE SUBQUERY         |                     | 73467 |  1506K|   126
(1)| 00:00:02 |KEY(SQ)|KEY(SQ)|
|   9 |       TABLE ACCESS BY LOCAL INDEX ROWID| SALES               | 73467 |  1506K|   126
(1)| 00:00:02 |KEY(SQ)|KEY(SQ)|
|  10 |        BITMAP CONVERSION TO ROWIDS     |                     |       |       |      |
|       |       |
|  11 |         BITMAP AND                    |                     |       |       |      |
|       |       |
|  12 |          BITMAP MERGE                 |                     |       |       |      |
|       |       |
|  13 |           BITMAP KEY ITERATION        |                     |       |       |      |
|       |       |
|  14 |           BUFFER SORT                 |                    |       |       |       |
|       |       |
|* 15 |            TABLE ACCESS FULL          | CHANNELS           |     2 |    26 |     3
(0)| 00:00:01 |         |       |
|* 16 |           BITMAP INDEX RANGE SCAN     | SALES_CHANNEL_BIX  |       |       |       |
|KEY(SQ)|KEY(SQ)|
|  17 |         BITMAP MERGE                  |                    |       |       |       |
|       |       |
|  18 |          BITMAP KEY ITERATION         |                    |       |       |       |
|       |       |
|  19 |           BUFFER SORT                 |                    |       |       |       |
|       |       |
|* 20 |            TABLE ACCESS FULL          | TIMES              |   365 |  5840 |    18
(0)| 00:00:01 |         |       |
|* 21 |           BITMAP INDEX RANGE SCAN     | SALES_TIME_BIX     |       |       |       |
|KEY(SQ)|KEY(SQ)|
|* 22 |         BITMAP INDEX SINGLE VALUE     | SALES_C_STATE_BJIX |       |       |       |
|KEY(SQ)|KEY(SQ)|
-----------------------------------------------------------------------------------------------
------------------------------

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

   2 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
   3 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
   4 - access("S"."TIME_ID"="T"."TIME_ID")
   5 - filter("T"."CALENDAR_QUARTER_DESC"='1999-01' OR "T"."CALENDAR_QUARTER_DESC"='1999-02' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-03' OR "T"."CALENDAR_QUARTER_DESC"='2000-04')
   6 - access("S"."CUST_ID"="C"."CUST_ID")
   7 - filter("C"."CUST_STATE_PROVINCE"='CA')
  15 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
  16 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
  20 - filter("T"."CALENDAR_QUARTER_DESC"='1999-01' OR "T"."CALENDAR_QUARTER_DESC"='1999-02' OR
              "T"."CALENDAR_QUARTER_DESC"='2000-03' OR "T"."CALENDAR_QUARTER_DESC"='2000-04')
  21 - access("S"."TIME_ID"="T"."TIME_ID")
  22 - access("S"."SYS_NC00008$"='CA')

Note
-----
   - star transformation used for this statement


Statistics
----------------------------------------------------------
      12651  recursive calls
          0  db block gets
       6398  consistent gets
       2024  physical reads
        332  redo size
       2168  bytes sent via SQL*Net to client
        546  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
        176  sorts (memory)
          0  sorts (disk)
         41  rows processed
```

9. Clean up your environment by removing the index that you created and by returning the constraint to its original state by using the `cleanup_star_schema_lab.sql` script.

```
SQL> @cleanup_star_schema_lab
```

## Manual Solutions for Practice 4-1: Implementing Fine-Grained Auditing

In the `orcl` database, there is a business requirement that a record must be logged whenever employee salary information is accessed. `INSERT`, `UPDATE`, and `DELETE` are recorded in a journal table by using triggers. Create a proof of concept solution for `SELECT` accesses. Create a user `PFAY` and prove that `SELECT` accesses will be recorded.

1. Create a security officer account that has privileges to create user accounts, grant privileges, and administer fine-grained auditing and fine-grained access control. This account is named `SEC` with the password `sec`. In this and subsequent practices, security functionality is implemented in a single user. Create this user, giving it the following properties:

```
Name is SEC
Password is sec
Can create a session and grant the privilege to other users to
create a session
Can select from any table in the database
Can create or drop any context in the database
Can create, alter, and drop users
Can create roles and can alter and drop any roles
Can create tables, procedures, and triggers (including the
ADMINISTER DATABASE TRIGGER privilege, which allows the user to
create database triggers)
Can execute DBMS_SESSION. This privilege is granted from the SYS
user to PUBLIC by default.
```

**Answer:**

Connect as `sysdba` and create the `SEC` user with proper privileges.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

SQL> DROP USER sec CASCADE;
DROP USER sec
          *
ERROR at line 1:
ORA-01918: user 'SEC' does not exist

SQL> CREATE USER sec IDENTIFIED BY sec;

User created.

SQL> GRANT create session
  2    TO sec
  3    WITH ADMIN OPTION;

Grant succeeded.

SQL> GRANT select any dictionary, select any table,
  2      create any context, drop any context,
  3      create user, alter user, drop user,
  4      create role, alter any role, drop any role,
  5      create table, create procedure,
  6      create any trigger, administer database trigger
  7    TO sec;

Grant succeeded.

SQL> GRANT execute on DBMS_SESSION to sec;

Grant succeeded.
```

2. As the `SEC` user, create the `PFAY` user and grant `SELECT` access to the `HR.EMPLOYEES` table to `PFAY`.
   Create the `PFAY` user with the password `e!car0` (*e*-exclamation point-*c*-*a*-*r*-zero).
   Grant `PFAY` the required access.

   **Answer:**

```
SQL> CONNECT sec/sec
Connected.
SQL> DROP USER pfay;
DROP USER pfay
          *
ERROR at line 1:
ORA-01918: user 'PFAY' does not exist

SQL> GRANT create session TO pfay IDENTIFIED BY "e!car0";

Grant succeeded.

SQL> CONNECT hr/hr
Connected.
SQL> GRANT select ON hr.employees TO pfay;

Grant succeeded.
```

3. Make sure that `EXAMPLE` is the default tablespace for the `SEC` user and that `SEC` has the ability to create objects in the `EXAMPLE` tablespace.

   **Answer:**

```
SQL> CONNECT system/oracle
Connected.
SQL> ALTER USER sec
  2   DEFAULT TABLESPACE example
  3   QUOTA UNLIMITED ON example;

User altered.
```

4. Enable the `SEC` user to execute the `DBMS_FGA` package.

**Answer:**

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> GRANT execute ON dbms_fga TO sec;

Grant succeeded.
```

5. Create a Fine-Grained Auditing (FGA) policy with the following properties:
   Object: `HR.EMPLOYEES`
   Name: `AUDIT_EMPS_SALARY`
   Audits: Any access to the `SALARY` column
   Policy: Disabled

**Answer:**

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> BEGIN
  2  dbms_fga.add_policy (
  3    object_schema         => 'hr',
  4    object_name           => 'employees',
  5    policy_name           => 'audit_emps_salary',
  6    audit_condition       => NULL,
  7    audit_column          => 'salary',
  8    enable                => TRUE );
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

6. As the `PFAY` user, select `SALARY` from the `HR.EMPLOYEES` table.

**Answer:**

```
SQL> CONNECT pfay/e!car0
Connected.
SQL>
SQL> SELECT salary FROM hr.employees;

    SALARY
----------
     24000
     17000
     17000
      9000
      6000
      4800
      4800
      4200
     12000
      9000
      8200

    SALARY
----------
      7700
      7800
      6900
     11000
      3100
      2900
      2800
      2600
      2500
      8000
      8200
…   Rows deleted  …
      6500
     10000
     12000
      8300

107 rows selected.

SQL>
```

7. As the SYSTEM user, display the audit record from the previous SELECT statement.

   **Note:** The time stamp that is shown is the time when step 6 was executed.

   **Answer:**

```
SQL> CONNECT system/oracle
Connected.
SQL>
SQL> SET pagesize 1000
SQL>
SQL> SELECT  to_char(timestamp, 'YYMMDDHH24MI') AS timestamp,
  2          db_user,
  3          object_schema,
  4          object_name,
  5          policy_name,
  6          sql_bind,
  7          sql_text
  8   FROM   dba_fga_audit_trail;

TIMESTAMP  DB_USER OBJECT_SCHEMA   OBJECT_NAME
---------- ------- --------------- ------------
POLICY_NAME          SQL_BIND
-------------------- ----------
SQL_TEXT
-----------------------------------------------------
0609070732 PFAY    HR              EMPLOYEES
```

```
AUDIT_EMPS_SALARY
SELECT salary FROM hr.employees

SQL>
```

## Manual Solutions for Practice 4-2: Using an Encrypted Tablespace

1. Create an Oracle encryption wallet.

   As the database software owner, create a directory for the wallet.

   **/u01/app/oracle/admin/orcl/wallet**

```
$ mkdir -p /u01/app/oracle/admin/orcl/wallet
```

2. Modify the `sqlnet.ora` file (located at `/u01/app/oracle/product/11.2.0/dbhome_1/network/admin`) to use the encryption wallet. Using your preferred editor, add the following lines:

```
ENCRYPTION_WALLET_LOCATION=
  (SOURCE=
     (METHOD = FILE)
     (METHOD_DATA =
        (DIRECTORY = /u01/app/oracle/admin/orcl/wallet)))
```

3. In the `orcl` database, create a database master encryption key with the password `welcome1`.

```
$ sqlplus / as sysdba

SQL> alter system set encryption key identified by "welcome1";

System altered.

SQL>
```

4. Create an encrypted tablespace named `ENCTS` that uses the default encryption algorithm with a size of 10 MB. Use the name of data file `$ORACLE_HOME/dbs/encts.dat`.

```
SQL> CREATE TABLESPACE encts
  2  DATAFILE '$ORACLE_HOME/dbs/encts.dat' SIZE 10M
  3  ENCRYPTION
  4  DEFAULT STORAGE (ENCRYPT);

Tablespace created.

SQL> exit
```

5. View the data dictionary.

```
SQL> select TABLESPACE_NAME,ENCRYPTED from dba_tablespaces
  2  where tablespace_name='ENCTS';

TABLESPACE_NAME                ENC
------------------------------ ---
ENCTS                          YES
```

6. Drop the `ENCTS` tablespace.

```
SQL> drop tablespace encts including contents and datafiles;

Tablespace dropped.

SQL>
```

## Manual Solutions for Practice 4-3: Using Flashback Versions Query

If there is no HR schema on the orcl database, run the HR.sh script that is located in/home/oracle/labs. The script will create the HR schema.

```
$ cd /home/oracle/labs
$ ./HR.sh
```

1. Execute the following SQL to query the HR.LOCATIONS table for location ID 1400.

```
SELECT *
FROM hr.locations
WHERE location_id = 1400;
```

**Answer:**

a) In SQL*Plus, execute the preceding SQL.

```
SQL> connect hr/hr
SQL> SELECT *
  2  FROM hr.locations
  3  WHERE location_id = 1400;

LOCATION_ID STREET_ADDRESS                           POSTAL_CODE
----------- ---------------------------------------- ------------
CITY                           STATE_PROVINCE            CO
------------------------------ ------------------------- --
       1400 2014 Jabberwocky Rd                      26192
Southlake                      Texas                     US
```

2. Execute the following SQL to update the POSTAL_CODE column in the HR.LOCATIONS table, simulating user error.

```
UPDATE hr.locations
SET postal_code = postal_code + 100
WHERE location_id = 1400;
commit;
```

**Answer:**

a) Invoke SQL*Plus and execute the preceding SQL.

```
SQL> UPDATE hr.locations
  2  SET postal_code = postal_code + 100
  3  WHERE location_id = 1400;

1 row updated.

SQL> commit;

Commit complete.
```

3. Execute the following SQL to query the `POSTAL_CODE` column in `HR.LOCATIONS` and view the change.

```
SELECT *
FROM hr.locations
WHERE location_id = 1400;
```

**Answer:**

a) Execute the preceding SQL.

```
SQL> SELECT *
  2  FROM hr.locations
  3  WHERE location_id = 1400;

LOCATION_ID STREET_ADDRESS                           POSTAL_CODE
----------- ---------------------------------------- ------------
CITY                           STATE_PROVINCE            CO
------------------------------ ------------------------- --
       1400 2014 Jabberwocky Rd                      26292
Southlake                      Texas                     US
```

4. Execute the following SQL to update the `POSTAL_CODE` column in the `HR.LOCATIONS` table, simulating user error.

```
UPDATE hr.locations
SET postal_code = postal_code + 100
WHERE location_id = 1400;
commit;
```

**Answer:**

a) Execute the following SQL:

```
SQL> UPDATE hr.locations
  2  SET postal_code = postal_code + 100
  3  WHERE location_id = 1400;

1 row updated.

SQL> commit;

Commit complete.
```

5. Perform Flashback Versions Query to correct user errors.

**Answer:**

a) Use SQL*Plus to perform Flashback Versions Query.

```
SQL> col VERSIONS_STARTTIME for a20
SQL> col VERSIONS_ENDTIME for a20
SQL> SELECT versions_startscn, versions_starttime,
  2         versions_endscn, versions_endtime,
  3        versions_xid, versions_operation,
  4        LOCATION_ID, POSTAL_CODE
  5    FROM hr.locations
  6    VERSIONS BETWEEN TIMESTAMP
  7       TO_TIMESTAMP('2010-11-26 22:10:00', 'YYYY-MM-DD HH24:MI:SS')
  8    AND TO_TIMESTAMP('2010-11-26 22:15:00', 'YYYY-MM-DD HH24:MI:SS')
  9    WHERE location_id = 1400;

VERSIONS_STARTSCN VERSIONS_STARTTIME   VERSIONS_ENDSCN VERSIONS_ENDTIME
----------------- -------------------- --------------- -----------------
---
VERSIONS_XID     V LOCATION_ID POSTAL_CODE
---------------- - ----------- ------------
           444828 26-NOV-10 10.12.05 P
                  M
03001E0080010000 U        1400 26392

           444817 26-NOV-10 10.11.38 P          444828 26-NOV-10
10.12.05 P
                  M                                    M
0800100085010000 U        1400 26292

                                                444817 26-NOV-10
10.11.38 P

VERSIONS_STARTSCN VERSIONS_STARTTIME   VERSIONS_ENDSCN VERSIONS_ENDTIME
----------------- -------------------- --------------- -----------------
---
VERSIONS_XID     V LOCATION_ID POSTAL_CODE
---------------- - ----------- ------------
                                                       M
                          1400 26192
```

b) Flashback the locations table to before the user error by the Flashback Table.

```
SQL> alter table hr.locations enable row movement;

Table altered.

SQL> FLASHBACK TABLE hr.locations TO SCN 444816;

Flashback complete.
```

6. Return to your SQL*Plus session. Query the HR.LOCATIONS table to confirm the Flashback operation.

```
SELECT *
FROM hr.locations
WHERE location_id = 1400;
```

**Answer:**

a) Execute the following SQL to query the HR.LOCATIONS table.

```
SQL> SELECT *
  2  FROM hr.locations
  3  WHERE location_id = 1400;

LOCATION_ID STREET_ADDRESS                           POSTAL_CODE
----------- ---------------------------------------- ------------
CITY                           STATE_PROVINCE            CO
------------------------------ ------------------------- --
       1400 2014 Jabberwocky Rd                      26192
Southlake                      Texas                     US
```

## Manual Solutions for Practice 4-4: Creating an Application Context

In this practice, you create an application context, set the context using a secure package, and test the context.

1. Connect as `SYSTEM` using `oracle` as the password and `orcl` as the connect string. Using the `SYS_CONTEXT` procedure, display the following session-related attributes:

   ```
   CURRENT_USER
   SESSION_USER
   PROXY_USER
   IP_ADDRESS
   NETWORK_PROTOCOL
   AUTHENTICATION_TYPE
   AUTHENTICATION_DATA
   CLIENT_IDENTIFIER
   ```

   You can use one of the following techniques to call `SYS_CONTEXT`:

   ```
   SELECT sys_context('userenv','…')FROM dual;

   EXEC dbms_output.put_line(syscontext('userenv','…'));
   ```

   (Make sure to issue `SET SERVEROUTPUT ON` before executing `DBMS_OUTPUT`.)

   **Answer:**

   Both ways of viewing the `userenv` context attributes are shown in the following solution:

```
$ sqlplus /nolog

SQL*Plus: Release 11.2.0.3.0 Production on Mon Mar 11 15:15:35 2013

Copyright (c) 1982, 2011, Oracle.  All rights reserved.
SQL> SET ECHO ON
SQL> CONNECT system/oracle@orcl
Connected.

SQL> SET SERVEROUTPUT ON

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'CURRENT_USER'));
SYSTEM

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'SESSION_USER'));
SYSTEM
```

```
PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('USERENV', 'IP_ADDRESS'));
10.190.112.211

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'NETWORK_PROTOCOL'));
tcp

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('USERENV',
'AUTHENTICATION_TYPE'));
DATABASE

PL/SQL procedure successfully completed.

SQL> SELECT sys_context('USERENV', 'CURRENT_USER') FROM DUAL;

SYS_CONTEXT('USERENV','CURRENT_USER')
-----------------------------------------------------------
SYSTEM

SQL> SELECT sys_context('USERENV', 'SESSION_USER') FROM DUAL;

SYS_CONTEXT('USERENV','SESSION_USER')
-----------------------------------------------------------
SYSTEM

SQL> SELECT sys_context('USERENV', 'IP_ADDRESS') FROM DUAL;

SYS_CONTEXT('USERENV','IP_ADDRESS')
-----------------------------------------------------------
10.190.112.211

SQL> SELECT sys_context('USERENV', 'NETWORK_PROTOCOL') FROM DUAL;

SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
--------------------------------------------------------
tcp

SQL> SELECT sys_context('USERENV', 'AUTHENTICATION_TYPE') FROM DUAL;

SYS_CONTEXT('USERENV','AUTHENTICATION_TYPE')
-----------------------------------------------------------
DATABASE

SQL>
```

2. Implement a local application context with the following properties:

   Name: `EMP_USER`
   Package Name: `current_emp`
   Owned by: `SEC`

   It contains the following attributes, which are listed with the column from the `HR.EMPLOYEES` table that is used to obtain the attribute value:

| Attribute | Column from `HR.EMPLOYEES` |
|---|---|
| `ID` | `EMPLOYEE_ID` |
| `NAME` | `FIRST_NAME \|\| ' ' \|\| LAST_NAME` |
| `EMAIL` | `EMAIL` |

   **Answer:**

```
SQL> CONNECT sec/sec
Connected.
SQL> CREATE CONTEXT emp_user USING current_emp;
Context created.
```

3. The row in the `EMPLOYEES` table that is used to populate the attributes is selected by comparing the `EMAIL` column to the `SESSION_USER` attribute from `SYS_CONTEXT`.

   The procedure that sets the application context has the following properties:

   Owned by: `SEC` user
   Part of: `CURRENT_EMP` package
   Name: `SET_EMP_INFO`
   It is called from a logon trigger named `EMP_LOGON` that is also owned by `SEC`. This trigger applies to all users.

   Execute `$HOME/labs/lab_04_04_03_a.sql` to create the package and package body.
   Execute `$HOME/labs/lab_04_04_03_b.sql` to create the logon trigger.

   **Answer:**

```
SQL> @lab_04_04_03_a.sql
SQL> CREATE OR REPLACE PACKAGE current_emp IS
  2    PROCEDURE set_emp_info;
  3  END;
  4  /

Package created.

SQL> CREATE OR REPLACE PACKAGE BODY current_emp IS
  2    PROCEDURE set_emp_info
  3    IS
  4      v_employee_id       hr.employees.employee_id%TYPE;
  5      v_first_name        hr.employees.first_name%TYPE;
  6      v_last_name         hr.employees.last_name%TYPE;
  7    BEGIN
  8      SELECT employee_id,
  9            first_name,
 10            last_name
 11        INTO v_employee_id,
 12             v_first_name,
 13             v_last_name
 14        FROM hr.employees
 15        WHERE email = SYS_CONTEXT('USERENV', 'SESSION_USER');
 16      DBMS_SESSION.SET_CONTEXT('emp_user', 'id',
 17        v_employee_id);
 18      DBMS_SESSION.SET_CONTEXT('emp_user', 'name',
 19        v_first_name || ' ' || v_last_name);
 20      DBMS_SESSION.SET_CONTEXT('emp_user', 'email',
 21        SYS_CONTEXT('USERENV', 'SESSION_USER'));
 22    EXCEPTION
 23      WHEN no_data_found THEN NULL;
 24    END;
 25  END;
 26  /

Package body created.

SQL> @lab_04_04_03_b.sql
SQL> CREATE or REPLACE TRIGGER emp_logon
  2  AFTER LOGON ON DATABASE
  3  BEGIN
  4    current_emp.set_emp_info;
  5  END;
  6  /

Trigger created.
```

4. Test the context that you created by performing the following steps:
   a) Create a user named `SKING` with the `CREATE SESSION` privilege.
   b) Log in as `SKING`.
   c) Use `SYS_CONTEXT` to verify that the `EMP_USER` context attributes are set. If you use `DBMS_OUTPUT`, remember to issue the `SET SERVEROUTPUT ON` command.

   **Answer:**

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> GRANT create session TO sking IDENTIFIED BY sking;

Grant succeeded.

SQL>
SQL> CONNECT sking/sking
Connected.
SQL> SET SERVEROUTPUT ON
SQL>
SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'id'));
100

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'name'));
Steven King

PL/SQL procedure successfully completed.

SQL> EXEC dbms_output.put_line(sys_context('emp_user', 'email'));
SKING

PL/SQL procedure successfully completed.
```

5. Review `lab_04_04_05.sql`, connect as `SKING`, and execute this script, which lists all the application context attributes that are set in the current session. Because Label Security is installed, the `LBAC$LABELS LBAC$LASTSEQ` attributes are part of the context but not populated because Label Security is not yet configured.

   **Answer:**

```
SQL> connect sking/sking
Connected.
SQL> @lab_04_04_05.sql
SQL> SET ECHO OFF
SQL> SET SERVEROUTPUT ON
SQL>
SQL> DECLARE
  2    list dbms_session.AppCtxTabTyp;
  3    cnt number;
  4  BEGIN
  5    dbms_session.list_context (list, cnt);
  6    IF cnt = 0
  7      THEN dbms_output.put_line('No contexts active.');
  8      ELSE
  9        FOR i IN 1..cnt LOOP
 10          dbms_output.put_line(list(i).namespace
 11            ||' ' || list(i).attribute
 12            || ' = ' || list(i).value);
 13        END LOOP;
 14    END IF;
 15  END;
 16  /
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ =

PL/SQL procedure successfully completed.
```

6. Log in as `SEC` and select information about the application context that you created from the data dictionary.

**Answer:**

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> SELECT *
  2    FROM dba_context
  3    WHERE namespace = 'EMP_USER';

NAMESPACE                      SCHEMA
------------------------------ ---------------------------
PACKAGE                        TYPE
------------------------------ ----------------------
EMP_USER                       SEC
CURRENT_EMP                    ACCESSED LOCALLY
```

7. What happens when you call `DBMS_SESSION.SET_CONTEXT` to set an attribute in the `EMP_USER` context? Assume that `SKING` wants to change the context setting.

   **Answer:**

   Because the application context is set with a package, `SKING` does not have sufficient privileges to execute the `DBMS_SESSION.SET_CONTEXT` procedure.

```
SQL> connect sking/sking
Connected.
SQL> @lab_04_04_05.sql
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ =

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_SESSION.SET_CONTEXT('emp_user', 'id', 1);
BEGIN DBMS_SESSION.SET_CONTEXT('emp_user', 'id', 1); END;

*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 82
ORA-06512: at line 1

SQL> @lab_04_04_05.sql
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100
LBAC$LABELS LBAC$LASTSEQ =

PL/SQL procedure successfully completed.
```

## Manual Solutions for Practice 4-5: Implementing a Fine-Grained Access Control Policy

In this practice, you create, enable, and test a Fine-Grained Access Control (FGAC) policy.

1. How does FGAC determine which rows belong in the Virtual Private Database (VPD) for the current user?

   **Answer:**

   FGAC adds a predicate (condition) to the `WHERE` clause on a `SELECT` or DML statement with an `AND` operator.

2. How does FGAC know which tables are defined in the VPD?

   **Answer:**

   You include a table name or view name when the FGAC policy is created.

3. The `SEC` user also needs the privilege to create policies. As `SYSTEM`, grant `SEC` the ability to execute the package that creates policies.

   **Answer:**

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba
SQL> GRANT execute ON dbms_rls TO sec;

Grant succeeded
```

4. What privilege exempts the user from access policies? Why does the `SEC` user need this privilege? Grant it to `SEC`.

   **Answer:**

   The `EXEMPT ACCESS POLICY` privilege is very powerful. Statements that are issued by a user with this privilege do not have any FGAC policies applied.

```
SQL> CONNECT system/oracle
Connected.
SQL>
SQL> GRANT exempt access policy TO sec;

Grant succeeded
```

5. The `lab_04_05_05.sql` script creates the package that is used by the security policy to return a predicate.

   a) Review and execute the script.

   **Answer:**

```
SQL> @lab_04_05_05.sql
SQL> set echo off
SQL> CONNECT sec/sec
Connected.
```

```
SQL>
SQL> CREATE OR REPLACE PACKAGE hr_policy_pkg IS
  2    FUNCTION limit_emp_emp (
  3      object_schema       IN      VARCHAR2,
  4      object_name         VARCHAR2 )
  5      RETURN VARCHAR2;
  6  END;
  7  /

Package created.

SQL>
SQL> CREATE OR REPLACE PACKAGE BODY hr_policy_pkg IS
  2    FUNCTION limit_emp_emp (
  3      object_schema       IN      VARCHAR2,
  4      object_name         VARCHAR2 )
  5      RETURN VARCHAR2
  6    IS
  7      v_emp_id NUMBER;
  8    BEGIN
  9      RETURN 'employee_id = SYS_CONTEXT(''emp_user'', ''id'')';
 10    END;
 11  END;
 12  /

Package body created.
```

b) What predicate does the policy use to limit the rows returned from the `EMPLOYEE` table?

**Answer:**

`employee_id = SYS_CONTEXT('emp_user', 'id')`

c) How does this predicate limit the rows?

**Answer:**

The user making the query must have an `EMAIL_ID` that matches the database username, and the `emp_user` attribute in `SYS_CONTEXT` is set equal to the `employee_id` of the user (see Practice 4-4, step 4). The predicate allows the user to access only the record describing the user.

6. Test the policy function.

**Answer:**

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> SELECT hr_policy_pkg.limit_emp_emp('a', 'b') FROM DUAL;

HR_POLICY_PKG.LIMIT_EMP_EMP('A','B')
----------------------------------------------------------
employee_id = SYS_CONTEXT('emp_user', 'id')
```

7. Implement a policy with the following characteristics:

   The policy limits the rows that are selected from the `HR.EMPLOYEES` table.

   The policy is named `HR_EMP_POL`.

   The function that is used to return a predicate is `SEC.HR_POLICY_PKG.LIMIT_EMP_EMP`.

   **Answer:**

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> EXEC dbms_rls.drop_policy('HR', 'EMPLOYEES', 'HR_EMP_POL');
BEGIN dbms_rls.drop_policy('HR', 'EMPLOYEES', 'HR_EMP_POL'); END;

*
ERROR at line 1:
ORA-28102: policy does not exist
ORA-06512: at "SYS.DBMS_RLS", line 59
ORA-06512: at line 1

SQL> EXEC dbms_rls.add_policy ('HR', 'EMPLOYEES', 'HR_EMP_POL', -
>   'SEC', 'HR_POLICY_PKG.LIMIT_EMP_EMP', 'SELECT')

PL/SQL procedure successfully completed.
```

8. Set up the `SKING` user so that he can access the `HR.EMPLOYEES` table.

   **Answer:**

```
SQL> connect hr/hr
Connected.
SQL> GRANT select ON employees TO sking;

Grant succeeded.

SQL>
```

9. As `SKING`, execute the `lab_04_04_05.sql` script, which displays the current context attributes.

   **Answer:**

```
SQL> connect sking/sking
Connected.
SQL> @lab_04_04_05
EMP_USER NAME = Steven King
EMP_USER EMAIL = SKING
EMP_USER ID = 100

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

10. Which rows are returned when SKING queries the HR.EMPLOYEE table without a WHERE clause? Try it.

    **Answer:**

```
SQL> select employee_id, first_name, last_name, email
  2  from HR.EMPLOYEES;

EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- -------------------------
EMAIL
-------------------------
        100 Steven               King
SKING
```

11. As the SEC user, delete the FGAC policy just created.

    **Answer:**

```
SQL> CONNECT sec/sec
Connected.
SQL>
SQL> EXEC dbms_rls.drop_policy('HR', 'EMPLOYEES', 'HR_EMP_POL');

PL/SQL procedure successfully completed.
```

## Manual Solutions for Practice 4-6: Using Flashback Data Archive

In this practice, you use Flashback Data Archive.

1. Ensure that you are pointing to the `orcl` database. Using SQL*Plus, connect to the database as the `SYS` user and execute the `lab_04_06_01.sql` script from the `/home/oracle/labs` directory. The script creates a small `FLA_TBS1` tablespace, creates the `ARCHIVE_ADMIN` user with the `ARCHIVE_ADMIN` password, and unlocks the `HR` user with the `hr` password. The password is case-sensitive by default.

```
$ sqlplus / as sysdba

SQL> @lab_04_06_01
Connected.
SQL> set serveroutput on
SQL> -- set verify on
SQL> set term on
SQL> set lines 200
SQL> set pages 44
SQL> set pause on
SQL>
SQL> /*== Create a tablespace for your flashback data archive ==*/
SQL> DROP TABLESPACE fla_tbs1 INCLUDING CONTENTS
  2  /
DROP TABLESPACE fla_tbs1 INCLUDING CONTENTS
*
ERROR at line 1:
ORA-00959: tablespace 'FLA_TBS1' does not exist


SQL> CREATE SMALLFILE TABLESPACE fla_tbs1
  2  DATAFILE '/u01/app/oracle/oradata/orcl/fla_tbs01.dbf'
  3  SIZE 10M REUSE AUTOEXTEND ON NEXT 640K MAXSIZE 32767M
  4  NOLOGGING EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO
  5  /

Tablespace created.

SQL> /*== Set up the HR database account for this lesson ==*/
SQL> /*==  Note: The HR user has the UNLIMITED TABLESPACE system
privilege. ==*/
SQL>
SQL> ALTER USER hr IDENTIFIED BY "hr" ACCOUNT UNLOCK
  2  /

User altered.

SQL> /*== Create an ARCHIVE_ADMIN user like the HR user ==*/
SQL> /*==   with FLA_TBS1 default tablespace ==*/
SQL>
SQL> CREATE USER ARCHIVE_ADMIN PROFILE DEFAULT IDENTIFIED BY
ARCHIVE_ADMIN
  2  DEFAULT TABLESPACE FLA_TBS1 TEMPORARY TABLESPACE TEMP
  3  ACCOUNT UNLOCK
  4  /

User created.

SQL> pause Press [Enter] to continue...
Press [Enter] to continue...

SQL> GRANT ALTER SESSION TO ARCHIVE_ADMIN;

Grant succeeded.

SQL> GRANT CREATE DATABASE LINK TO ARCHIVE_ADMIN;

Grant succeeded.

SQL> GRANT CREATE SEQUENCE TO ARCHIVE_ADMIN;

Grant succeeded.

SQL> GRANT CREATE SESSION TO ARCHIVE_ADMIN;

Grant succeeded.

SQL> GRANT CREATE SYNONYM TO ARCHIVE_ADMIN;

Grant succeeded.

SQL> GRANT CREATE VIEW TO ARCHIVE_ADMIN;

Grant succeeded.

SQL> GRANT UNLIMITED TABLESPACE TO ARCHIVE_ADMIN;

Grant succeeded.

SQL> GRANT EXECUTE ON SYS.DBMS_STATS TO ARCHIVE_ADMIN;

Grant succeeded.

SQL> GRANT CONNECT, RESOURCE TO ARCHIVE_ADMIN;

Grant succeeded.

SQL>
SQL> set pause off
SQL> set feedback on
```

2. Give the ARCHIVE_ADMIN user administrative privileges for creating, maintaining, and dropping flashback data archives.

```
SQL> GRANT FLASHBACK ARCHIVE ADMINISTER TO archive_admin;

Grant succeeded.
```

3. In SQL*Plus, connect as the ARCHIVE_ADMIN user with the ARCHIVE_ADMIN password.

```
SQL> connect archive_admin/ARCHIVE_ADMIN
Connected.
```

4. Create a Flashback Data Archive named fla1 by using fla_tbs1 tablespace and set retention to 1 year.

```
SQL> CREATE FLASHBACK ARCHIVE fla1
  2  TABLESPACE fla_tbs1
  3  QUOTA 10M
  4  RETENTION 1 YEAR
  5 /

Flashback archive created.
```

5. Give the privilege to use the fla1 archive to the HR user.

```
SQL> GRANT FLASHBACK ARCHIVE on FLA1 to HR;

Grant succeeded.
```

6. Now, switch to the role of a flashback archive user. Connect as the HR user with the hr password. Enable this flashback archive for the EMPLOYEES table.

```
SQL> conn hr/hr
Connected.
SQL> ALTER TABLE hr.employees FLASHBACK ARCHIVE fla1;

Table altered.
```

7. To view and increase the salary of Mr. Fox three times by 1000, execute the lab_04_06_07.sql script. This produces activity in the Flashback Data Archive.

```
SQL> @lab_04_06_07
SQL> REM "********************************************* "
SQL> REM "For demo purposes ONLY: Flashback Data Archive "
SQL>
SQL> connect hr/hr
Connected.
SQL>
SQL> set echo on
SQL> set serveroutput on
SQL> -- set verify on
SQL> set term on
SQL> set lines 200
SQL> set pages 44
SQL> set pause on pause "Press [Enter] to continue..."
SQL>
SQL> /*== Query the current salary for Mr. Fox ==*/
SQL>
SQL> SELECT employee_id, last_name, salary
  2  FROM   hr.employees
  3  WHERE  last_name = 'Fox'
  4  /
Press [Enter] to continue...

EMPLOYEE_ID LAST_NAME                     SALARY
----------- ------------------------- ----------
        170 Fox                             9600

1 row selected.

SQL>
SQL> pause Press [Enter] to continue...
Press [Enter] to continue...

SQL>
SQL> /*== Increase the  salary three times by 1000 ==*/
SQL>
SQL> UPDATE hr.employees
  2  SET salary = salary + 1000
  3  WHERE last_name = 'Fox'
  4  /

1 row updated.

SQL> COMMIT
  2  /

Commit complete.

SQL> UPDATE hr.employees
  2  SET salary = salary + 1000
  3  WHERE last_name = 'Fox'
  4  /

1 row updated.

SQL> COMMIT
  2  /

Commit complete.

SQL> UPDATE hr.employees
  2  SET    salary = salary + 1000
  3  WHERE  last_name = 'Fox'
  4  /

1 row updated.

SQL> COMMIT
```

```
  2  /

Commit complete.

SQL> pause Press [Enter] to continue...
Press [Enter] to continue...

SQL>
SQL> /*== Query the up-to-date value for Mr. Fox ==*/
SQL>
SQL> SELECT employee_id, last_name, salary
  2  FROM   hr.employees
  3  WHERE  last_name = 'Fox'
  4  /
Press [Enter] to continue...

EMPLOYEE_ID LAST_NAME                     SALARY
----------- ------------------------- ----------
        170 Fox                            12600

1 row selected.

SQL>
SQL> pause  Press [Enter] to continue...
Press [Enter] to continue...

SQL>
SQL> set pause off
SQL> set feedback on
SQL>
```

8. Query the internal name of the archive table.

```
SQL> SELECT * FROM USER_FLASHBACK_ARCHIVE_TABLES;

TABLE_NAME                     OWNER_NAME
------------------------------ ------------------------------
FLASHBACK_ARCHIVE_NAME
----------------------------------------------------------------------
----------------------------------------------------------------------
-------------------------------------------------------------
ARCHIVE_TABLE_NAME                                     STATUS
----------------------------------------------------- --------
EMPLOYEES                      HR
FLA1
SYS_FBA_HIST_17351                                     ENABLED


1 row selected.
```

9. As the HR user, choose a time after the creation of the Flashback Data Archive and before you executed the erroneous DML. To view Mr. Fox's employee record as of that time,

execute the following query ( replace `'10' MINUTES` with your chosen historic date, format examples:`'50' SECOND`,`'10' DAY`,`'5' MONTH`):

**Note:** You receive an ORA-1466 error, if you specify a time before the Flashback Data Archive was started. Reduce the time to a smaller interval and try again. If you still see the salary of 12600, increase your time interval.

```
SELECT employee_id, last_name, salary
FROM hr.employees AS OF TIMESTAMP
(SYSTIMESTAMP - INTERVAL '10' MINUTE)
WHERE last_name = 'Fox';
```

```
SQL> connect hr/hr
Connected.
SQL> SELECT employee_id, last_name, salary
  2  FROM employees  AS OF TIMESTAMP
  3  (SYSTIMESTAMP - INTERVAL '10' MINUTE)
  4  WHERE last_name = 'Fox';

EMPLOYEE_ID LAST_NAME                     SALARY
----------- ------------------------- ----------
        170 Fox                             9600

1 row selected.
```

10. As the `HR` user, you realize that the recent updates were mistakes. Revert to the original values for your chosen historic date (for example, 10 minutes ago).

```
UPDATE hr.employees
SET salary = (SELECT salary FROM hr.employees
AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '10' MINUTE)
WHERE last_name = 'Fox')
WHERE last_name = 'Fox';
```

```
SQL> UPDATE hr.employees
  2  SET salary = (SELECT salary FROM hr.employees
  3  AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '10' MINUTE)
  4  WHERE last_name = 'Fox')
  5  WHERE last_name = 'Fox';

1 row updated.

SQL> SELECT employee_id, last_name, salary
  2  FROM employees
  3  WHERE last_name = 'Fox';

EMPLOYEE_ID LAST_NAME                     SALARY
----------- ------------------------- ----------
        170 Fox                             9600

1 row selected.
```

11. As the `ARCHIVE_ADMIN` user, drop the `fla1` Flashback Data Archive.
    **Note:** Dropping a Flashback Data Archive includes dropping the internal tamper-proofed history table. You cannot drop this table directly due to auditing and security requirements. Dropping a Flashback Data Archive does not drop the tablespaces in which they are stored, because the tablespaces might contain other data.

```
SQL> connect ARCHIVE_ADMIN/ARCHIVE_ADMIN
Connected.
SQL> DROP FLASHBACK ARCHIVE fla1;

Flashback archive dropped.
```

12. Connected as the `SYS` user, clean up your environment by deleting the `fla_tbs1` tablespace and the `ARCHIVE_ADMIN` user.

```
SQL> connect / as sysdba
Connected.
SQL> DROP TABLESPACE fla_tbs1 INCLUDING CONTENTS AND DATAFILES;

Tablespace dropped.

SQL> DROP USER archive_admin CASCADE;

User dropped.
```

## Manual Solutions for Practice 4-7: Using SecureFiles

1. In the `orcl` database, create a tablespace, `secf_tbs`, by using the following SQL statements. Enable `sh` to be able to use the `secf_tbs` tablespace.

```
create tablespace secf_tbs datafile
'/u01/app/oracle/oradata/orcl/secf_tbs.dbf' size 10M;
```

```
$ sqlplus / as sysdba

SQL> create tablespace secf_tbs datafile
'/u01/app/oracle/oradata/orcl/secf_tbs.dbf' size 10M;

Tablespace created.
```

2. Connect to the `orcl` database as the `sh` user, create the `T1` table in the `secf_tbs` tablespace with the following columns:

- EMPNO NUMBER
- C1 CLOB

Enable compression (High) and deduplication for the `C1` column.

```
SQL> connect sh/sh
Connected.

SQL> create table t1(empno number,c1 clob) LOB(c1)
  2  store as securefile
  3  (tablespace secf_tbs compress high deduplicate);
Table created.

SQL> col table_name for a10
SQL> col column_name for a10
SQL> col tablespace_name for a10
SQL> select table_name,column_name,tablespace_name,
  2  compression,deduplication
  3  from user_lobs;

TABLE_NAME COLUMN_NAM TABLESPACE COMPRESSION
---------- ---------- ---------- ------------------
DEDUPLICATION
---------------------------------------------
T1         C1         SECF_TBS   HIGH
LOB
```

3. Clean up your environment by dropping table t1 and deleting the secf_tbs tablespace.

```
SQL> connect sh/sh
Connected.
SQL> drop table t1;
```

```
Table dropped.

SQL> connect / as sysdba
Connected.
SQL> drop tablespace secf_tbs including contents and datafiles;

Tablespace dropped.

SQL> exit
```

## Manual Solutions for Practice 4-8: Using Reference Partitioning

Reference partitioning enables tables with a parent/child relationship to be logically equipartitioned by inheriting the partition key from the parent table without duplicating the key columns. In this practice, you will create a range-partitioned table called `ORDERS`, then create a table called `ORDER_ITEMS`, which is reference-partitioned on a foreign key contained in the `ORDERS` table.

If there is no SH schema on the `orcl` database, run the `SH.sh` script that is located in `/home/oracle/labs`. The script will create the `SH` schema.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. Start a SQL*Plus session connected to the `orcl` database as the `SYS` user. From your SQL*Plus session, execute the `lab_04_08_01.sql` script.

```
$ sqlplus / as sysdba

SQL> @lab_04_08_01
SQL> set echo on
SQL> alter user sh identified by sh account unlock;

User altered.

SQL> alter user sh quota unlimited on users;

User altered.
SQL>
```

2. Still in your SQL*Plus session, connect as the `SH` user. Execute the `lab_04_08_02.sql` script to create the range-partitioned `ORDERS` table.

```
SQL> connect sh/sh
SQL> @lab_04_08_02
SQL> REM
SQL> REM REF Partitioning
SQL> REM 1.Introduce the concept of REF Partitioning
SQL> REM 2.Create a simple partitioned table (e.g. Orders,
partitioned by order_date, PK order_id)
SQL>
SQL> set echo on
SQL>
SQL> set pagesize 2000
SQL> set long 10000
SQL> set linesize 200
SQL> set feedback on
SQL> column partition_name format a25
SQL> column high_value format a85
SQL>
SQL> -- cleanup
SQL>
SQL> drop table order_items
  2  /
drop table order_items
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> drop table orders
  2  /
drop table orders
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> -- create a range-partitioned table orders
SQL>
SQL> create table orders
  2  ( order_id number(12) not null
  3  , order_date date not null
  4  , order_mode varchar2(8)
  5  , order_status varchar2(1)
  6  )
  7  partition by range (order_date)
  8  ( partition p_before_jan_2006 values less than (to_date('01-JAN-
2006','dd-MON-yyyy'))
  9  , partition p_2006_jan values less than (to_date('01-FEB-
2006','dd-MON-yyyy'))
 10  , partition p_2006_feb values less than (to_date('01-MAR-
2006','dd-MON-yyyy'))
 11  , partition p_2006_mar values less than (to_date('01-APR-
2006','dd-MON-yyyy'))
 12  , partition p_2006_apr values less than (to_date('01-MAY-
2006','dd-MON-yyyy'))
 13  , partition p_2006_may values less than (to_date('01-JUN-
2006','dd-MON-yyyy'))
 14  , partition p_2006_jun values less than (to_date('01-JUL-
2006','dd-MON-yyyy'))
 15  , partition p_2006_jul values less than (to_date('01-AUG-
2006','dd-MON-yyyy'))
 16  , partition p_2006_aug values less than (to_date('01-SEP-
2006','dd-MON-yyyy'))
 17  , partition p_2006_sep values less than (to_date('01-OCT-
2006','dd-MON-yyyy'))
 18  , partition p_2006_oct values less than (to_date('01-NOV-
2006','dd-MON-yyyy'))
```

```
 19  , partition p_2006_nov values less than (to_date('01-DEC-
2006','dd-MON-yyyy'))
 20  , partition p_2006_dec values less than (to_date('01-JAN-
2007','dd-MON-yyyy'))
 21  ) parallel
 22  /

Table created.

SQL>
SQL> alter table orders add constraint orders_pk
  2  primary key (order_id)
  3  /

Table altered.
```

3. Create a reference-partitioned `ORDER_ITEMS` table as follows:

```
Table Name:ORDER_ITEMS
Columns:order_id number(12) not null
  , product_id number not null
  , quantity number not null
  , sales_amount number not null
References:order_id references order_id of ORDERS
```

```
SQL> create table order_items
  2 ( order_id number(12) not null
  3  , product_id number not null
  4  , quantity number not null
  5  , sales_amount number not null
  6  , constraint order_items_orders_fk foreign key (order_id)
  7  references orders(order_id))
  8  partition by reference (order_items_orders_fk)
  9  /

Table created.
```

4. View information about the tables.

```
SQL> select table_name, partitioning_type, ref_ptn_constraint_name
  2  from user_part_tables
  3  where table_name in ('ORDERS','ORDER_ITEMS');

TABLE_NAME                     PARTITION REF_PTN_CONSTRAINT_NAME
------------------------------ --------- ----------------------------
--
ORDERS                         RANGE
ORDER_ITEMS                    REFERENCE ORDER_ITEMS_ORDERS_FK

2 rows selected.
```

```
SQL> select table_name, partition_name, high_value
  2  from user_tab_partitions
  3  where table_name in ('ORDERS','ORDER_ITEMS')
  4  order by partition_position, table_name;

TABLE_NAME                      PARTITION_NAME            HIGH_VALUE
------------------------------- ------------------------- ------------
----------------------------------------------------------------------
----
ORDERS                          P_BEFORE_JAN_2006         TO_DATE('
2006-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_BEFORE_JAN_2006
ORDERS                          P_2006_JAN                TO_DATE('
2006-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_JAN
ORDERS                          P_2006_FEB                TO_DATE('
2006-03-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_FEB
ORDERS                          P_2006_MAR                TO_DATE('
2006-04-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_MAR
ORDERS                          P_2006_APR                TO_DATE('
2006-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_APR
ORDERS                          P_2006_MAY                TO_DATE('
2006-06-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_MAY
ORDERS                          P_2006_JUN                TO_DATE('
2006-07-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_JUN
ORDERS                          P_2006_JUL                TO_DATE('
2006-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_JUL
ORDERS                          P_2006_AUG                TO_DATE('
2006-09-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_AUG
ORDERS                          P_2006_SEP                TO_DATE('
2006-10-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_SEP
```

```
ORDERS                          P_2006_OCT                 TO_DATE('
2006-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_OCT
ORDERS                          P_2006_NOV                 TO_DATE('
2006-12-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_NOV
ORDERS                          P_2006_DEC                 TO_DATE('
2007-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS',
'NLS_CALENDAR=GREGORIAN')
ORDER_ITEMS                     P_2006_DEC

26 rows selected.

SQL>
```

## Manual Solutions for Practice 4-9: Using Interval Partitioning

Interval partitioning fully automates the creation of range partitions. Managing the creation of new partitions can be a cumbersome and highly repetitive task. This is especially true for predictable additions of partitions covering small ranges, such as adding new daily partitions. Interval partitioning automates this operation by creating partitions on demand.

1. Connect as the `sh` user and create an interval-partitioned `NEWSALES` table using the following SQL statement:

```
Table Name:NEWSALES
Columns:prod_id number(6) not null
  , cust_id number not null
  , time_id date not null
  , channel_id char(1) not null
  , promo_id number(6) not null
  , quantity_sold number(3) not null
  , amount_sold number(10,2) not null
Partition Method: range,interval
Partition Column: time_id
Interval :1'DAY'
Initial partition:
p_before_1_jan_2005 values less than (to_date('01-01-
2005','dd-mm-yyyy'))
```

```
SQL> conn sh/sh
SQL> create table newsales
  2 ( prod_id number(6) not null
  3 , cust_id number not null
  4 , time_id date not null
  5 , channel_id char(1) not null
  6 , promo_id number(6) not null
  7 , quantity_sold number(3) not null
  8 , amount_sold number(10,2) not null
  9 )
 10 partition by range (time_id)
 11 interval (numtodsinterval(1,'DAY'))
 12 ( partition p_before_1_jan_2005 values
 13 less than (to_date('01-01-2005','dd-mm-yyyy')))
 14 /

Table created.
SQL>
```

2. Find information about the `NEWSALES` table in the dictionary view.

```
SQL> select partition_name, high_value
  2  from user_tab_partitions
  3  where table_name = 'NEWSALES'
  4  order by partition_position
  5  /

PARTITION_NAME            HIGH_VALUE
------------------------- -------------------------------------------
----------------------------------------
P_BEFORE_1_JAN_2005       TO_DATE(' 2005-01-01 00:00:00', 'SYYYY-MM-
DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')

1 row selected.

SQL>
```

3. Execute the `lab_04_09_03.sql` script to insert new data into the `NEWSALES` table that forces the creation of a new partition (segment).

```
SQL> @lab_04_09_03
SQL> set echo on
SQL> set pagesize 2000
SQL> set long 10000
SQL> set linesize 200
SQL> set feedback on
SQL> column partition_name format a25
SQL> column high_value format a85
SQL> REM
SQL> REM Interval Partitioning
SQL> REM 4.Insert new data that forces the creation of a new
partition (segment)
SQL> set pagesize 2000
SQL> set long 10000
SQL> set linesize 200
SQL> set feedback on
SQL> column partition_name format a25
SQL> column high_value format a85
SQL> -- insert some data after January 1 2005
SQL>
SQL> insert into newsales values (11160,17450,to_date('01-jan-
2005','dd-mon-yyyy'),'I',9999,19,798) ;

1 row created.

SQL> insert into newsales values (1340,33710,to_date('02-jan-
2005','dd-mon-yyyy'),'S',9999,16,1264) ;

1 row created.

SQL> insert into newsales values (25270,65880,to_date('05-jan-
2005','dd-mon-yyyy'),'I',9999,5,210) ;

1 row created.
```

```
SQL> insert into newsales values (1615,73480,to_date('05-jan-
2005','dd-mon-yyyy'),'I',9999,8,96) ;

1 row created.

SQL> insert into newsales values (1900,84910,to_date('06-jan-
2005','dd-mon-yyyy'),'I',9999,42,378) ;

1 row created.

SQL> insert into newsales values (8085,37900,to_date('09-jan-
2005','dd-mon-yyyy'),'S',9999,1,68) ;

1 row created.

SQL> insert into newsales values (755,26590,to_date('09-jan-
2005','dd-mon-yyyy'),'I',9999,11,132) ;

1 row created.

SQL> insert into newsales values (10,68060,to_date('09-jan-2005','dd-
mon-yyyy'),'P',9999,28,4900) ;

1 row created.

SQL> insert into newsales values (13425,109310,to_date('10-jan-
2005','dd-mon-yyyy'),'I',9999,1,68) ;

1 row created.

SQL> insert into newsales values (1955,65190,to_date('10-jan-
2005','dd-mon-yyyy'),'S',9999,28,1512) ;

1 row created.

SQL>
SQL> set feedback on
SQL>
SQL> commit
  2  /

Commit complete.

SQL>
```

4. Check the information about the new partition.

```
SQL> select partition_name, high_value
  2  from user_tab_partitions
  3  where table_name = 'NEWSALES'
  4  order by partition_position
  5  /
```

```
PARTITION_NAME            HIGH_VALUE
------------------------- ------------------------------------------
------------------------------------------
P_BEFORE_1_JAN_2005       TO_DATE(' 2005-01-01 00:00:00', 'SYYYY-MM-
DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
SYS_P21                   TO_DATE(' 2005-01-02 00:00:00', 'SYYYY-MM-
DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
SYS_P22                   TO_DATE(' 2005-01-03 00:00:00', 'SYYYY-MM-
DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
SYS_P23                   TO_DATE(' 2005-01-06 00:00:00', 'SYYYY-MM-
DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
SYS_P24                   TO_DATE(' 2005-01-07 00:00:00', 'SYYYY-MM-
DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
SYS_P25                   TO_DATE(' 2005-01-10 00:00:00', 'SYYYY-MM-
DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
SYS_P26                   TO_DATE(' 2005-01-11 00:00:00', 'SYYYY-MM-
DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')

7 rows selected.
```

## Manual Solutions for Practice 5-1: **EXPLAIN PLAN** and SQL*Plus **AUTOTRACE**

If there is no SH schema on the orcl database, run the SH.sh script that is located at /home/oracle/labs. The script will create the SH schema. You can ignore the errors on the Data Pump Import utility.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. Open a terminal window. Set the current directory to **labs**. Start SQL*Plus. Log in to the SH schema as SH with the password SH. Then check the existence and column structure of the CUSTOMERS table. View the structure of the PLAN_TABLE table.

   Also list the existing indexes on the CUSTOMERS table by using the lab_05_01_01.sql script.

```
$ cd /home/oracle/labs
$ sqlplus sh/sh
SQL> DESCRIBE customers
Name                               Null?    Type
 --------------------------------- -------- ----
 CUST_ID                           NOT NULL NUMbER
 CUST_FIRST_NAME                     NOT NULL VARCHAR2()
 CUST_LAST_NAME                      NOT NULL VARCHAR2()
 CUST_GENDER                        NOT NULL CHAR()
 CUST_YEAR_OF_BIRTH                   NOT NULL NUMBER
 CUST_MARITAL_STATUS                          VARCHAR2()
 CUST_STREET_ADDRESS                  NOT NULL VARCHAR2()
 CUST_POSTAL_CODE                    NOT NULL VARCHAR2()
 CUST_CITY                          NOT NULL VARCHAR2()
…
SQL> DESCRIBE plan_table
Name                                   Null?    Type
 ------------------------------------- -------- ----
 STATEMENT_ID                                   VAR)
 PLAN_ID                                       NUMR
 TIMESTAMP                                      DATE
 REMARKS                                       VAR)
 OPERATION                                      VAR)
 OPTIONS                                       VAR)
 OBJECT_NODE                                    VAR)
 OBJECT_OWNER                                   VAR)
…
SQL> @lab_05_01_01

List indexes on table : customers

TABLE_NAME                INDEX_TYPE           INDEX_NAME
------------------------- -------------------- ------------------------------
CUSTOMERS                   UNIQUE              CUSTOMERS_PK
                          BITMAP             CUSTOMERS_GENDER_BIX
                                            CUSTOMERS_YOB_BIX
                                            CUSTOMERS_MARITAL_BIX
SQL>
```

2. Explain the following SQL statement by using `EXPLAIN PLAN` and SQL*Plus `AUTOTRACE`.

```
SELECT cust_first_name,cust_last_name
FROM   customers
WHERE  cust_id =100
```

a) Enter the SQL statement by using `EXPLAIN PLAN`.

```
SQL> explain plan for
  2  SELECT cust_first_name,cust_last_name
  3  FROM   customers
  4  WHERE  cust_id =100;

Explained.
```

b) Query the `PLAN_TABLE` table:

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 4238351645

--------------------------------------------------------------------------------
------------

| Id  | Operation                   | Name         | Rows  | Bytes | Cost (%CPU)
| Time     |

--------------------------------------------------------------------------------
------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |              |     1 |    20 |     2   (0)
| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS    |     1 |    20 |     2   (0)
| 00:00:01 |

|*  2 |   INDEX UNIQUE SCAN         | CUSTOMERS_PK |     1 |       |     1   (0)
| 00:00:01 |

--------------------------------------------------------------------------------
------------

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

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

   2 - access("CUST_ID"=100)

14 rows selected.
```

c) Enable SQL*Plus `AUTOTRACE` to display execution plans and run the query again without the `EXPLAIN PLAN`.

```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> set linesize 100
SQL> SELECT cust_first_name, cust_last_name
  2  FROM   customers
  3  WHERE  cust_id = 100;

Execution Plan
----------------------------------------------------------
Plan hash value: 4238351645

--------------------------------------------------------------------------------------
-------
| Id  | Operation                   | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT            |              |     1 |    20 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS    |     1 |    20 |     2   (0)| 00:00:01
|
|*  2 |   INDEX UNIQUE SCAN         | CUSTOMERS_PK |     1 |       |     1   (0)| 00:00:01 |
--------------------------------------------------------------------------------------
-------

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

   2 - access("CUST_ID"=100)
```

d) If you have some time left, enter your own SQL statements and experiment with `EXPLAIN PLAN` and `AUTOTRACE`. Remember to disable `AUTOTRACE` when you have finished.

```
SQL> set autotrace off
```

## Manual Solutions for Practice 5-2: SQL Trace and TKPROF

1. Open a terminal window. Set the current directory to **labs**. Start SQL*Plus. Log in to the orcl database as the SH user with the password sh. Make sure that AUTOTRACE is disabled.

```
$ export ORACLE_SID=orcl
$ sqlplus sh/sh
SQL> SET AUTOTRACE OFF
```

2. Provide an identifier for the trace file to help you locate it. Drop all indexes (except the index of the primary key) on the CUSTOMERS table. Enable SQL Trace. Analyze the following SQL statement by using SQL Trace and TKPROF.

```
 SELECT max(cust_credit_limit)
FROM customers
WHERE cust_city ='Paris';
```

```
SQL> ALTER SESSION SET TRACEFILE_IDENTIFIER = 'OCMWS';

Session altered.

SQL> select index_name from user_indexes where table_name = 'CUSTOMERS';

INDEX_NAME
------------------------------
CUSTOMERS_YOB_BIX
CUSTOMERS_MARITAL_BIX
CUSTOMERS_GENDER_BIX
CUSTOMERS_PK

4 rows selected.

SQL> drop index CUSTOMERS_YOB_BIX;

Index dropped.

SQL> drop index CUSTOMERS_MARITAL_BIX;

Index dropped.

SQL> drop index CUSTOMERS_GENDER_BIX;

Index dropped.

SQL> drop index CUSTOMERS_PK;
drop index CUSTOMERS_PK
           *
ERROR at line 1:
```

```
ORA-02429: cannot drop index used for enforcement of unique/primary key

SQL> ALTER SESSION SET SQL_TRACE = TRUE;

Session altered.

SQL> SELECT max(cust_credit_limit)
  2  FROM customers
  3  WHERE cust_city ='Paris';

MAX(CUST_CREDIT_LIMIT)
----------------------
                 15000

1 row selected.
```

3. Now create an index on the `CUST_CITY` column, and then run the same query again.

```
SQL> CREATE INDEX cust_city_idx ON customers(cust_city);

Index created.

SQL> SELECT max(cust_credit_limit)
  2    FROM customers
  3    WHERE cust_city ='Paris';

MAX(CUST_CREDIT_LIMIT)
----------------------
                 15000
```

4. Disable tracing.

```
SQL> ALTER SESSION SET SQL_TRACE = FALSE;
```

5. Determine the location of the trace files by using the `SHOW PARAMETER DIAGNOSTIC` command and making a note of the `DIAGNOSTIC_DEST` destination.

```
SQL> show parameter diagnostic

NAME                            TYPE       VALUE
------------------------------ --------- ------------------------------
diagnostic_dest                 string      /u01/app/oracle
```

6. Exit your session.

```
SQL> exit
```

7. Change directory to the `DIAGNOSTIC_DEST` destination.

```
$ cd /u01/app/oracle/diag/rdbms/orcl/orcl/trace
```

8. Locate your file by using the file identifier that you gave in step 2. Look for a file called **orcl_ora_xxxxx_OCMWS.trc**.

```
$ more <your file name>.trc
$ tkprof <your file name>.trc output.txt aggregate=no sys=no
```

9. View the difference in the execution plans and statistics of the SQL statement with and without an index. You can use **gedit** to do this. Change back to your home directory, and then to the `labs` directory.

```
$ gedit output.txt
$ cd /home/oracle/labs
```

10. Now take a look at `DBMS_MONITOR`. Start two sessions, one connected as `SYS as SYSDBA` and the other connected as `SH`.

```
$ sqlplus / as sysdba
```

```
$ sqlplus sh/sh
```

11. From the `SYSDBA` session, determine the session ID (**`sid`**) and serial number (**`serial#`**) from `v$session` for the `SH` user, and then describe the `DBMS_MONITOR` package. Then, from the `SYSDBA` session, enable tracing by using the `sid` and `serial#` values for the other session, including the waits and bind information.

```
SQL> SELECT sid, serial# FROM v$session
  2   WHERE  username = 'SH';

       SID    SERIAL#
---------- ----------
       149      41847

SQL> EXEC DBMS_MONITOR.SESSION_TRACE_ENABLE ( -
>  session_id => 149,-
>  serial_num => 41847,-
>  waits => true,-
>  binds => true);

PL/SQL procedure successfully completed.
```

12. From the `SH` session, execute the following SQL statement, and then exit your session.

```
SELECT c.cust_last_name, t.calendar_year, sum(s.amount_sold)
  FROM sales     s JOIN
       customers c USING (cust_id) JOIN
       times t USING (time_id)
  GROUP BY c.cust_last_name, t.calendar_year
  ORDER BY c.cust_last_name, t.calendar_year;
```

```
SQL> SELECT c.cust_last_name, t.calendar_year, sum(s.amount_sold)
  2   FROM sales    s JOIN
  3        customers c USING (cust_id) JOIN
  4        times t USING (time_id)
  5   GROUP BY c.cust_last_name, t.calendar_year
  6   ORDER BY c.cust_last_name, t.calendar_year;

CUST_LAST_NAME                           CALENDAR_YEAR SUM(S.AMOUNT_SOLD)
---------------------------------------- -------------
------------------
Aaron                                             1998          46869.88
Aaron                                             1999          59321.86
Aaron                                             2000          70431.64
Aaron                                             2001          65056.16
…
```

13. From the remaining `SYSDBA` session, determine your `DIAGNOSTIC_DEST` location, locate the trace file, and view the contents. Determine the location of the trace files by using the `SHOW PARAMETER DIAGNOSTIC` command and making a note of the `DIAGNOSTIC_DEST` destination.

```
SQL>show parameter diagnostic

NAME                             TYPE       VALUE
------------------------------ ----------- ----------------------------
diagnostic_dest                  string      /u01/app/oracle
```

14. Exit your session.

```
SQL> exit
```

15. Change directory to the `DIAGNOSTIC_DEST` destination that you retrieved by using the previous query.

```
$ cd /u01/app/oracle/diag/rdbms/orcl/orcl/trace
```

16. Locate your file with the following command:

```
$ ls -ltr
```

17. View the file at the bottom that is the most recent:

```
$ more <your file name.trc>
```

18. Convert the file to readable format by using the following command:

```
$ tkprof <your file name.trc> monitor.txt sys=no
```

19. You can use `gedit` to view the file. Change back to your home directory, and then to the **`labs`** directory.

```
$ gedit monitor.txt
$ cd
$ cd labs
```

## Manual Solutions for Practice 5-3: Access Paths

1. Open a terminal window. Change the working directory to `/home/oracle/labs`. Connect to the `orcl` database as `SH` with the password `sh` by using SQL*Plus. Drop all existing indexes (except the index of the primary key) on the `CUSTOMERS` table, and then create three indexes on the following columns: `CUST_GENDER`, `CUST_POSTAL_CODE`, and `CUST_CREDIT_LIMIT`.

```
$ cd /home/oracle/labs
$ sqlplus sh/sh@orcl

SQL> select index_name from user_indexes where table_name = 'CUSTOMERS';

INDEX_NAME
------------------------------
CUST_CITY_IDX
CUSTOMERS_PK

SQL> drop index CUST_CITY_IDX;

Index dropped.

SQL> drop index CUSTOMERS_PK;
drop index CUSTOMERS_PK
           *
ERROR at line 1:
ORA-02429: cannot drop index used for enforcement of unique/primary key

SQL> CREATE INDEX cust_gender_idx ON customers(cust_gender);

Index created.

SQL> CREATE INDEX cust_postal_code_idx ON customers(cust_postal_code);

Index created.

SQL> CREATE INDEX cust_credit_limit_idx ON customers(cust_credit_limit);

Index created.
```

2. Enable `AUTOTRACE` and run the following SQL statement. The `WHERE` clause contains three predicates. Execute this statement and take note of the indexes used, the cost of the execution plan, and the amount of I/O performed.

```
SELECT c.*
  FROM customers c
  WHERE cust_gender = 'M'
    AND cust_postal_code = '40804'
    AND cust_credit_limit = 10000;
```

```
SQL> SET AUTOTRACE TRACE
SQL> SELECT c.*
  2    FROM customers c
  3    WHERE cust_gender = 'M'
  4      AND cust_postal_code = '40804'
  5      AND cust_credit_limit = 10000;

Execution Plan
----------------------------------------------------------
Plan hash value: 275081764

--------------------------------------------------------------------------------
--------------------------

| Id  | Operation                     | Name                | Rows  | Bytes
 | Cost (%CPU)| Time     |

--------------------------------------------------------------------------------
--------------------------

|   0 | SELECT STATEMENT               |                     |     6 |  1080
 |    68   (2)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID    | CUSTOMERS           |     6 |  1080
 |    68   (2)| 00:00:01 |

|   2 |   BITMAP CONVERSION TO ROWIDS   |                     |       |
 |           |          |

|   3 |    BITMAP AND                  |                     |       |
 |           |          |

|   4 |     BITMAP CONVERSION FROM ROWIDS|                     |       |
 |           |          |

|*  5 |      INDEX RANGE SCAN          | CUST_POSTAL_CODE_IDX  |    89 |
 |     1   (0)| 00:00:01 |

|   6 |     BITMAP CONVERSION FROM ROWIDS|                     |       |
 |           |          |

|*  7 |      INDEX RANGE SCAN          | CUST_CREDIT_LIMIT_IDX |    89 |
 |    14   (0)| 00:00:01 |

|   8 |     BITMAP CONVERSION FROM ROWIDS|                     |       |
 |           |          |

|*  9 |      INDEX RANGE SCAN          | CUST_GENDER_IDX      |    89 |
 |    51   (0)| 00:00:01 |

--------------------------------------------------------------------------------
--------------------------


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

   5 - access("CUST_POSTAL_CODE"='40804')
   7 - access("CUST_CREDIT_LIMIT"=10000)
   9 - access("CUST_GENDER"='M')


Statistics
----------------------------------------------------------
        546  recursive calls
          0  db block gets
        237  consistent gets
         86  physical reads
          0  redo size
```

```
       3369  bytes sent via SQL*Net to client
        524  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         27  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

3. Drop the indexes (except the index of the primary key) again and replace them with a single concatenated index of three columns `CUST_GENDER`, `CUST_POSTAL_CODE`, and `CUST_CREDIT_LIMIT`. Then run the SQL statement of step 2 again.

```
SQL> SET AUTOTRACE OFF;
SQL> select index_name from user_indexes where table_name = 'CUSTOMERS';

INDEX_NAME
------------------------------
CUST_CREDIT_LIMIT_IDX
CUST_POSTAL_CODE_IDX
CUST_GENDER_IDX
CUSTOMERS_PK

SQL> drop index CUST_CREDIT_LIMIT_IDX;

Index dropped.

SQL> drop index CUST_POSTAL_CODE_IDX;

Index dropped.

SQL> drop index CUST_GENDER_IDX;

Index dropped.

SQL> CREATE INDEX cust_gen_pos_credit_idx
  2   ON customers(cust_gender, cust_postal_code, cust_credit_limit);

Index created.
SQL> SET AUTOTRACE TRACE
SQL> SELECT c.*
  2   FROM customers c
  3   WHERE cust_gender = 'M'
  4     AND cust_postal_code = '40804'
  5     AND cust_credit_limit = 10000;

6 rows selected.

Execution Plan
------------------------------------------------------------
Plan hash value: 823835768

-----------------------------------------------------------------------------------
-----------------------

| Id  | Operation                  | Name                    | Rows  | Bytes |
Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
----------------------

|   0 | SELECT STATEMENT          |                      |     7 |  1260 |
    8   (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS            |     7 |  1260 |
    8   (0)| 00:00:01 |

|*  2 |   INDEX RANGE SCAN        | CUST_GEN_POS_CREDIT_IDX |     7 |       |
    1   (0)| 00:00:01 |

--------------------------------------------------------------------------------
----------------------

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

   2 - access("CUST_GENDER"='M' AND "CUST_POSTAL_CODE"='40804' AND "CUST_CREDIT_
LIMIT"=10000)

Statistics
----------------------------------------------------------
         31  recursive calls
          0  db block gets
         44  consistent gets
          1  physical reads
          0  redo size
       3369  bytes sent via SQL*Net to client
        524  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

4. Drop all indexes (except the index of the primary key) on the CUSTOMERS table. Create three bitmapped indexes on CUST_GENDER, CUST_POSTAL_CODE, and CUST_CREDIT_LIMIT, and then run the SQL statement of step 2 again. This statement has a complicated WHERE clause. Bitmapped indexes are good for this type of statement. You see several bitmap operations in the execution plan.

```
SQL> SET AUTOTRACE OFF
SQL> select index_name from user_indexes where table_name = 'CUSTOMERS';

INDEX_NAME
------------------------------
CUST_GEN_POS_CREDIT_IDX
CUSTOMERS_PK

SQL> drop index CUST_GEN_POS_CREDIT_IDX;

Index dropped.

SQL> CREATE BITMAP INDEX cust_gender_bidx ON customers(cust_gender);

Index created.
```

```
SQL> CREATE BITMAP INDEX cust_postal_code_bidx ON
customers(cust_postal_code);

Index created.

SQL> CREATE BITMAP INDEX cust_credit_limit_bidx ON
customers(cust_credit_limit);

Index created.
SQL> SET AUTOTRACE TRACE
SQL> SELECT c.*
  2    FROM customers c
  3    WHERE cust_gender = 'M'
  4      AND cust_postal_code = '40804'
  5      AND cust_credit_limit = 10000;

6 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2458303217

--------------------------------------------------------------------------------
-----------------------

| Id  | Operation                  | Name                 | Rows  | Bytes |
Cost (%CPU)| Time     |

--------------------------------------------------------------------------------
-----------------------

|   0 | SELECT STATEMENT           |                      |     6 |  1080 |
    4   (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID | CUSTOMERS            |     6 |  1080 |
    4   (0)| 00:00:01 |

|   2 |   BITMAP CONVERSION TO ROWIDS|                      |       |       |
         |          |

|   3 |    BITMAP AND              |                      |       |       |
         |          |

|*  4 |     BITMAP INDEX SINGLE VALUE| CUST_POSTAL_CODE_BIDX  |       |       |
         |          |

|*  5 |     BITMAP INDEX SINGLE VALUE| CUST_CREDIT_LIMIT_BIDX |       |       |
         |          |

|*  6 |     BITMAP INDEX SINGLE VALUE| CUST_GENDER_BIDX      |       |       |
         |          |

--------------------------------------------------------------------------------
-----------------------

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

```
   4 - access("CUST_POSTAL_CODE"='40804')
   5 - access("CUST_CREDIT_LIMIT"=10000)
   6 - access("CUST_GENDER"='M')

Statistics
----------------------------------------------------------
         31  recursive calls
          0  db block gets
         50  consistent gets
          5  physical reads
          0  redo size
       3369  bytes sent via SQL*Net to client
        524  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

5. Finally, investigate the benefits of function-based indexes. Drop all indexes on the CUSTOMERS table. First, create a normal index on the CUST_LAST_NAME column and run the following SQL statement:

```
SELECT cust_id, country_id
FROM customers
WHERE lower( cust_last_name) like 'gentle';
```

Create a function-based index that utilizes the LOWER function on the CUST_LAST_NAME column.

```
SQL> SET AUTOTRACE OFF
SQL> select index_name from user_indexes where table_name = 'CUSTOMERS';

INDEX_NAME
------------------------------
CUST_CREDIT_LIMIT_BIDX
CUST_POSTAL_CODE_BIDX
CUST_GENDER_BIDX
CUSTOMERS_PK

SQL> drop index CUST_CREDIT_LIMIT_BIDX;

Index dropped.

SQL> drop index CUST_POSTAL_CODE_BIDX;

Index dropped.

SQL> drop index CUST_GENDER_BIDX;

Index dropped.

SQL> CREATE INDEX lower_cust_last_name_idx ON customers(cust_last_name);
```

```
Index created.

SQL> SET AUTOTRACE TRACE
SQL> SELECT cust_id, country_id
  2    FROM customers
  3    WHERE lower( cust_last_name) like 'gentle';

80 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 2008213504

-------------------------------------------------------------------------------
| Id  | Operation        | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |   555 |  9990 |   405   (1)| 00:00:05 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |   555 |  9990 |   405   (1)| 00:00:05 |
-------------------------------------------------------------------------------

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

   1 - filter(LOWER("CUST_LAST_NAME")='gentle')

Statistics
----------------------------------------------------------
       385  recursive calls
         0  db block gets
      1631  consistent gets
         0  physical reads
         0  redo size
      2492  bytes sent via SQL*Net to client
       579  bytes received via SQL*Net from client
         7  SQL*Net roundtrips to/from client
        12  sorts (memory)
         0  sorts (disk)
        80  rows processed
```

```
SQL> CREATE INDEX lower_cust_last_name_fidx ON
customers(LOWER(cust_last_name));

Index created.

SQL> SET AUTOTRACE TRACE
SQL> SELECT cust_id, country_id
  2    FROM customers
  3    WHERE lower( cust_last_name) like 'gentle';

80 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 744516409

--------------------------------------------------------------------------------
-------------------------
```

```
| Id  | Operation                   | Name                     | Rows  | Bytes
| Cost (%CPU)| Time     |

--------------------------------------------------------------------------------
-------------------------

|   0 | SELECT STATEMENT            |                          |   555 | 17760
|    41   (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                |   555 | 17760
|    41   (0)| 00:00:01 |

|*  2 |   INDEX RANGE SCAN          | LOWER_CUST_LAST_NAME_FIDX |   222 |
|     1   (0)| 00:00:01 |

--------------------------------------------------------------------------------
-------------------------

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

   2 - access(LOWER("CUST_LAST_NAME")='gentle')

Statistics
----------------------------------------------------------
         24  recursive calls
          0  db block gets
         24  consistent gets
          1  physical reads
          0  redo size
       2492  bytes sent via SQL*Net to client
        579  bytes received via SQL*Net from client
          7  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         80  rows processed

SQL> set autot off
```

## Manual Solutions for Practice 5-4: Gathering Optimizer Statistics

1. Connect to the `orcl` database as the `sh` user and create a `MY_CUST` table that is identical to the `CUSTOMERS` table.

   `CREATE table my_cust AS SELECT * FROM customers;`

```
SQL> connect sh/sh
SQL> CREATE table my_cust AS SELECT * FROM customers;
```

2. Query `USER_TABLES` to verify the existence of statistics for `MY_CUST` table.

```
SQL> SELECT last_analyzed analyzed, sample_size,monitoring,
  2  table_name
  3  FROM user_tables
  4  WHERE table_name = 'MY_CUST';

ANALYZED  SAMPLE_SIZE MON TABLE_NAME
--------- ----------- --- -------------------------
                      YES MY_CUST
```

3. Run the following query on the table, and then view the execution plan by using `AUTOTRACE TRACEONLY EXPLAIN`.
   `SELECT * FROM my_cust WHERE cust_id < 50;`

```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT * FROM my_cust WHERE cust_id < 50;
Execution Plan
----------------------------------------------------------
Plan hash value: 2970555845
-----------------------------------------------------------------------------
-
| Id  | Operation         | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT  |         |    16 |  4768 |   396   (1)| 00:00:05 |
|*  1 |  TABLE ACCESS FULL| MY_CUST |    16 |  4768 |   396   (1)| 00:00:05 |
-----------------------------------------------------------------------------
-

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

   1 - filter("CUST_ID"<50)

Note
-----
   - dynamic sampling used for this statement (level=2)
SQL>
```

4. Disable `AUTOTRACE`. Create an index `MY_CUST_ID_IDX` on the `CUST_ID` column.

```
SQL> SET AUTOTRACE OFF
SQL> CREATE INDEX my_cust_id_idx ON my_cust(cust_id);

Index created.
```

5. Run the query again and view the execution plan by using `AUTOTRACE`.

```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT * FROM my_cust WHERE cust_id < 50;
Execution Plan
----------------------------------------------------------
Plan hash value: 3210149128
-----------------------------------------------------------------------------
----
--------------

| Id  | Operation                   | Name           | Rows  | Bytes | Cost (%CP
U)| Time     |

-----------------------------------------------------------------------------
--------------

|   0 | SELECT STATEMENT            |                |    49 | 14602 |    57   (
0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| MY_CUST        |    49 | 14602 |    57   (
0)| 00:00:01 |

|*  2 |   INDEX RANGE SCAN          | MY_CUST_ID_IDX |    49 |       |     2   (
0)| 00:00:01 |

-----------------------------------------------------------------------------
----
--------------

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

   2 - access("CUST_ID"<50)

Note
-----
   - dynamic sampling used for this statement (level=2)
```

6. How does the optimizer know to use the index?

**Hint:** Query `USER_INDEXES` and `USER_TABLES` by using `tabstats.sql` and `indstats.sql` to get the answer.

```
SQL> SET AUTOTRACE OFF
SQL> get tabstats.sql
  1  accept table_name  -
  2       prompt 'on which table : '
  3  SELECT last_analyzed analyzed, sample_size, monitoring,
  4        table_name
  5  FROM user_tables
  6  WHERE table_name = upper('&table_name');
  7* undef  TABLE_NAME
  8
SQL> @tabstats
on which table : my_cust
old   4: WHERE table_name = upper('&table_name')
new   4: WHERE table_name = upper('my_cust')

ANALYZED  SAMPLE_SIZE MON TABLE_NAME
--------- ----------- --- ------------------------------
                      YES MY_CUST

SQL> get indstats.sql
  1  accept table_name  -
  2       prompt 'on which table : '
  3  SELECT index_name name, num_rows n_r,
  4  last_analyzed l_a, distinct_keys d_k,
  5  leaf_blocks l_b,   avg_leaf_blocks_per_key a_l,join_index j_I
  6  FROM user_indexes
  7  WHERE table_name = upper('&table_name');
  8* undef table_name
  9
SQL> @indstats
on which table : my_cust
old   5: WHERE table_name = upper('&table_name')
new   5: WHERE table_name = upper('my_cust')

NAME                                  N_R L_A              D_K        L_B
------------------------------ ---------- --------- ---------- ----------
----------
       A_L J_I
---------- ---
MY_CUST_ID_IDX                      55501 28-NOV-10      55501        123
         1 NO
```

7. Drop the index that you created. Then verify the index statistics again. What do you see?

```
SQL> DROP INDEX my_cust_id_idx;
Index dropped.

SQL> @indstats.sql
on which table : my_cust
old   5: WHERE table_name = upper('&table_name')
new   5: WHERE table_name = upper('my_cust')
no rows selected
```

8. Identify the last analyzed date and sample size for all the tables in your schema.

```
SQL> SELECT last_analyzed analyzed, sample_size, monitoring, table_name
  2   FROM user_tables;

ANALYZED  SAMPLE_SIZE MON TABLE_NAME
--------- ----------- --- ------------------------------
20-JAN-09        2000 YES SALES
20-JAN-09        2000 YES COSTS
20-JAN-09        2000 YES TIMES
20-JAN-09        2000 YES PRODUCTS
20-JAN-09        2000 YES CHANNELS
20-JAN-09        2000 YES PROMOTIONS
20-JAN-09        2000 YES CUSTOMERS
20-JAN-09        2000 YES COUNTRIES
20-JAN-09        2000 YES SUPPLEMENTARY_DEMOGRAPHICS
20-JAN-09        2000 YES MVIEW$_EXCEPTIONS
20-JAN-09        2000 YES CAL_MONTH_SALES_MV
20-JAN-09        2000 YES FWEEK_PSCAT_SALES_MV
                      NO  SALES_DELTA_XT
                      NO  SALES_CH
                      YES MV1
                      YES CUST_ID_SALES_AGGR
                      YES ORDERS
                      YES ORDER_ITEMS
                      YES NEWSALES
                      YES MY_CUST

20 rows selected.
```

9. Identify the types of histograms for all the columns in your schema.

**Hint:** Query USER_TAB_COL_STATISTICS.

```
SQL> SELECT column_name, num_distinct, num_buckets, histogram
  2   FROM USER_TAB_COL_STATISTICS
  3   WHERE histogram <> 'NONE';

COLUMN_NAME                  NUM_DISTINCT NUM_BUCKETS HISTOGRAM
------------------------------ ------------ ----------- ---------------
CHANNEL_TOTAL_ID                        1           1 FREQUENCY
CHANNEL_CLASS_ID                        3           3 FREQUENCY
…
```

10. Gather statistics for all tables on the SH schema.

```
SQL> EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS('SH');
PL/SQL procedure successfully completed.
SQL> SELECT last_analyzed analyzed, sample_size, monitoring, table_name
  2  FROM user_tables;
ANALYZED  SAMPLE_SIZE MON TABLE_NAME
--------- ----------- --- ------------------------------
28-NOV-10      994828 YES SALES
```

```
28-NOV-10      82112 YES COSTS
28-NOV-10       1826 YES TIMES
28-NOV-10         72 YES PRODUCTS
28-NOV-10          5 YES CHANNELS
28-NOV-10        503 YES PROMOTIONS
28-NOV-10      55501 YES CUSTOMERS
28-NOV-10         23 YES COUNTRIES
28-NOV-10       4500 YES SUPPLEMENTARY_DEMOGRAPHICS
28-NOV-10          0 YES MVIEW$_EXCEPTIONS
28-NOV-10         48 YES CAL_MONTH_SALES_MV
28-NOV-10      11266 YES FWEEK_PSCAT_SALES_MV
                       NO  SALES_DELTA_XT
                       NO  SALES_CH
28-NOV-10       7059 YES MV1
28-NOV-10       7059 YES CUST_ID_SALES_AGGR
28-NOV-10          0 YES ORDERS
28-NOV-10          0 YES ORDER_ITEMS
28-NOV-10         10 YES NEWSALES
28-NOV-10      55501 YES MY_CUST
```

11. Now consider histograms. First, flush the shared pool. Then run `lab_05_04_11.sql` that creates the `NEW_CUST` table and populates it with skewed data. It also creates an index and a histogram on the skew data column `CUST_ID`. After this script is run, the `CUST_ID` column has 1,000 rows with a value of 1, one row with a value of 2, and one row with a value of 3.

```
SQL> connect / as sysdba
SQL> ALTER SYSTEM FLUSH SHARED_POOL;

System altered.
SQL> connect sh/sh
SQL> get lab_05_04_11.sql
  1  column operations format a20
  2  column object_name format a20
  3  column options format a20
  4  DROP TABLE new_cust;
  5  CREATE TABLE new_cust(cust_id number, ord_total number)
  6  /
  7  --Now fill cust_id with 10000 rows with value 1, one row with 2 and
one with 3
  8  DECLARE
  9  i integer;
 10  BEGIN
 11    FOR i in 1..10000 LOOP
 12      INSERT INTO new_cust VALUES(1, i);
 13    END LOOP;
 14    INSERT INTO new_cust values(2, 10000);
 15    INSERT INTO new_cust values(3, 1500);
 16    COMMIT;
 17  END;
 18  /
 19  --Create an index on cust_id
 20  CREATE INDEX new_cust_cust_id ON new_cust(cust_id)
 21  /
 22  --Create histogranm with 200 buckets
 23  BEGIN
 24  dbms_stats.gather_table_stats(ownname => 'SH', tabname =>
'new_cust',
 25  method_opt => 'FOR ALL COLUMNS SIZE 200');
 26* END;
 27
SQL> @lab_05_04_11
DROP TABLE new_cust
           *
ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

PL/SQL procedure successfully completed.

Index created.

PL/SQL procedure successfully completed.

SQL>
```

Now run the following statement and use `AUTOTRACE` to get the execution plan:

`SELECT count(ord_total) FROM new_cust where cust_id = 1;`

```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT count(ord_total) FROM new_cust where cust_id = 1;

Execution Plan
----------------------------------------------------------
Plan hash value: 965923628

-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |     1 |     7 |     7   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |          |     1 |     7 |            |          |
|*  2 |   TABLE ACCESS FULL| NEW_CUST | 10000 | 70000 |     7   (0)| 00:00:01 |
-------------------------------------------------------------------------------

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

   2 - filter("CUST_ID"=1)
```

You see that this is a full table scan. Now try running the following SQL. Is the index used?

`SELECT count(ord_total) FROM new_cust where cust_id = 2;`

```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT count(ord_total) FROM new_cust where cust_id = 2;

Execution Plan
```

```
-----------------------------------------------------------
Plan hash value: 2084750850

---------------------------------------------------------------------------------
-----------------

| Id  | Operation                 | Name             | Rows  | Bytes | Cost (
%CPU)| Time     |

---------------------------------------------------------------------------------
-----------------

|   0 | SELECT STATEMENT           |                  |     1 |     7 |     2
  (0)| 00:00:01 |

|   1 |  SORT AGGREGATE            |                  |     1 |     7 |
    |          |

|   2 |   TABLE ACCESS BY INDEX ROWID| NEW_CUST         |     1 |     7 |     2
  (0)| 00:00:01 |

|*  3 |    INDEX RANGE SCAN         | NEW_CUST_CUST_ID |     1 |       |     1
  (0)| 00:00:01 |

---------------------------------------------------------------------------------
-----------------


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

   3 - access("CUST_ID"=2)
```

The optimizer uses the histogram to determine whether to use an index.

## Manual Solutions for Practice 5-5: Using Deferred Optimizer Statistics

In this practice, you manipulate both deferred statistics publishing and statistics extensions. The basic idea of this practice is to test various statistic gatherings on a particular table before publishing the best ones in your production environment.

1. Confirm whether `ORACLE_SID` was set to `orcl`. Execute the `stats_setup.sh` script. This script creates a new user called `STATS` and creates and populates a new table called `STATS.TABJFV`.

```
$ cd /home/oracle/labs
$ echo $ORACLE_SID
orcl

$ ./stats_setup.sh
...
SQL> SQL> SQL> SQL> drop user stats cascade
          *
ERROR at line 1:
ORA-01918: user 'STATS' does not exist

SQL> SQL>
User created.

SQL> SQL>
Grant succeeded.

SQL> SQL> Connected.
SQL> SQL> drop table tabjfv purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist
SQL> SQL> SQL>   2
Table created.

SQL> SQL> SQL> SQL>   2    3    4    5    6    7    8
PL/SQL procedure successfully completed.

SQL>
Commit complete.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.1.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
$
```

2. Start a SQL*Plus session connected as user `STATS`. ***Do not disconnect from that session***. Make sure that you delete all existing statistics on `STATS.TABJFV` and verify that none exist either as public statistics or as pending statistics. Use the `delete_stats.sql` script.

```
$ sqlplus stats/stats
```

```
SQL> @delete_stats
Session altered.

PL/SQL procedure successfully completed.
old   3: where table_name = '&1'
new   3: where table_name = 'TABJFV'

TABLE_NAME                 ANALYZE_TIME    NUM_ROWS    BLOCKS
AVG_ROW_LEN
----------------- -------------- ---------- ---------- -----------
TABJFV
old   4: where table_name = '&1'
new   4: where table_name = 'TABJFV'

no rows selected

old   4: where table_name = '&1'
new   4: where table_name = 'TABJFV'

COLUMN_NAME                ANALYZE_TIME   NUM_DISTINCT  NUM_NULLS
DENSITY
------------------------------ -------------- ------------ ----------
----------
C1
C2
C3
C4
C5
C6
C7
C8
C9

9 rows selected.
SQL>
SQL>
SQL> -- private statistics: none
SQL> @ show_pending_stats TABJFV
SQL>
SQL> set echo off
old   3: where table_name = '&1' and partition_name is null
new   3: where table_name = 'TABJFV' and partition_name is null
no rows selected
old   4: where table_name = '&1' and partition_name is null
new   4: where table_name = 'TABJFV' and partition_name is null

no rows selected

old   4: where table_name = '&1' and partition_name is null
new   4: where table_name = 'TABJFV' and partition_name is null
```

```
no rows selected

SQL>
```

3. Determine the publishing mode for `STATS.TABJFV` statistics, and set it to `PENDING` mode.

```
SQL> select dbms_stats.get_prefs('PUBLISH', 'stats', 'tabjfv') publish
from dual;

PUBLISH
----------------------------------------------------------------------
-----------
TRUE

SQL> exec dbms_stats.set_table_prefs('stats', 'tabjfv', 'PUBLISH',
'FALSE');

PL/SQL procedure successfully completed.

SQL> select dbms_stats.get_prefs('PUBLISH', 'stats', 'tabjfv') publish
from dual;

PUBLISH
----------------------------------------------------------------------
-----------
FALSE

SQL>
```

4. Collect statistics on `STATS.TABJFV` and investigate the result. What do you observe? Use the `collect_pending.sql` script.

```
SQL> @collect_pending
SQL> -- Collect stats
SQL> begin
  2   dbms_stats.gather_table_stats(null, 'tabjfv',
  3     method_opt => 'for all columns size 1 for columns c1 size 254 for
columns c2 size 254');
  4  end;
  5  /

PL/SQL procedure successfully completed.

SQL>
SQL> -- public statistics: still nothing
SQL> @ show_public_stats  TABJFV
SQL>
SQL> set echo off
old   3: where table_name = '&1'
new   3: where table_name = 'TABJFV'
```

```
TABLE_NAME                     ANALYZE_TIME     NUM_ROWS     BLOCKS
AVG_ROW_LEN
------------------------------ -------------- ---------- ----------
-----------
TABJFV

old   4: where table_name = '&1'
new   4: where table_name = 'TABJFV'

no rows selected

old   4: where table_name = '&1'
new   4: where table_name = 'TABJFV'

COLUMN_NAME                    ANALYZE_TIME   NUM_DISTINCT  NUM_NULLS
DENSITY
------------------------------ -------------- ------------ ----------
----------
C1
C2
C3
C4
C5
C6
C7
C8
C9

9 rows selected.

SQL>
SQL>
SQL> -- private statistics: should see values (0.001 for density for both
C1 and C2, 0.2 for others)
SQL> @ show_pending_stats TABJFV
SQL>
SQL> set echo off
old   3: where table_name = '&1' and partition_name is null
new   3: where table_name = 'TABJFV' and partition_name is null

TABLE_NAME                     analyze time     NUM_ROWS     BLOCKS AVG_ROW_LEN
------------------------------ -------------- ---------- ----------
-----------
TABJFV                        11/29 13:10:02       500          3         27

old   4: where table_name = '&1' and partition_name is null
new   4: where table_name = 'TABJFV' and partition_name is null

no rows selected

old   4: where table_name = '&1' and partition_name is null
new   4: where table_name = 'TABJFV' and partition_name is null
```

```
COLUMN_NAME                   analyze time   NUM_DISTINCT  NUM_NULLS
DENSITY
----------------------------- -------------- ------------ ----------
----------
C1                          11/29 13:10:02          5         0      .001
C2                          11/29 13:10:02          5         0      .001
C3                          11/29 13:10:02          5         0        .2
C4                          11/29 13:10:02          5         0        .2
C5                          11/29 13:10:02          5         0        .2
C6                          11/29 13:10:02          5         0        .2
C7                          11/29 13:10:02          5         0        .2
C8                          11/29 13:10:02          5         0        .2
C9                          11/29 13:10:02          5         0        .2

9 rows selected.

SQL>
SQL>
SQL> -- Should be false: use public statistics
SQL> show parameter optimizer_use_pending_statistics

NAME                                 TYPE       VALUE
------------------------------------ -----------
------------------------------
optimizer_use_pending_statistics     boolean    FALSE
SQL>
SQL>
```

5. From a terminal window, connected as the `STATS` user in a SQL*Plus session (do not exit this session after this step), disable dynamic sampling for your session and determine the number of rows that the optimizer can currently estimate for the following query:

   `select count(*) from tabjfv where c1 = 1 and c2 = 1;`
   What is your conclusion?

   **Hint:** Use an explain plan to view the execution plan and SQL statement as follows:

   ```
   select plan_table_output
   from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));
   ```

   a) Because you are not using dynamic sampling, and because there are currently no public statistics for your table, the optimizer makes a wrong guess on the number of rows that the query returns: It is using 1 instead of 100.

```
SQL> alter session set optimizer_dynamic_sampling = 0;

Session altered.

SQL> explain plan for
  2  select count(*) from tabjfv where c1 = 1 and c2 = 1;

Explained.
```

```
SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------
-----------
Plan hash value: 3122694931

---------------------------------------------
| Id  | Operation          | Name   | Rows  |
---------------------------------------------
|   0 | SELECT STATEMENT   |        |     1 |
|   1 |  SORT AGGREGATE    |        |     1 |
|   2 |   TABLE ACCESS FULL| TABJFV |     1 |
---------------------------------------------

9 rows selected.
```

6. Now, switch your session to use the pending statistics that were previously collected.

```
SQL> alter session set optimizer_use_pending_statistics = true;

Session altered.
```

7. Determine again the optimizer's estimation of the number of rows returned by your query. What do you observe?
   a) Again, the optimizer picks up a wrong estimation. This is due to the fact that the optimizer is unable to guess that `C1` and `C2` are highly correlated for value 1.

```
SQL> explain plan for
  2  select count(*) from tabjfv where c1 = 1 and c2 = 1;

Explained.

SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------
-----------
Plan hash value: 3122694931

---------------------------------------------
| Id  | Operation          | Name   | Rows  |
---------------------------------------------
|   0 | SELECT STATEMENT   |        |     1 |
|   1 |  SORT AGGREGATE    |        |     1 |
|   2 |   TABLE ACCESS FULL| TABJFV |    20 |
---------------------------------------------

9 rows selected.
```

8. Create a statistics extension to groups `C1` and `C2` to indicate that both columns are correlated in `STATS.TABJFV`. When done, gather statistics again on your table with maximum precision for your extension.

```
SQL> col table_name for a10
SQL> col extension for a12
SQL> select
  2   dbms_stats.create_extended_stats(null,'tabjfv','(c1,c2)')
  3   from dual;
BMS_STATS.CREATE_EXTENDED_STATS(NULL,'TABJFV','(C1,C2)')
----------------------------------------------------------------------
SYS_STUF3GLKIOP5F4B0BTTCFTMX0W

SQL> select * from user_stat_extensions;

TABLE_NAME EXTENSION_NAME                 EXTENSION    CREATO DRO
---------- ------------------------------ ------------ ------ ---
TABJFV     SYS_STUF3GLKIOP5F4B0BTTCFTMX0W ("C1","C2")  USER   YES

SQL> begin
  2  dbms_stats.gather_table_stats(null, 'tabjfv',
  3   method_opt => 'for all columns size 1 for columns (c1,c2) size 254');
  4  end;
  5  /

PL/SQL procedure successfully completed.
```

9. Determine again the optimizer's estimation of the number of rows returned by your query. What do you observe?

```
SQL> explain plan for
  2  select count(*) from tabjfv where c1 = 1 and c2 = 1;

Explained.

SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------
-----------
Plan hash value: 3122694931

---------------------------------------------
| Id  | Operation         | Name   | Rows  |
---------------------------------------------
|   0 | SELECT STATEMENT   |        |     1 |
|   1 |  SORT AGGREGATE    |        |     1 |
|   2 |   TABLE ACCESS FULL| TABJFV |   100 |
---------------------------------------------
9 rows selected.
```

10. Execute `stats_cleanup.sh` to clean up your environment.

```
$ ./stats_cleanup.sh


SQL*Plus: Release 11.2.0.3.0 Production on Mon Mar 11 21:31:33 2013

Copyright (c) 1982, 2011, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL>
User dropped.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
$
```

## Manual Solutions for Practice 5-6: Using Result Cache

In this practice, you explore the various possibilities to cache query results in the SGA. Perform the following steps to understand the use of Query Result Cache.

1. Change to the `$HOME/labs` directory and execute the `result_cache_setup.sh` script.

```
$ cd /home/oracle/labs
$ export ORACLE_SID=orcl
$./result_cache_setup.sh

SQL*Plus: Release 11.2.0.3.0 Production on Mon Mar 11 21:32:34 2013
Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL> drop user qrc cascade
          *
ERROR at line 1:
ORA-01918: user 'QRC' does not exist


SQL> SQL>   2    3
User created.

SQL> SQL>
Grant succeeded.

SQL> SQL> Connected.
SQL> SQL>
PL/SQL procedure successfully completed.

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

SQL> SQL>
Table created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
2 rows created.

…
```

```
SQL>
524288 rows created.

SQL> SQL>
1 row created.

SQL> SQL>
Commit complete.

SQL> SQL>
System altered.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edaor2p3-orcl labs]$
```

2. In your terminal window, log in to SQL*Plus as the `QRC` user. From now on, do not disconnect from this session. Determine the current content of the query cache by using the following statement:

```
select type,status,name,object_no,row_count,row_size_avg
from v$result_cache_objects order by 1;
```

```
$ sqlplus qrc/qrc

SQL> select type,status,name,object_no,row_count,row_size_avg
  2  from v$result_cache_objects order by 1;

no rows selected
```

3. Set timing on and execute the following query. You can use the `rc_query1.sql` script. Note the time that it takes to execute.

```
select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,
cachejfv c5
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b'
and c5.c='b';
```

```
SQL> @rc_query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.37
```

4. Determine the execution plan of the previous query by using `explain_rc_query1.sql`. What do you observe?

```
SQL> @explain_rc_query1
SQL> set echo on
SQL>
SQL> explain plan for select /*+ result_cache q_name(Q1) */ count(*) from
cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv c5 where
c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b';

Explained.

Elapsed: 00:00:00.01
SQL>
SQL> set linesize 180
SQL> set echo off

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------
-------------------------------------------
Plan hash value: 2522916280

----------------------------------------------------------------------
------------------------------------
| Id  | Operation                | Name                       | Rows  | Bytes
| Cost (%CPU)| Time     |
----------------------------------------------------------------------
------------------------------------
|   0 | SELECT STATEMENT         |                            |     1 |  1260
|  2291G  (1)|999:59:59 |
|   1 |  RESULT CACHE            | g5nsjhgmqsdsg5kwuqq5jz5m1u |       |
|           |           |
|   2 |   SORT AGGREGATE         |                            |     1 |  1260 |
|           |
|   3 |    MERGE JOIN CARTESIAN   |                           |   92G|
105T|  2291G  (1)|999:59:59 |
|   4 |     MERGE JOIN CARTESIAN  |                           |   590M|
554G|    14G  (1)|999:59:59 |
|   5 |      MERGE JOIN CARTESIAN |                           |  3789K|
2732M|    94M  (1)|314:14:19 |

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------
----------------------------------------------------------------------
-------------------------------------------
|   6 |       MERGE JOIN CARTESIAN|                           | 24306 |
11M|   605K  (1)| 02:01:01 |
|*  7 |        TABLE ACCESS FULL  | CACHEJFV                  |   156 | 39312
|  3856   (1)| 00:00:47 |
|   8 |        BUFFER SORT        |                           |   156 | 39312
|   601K  (1)| 02:00:15 |
|*  9 |        TABLE ACCESS FULL | CACHEJFV                   |   156 | 39312
|  3854   (1)| 00:00:47 |
|  10 |       BUFFER SORT        |                            |   156 | 39312
|    94M  (1)|314:13:32 |
|* 11 |        TABLE ACCESS FULL  | CACHEJFV                  |   156 | 39312
|  3854   (1)| 00:00:47 |
|  12 |      BUFFER SORT         |                            |   156 | 39312
|    14G  (1)|999:59:59 |
|* 13 |       TABLE ACCESS FULL   | CACHEJFV                  |   156 | 39312
|  3854   (1)| 00:00:47 |
|  14 |     BUFFER SORT          |                            |   156 | 39312
|  2291G  (1)|999:59:59 |
|* 15 |      TABLE ACCESS FULL    | CACHEJFV                  |   156 | 39312
|  3854   (1)| 00:00:47 |
---------------------------------------------------------------------
-----------------------------------

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
---------------------------------------------------------------------
------------------------------------------

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

   7 - filter("C1"."C"='b')
   9 - filter("C2"."C"='b')
  11 - filter("C3"."C"='b')
  13 - filter("C4"."C"='b')
  15 - filter("C5"."C"='b')

Result Cache Information (identified by operation id):

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
------------------------------------------
------------------------------------------------------


   1 - column-count=1; dependencies=(QRC.CACHEJFV);
attributes=(single-row); parameters=(nls); name="select /*+
result_cache q_name(Q1) */ count(*) from cachejfv c1,cachejfv c2,cac
hejfv c3,cachejfv c4,cachejfv c5 where c1.c='b' a"


Note
-----
   - dynamic sampling used for this statement (level=2)
40 rows selected.

Elapsed: 00:00:00.02
```

5. Determine the current content of the query cache. What do you observe?

```
SQL>  SELECT type,status,name,object_no,row_count,row_size_avg
  2  FROM v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- ---------
----------------------------------------------------------------------
------------------------------------------------------------ ----------
----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
18061        0
         0

Result     Published select /*+ result_cache q_name(Q1) */ count(*)
0        1
                 from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
                 where c1.c='b' a
         5
```

6. Flush the buffer cache of your instance and rerun the query executed in step 3. What do you observe?

```
SQL> alter system flush buffer_cache;
System altered.

Elapsed: 00:00:00.02
SQL> @rc_query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b';

  COUNT(*)
----------
       1

Elapsed: 00:00:00.00
SQL>
```

7. Insert a new row into the `CACHEJFV` table by using the following statement:

   ```
   insert into cachejfv values('c');
   commit;
   ```

   What do you observe?

   a) The corresponding result cache entry is automatically invalidated.

```
SQL> insert into cachejfv values('c');

1 row created.

Elapsed: 00:00:00.00
SQL> commit;

Commit complete.

Elapsed: 00:00:00.00
SQL> select type,status,name,object_no,row_count,row_size_avg
  2  from v$result_cache_objects order by 1;

TYPE      STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- ---------
-------------------------------------------------------------------
-------------------------------------------------------------- ---------
----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
18061        0
         0

Result     Invalid   select /*+ result_cache q_name(Q1) */ count(*)
0        1
                 from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv
c4,cachejfv c5
                 where c1.c='b' a
         5


Elapsed: 00:00:00.00
SQL>
```

8. Execute your first query again and check the result cache. What do you observe?

```
SQL> @rc_query1
SQL> set echo on
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b';

  COUNT(*)
----------
       1

Elapsed: 00:00:00.34
SQL>
```

9. Generate a detailed result cache memory report.

```
SQL> set serveroutput on
SQL> EXEC DBMS_RESULT_CACHE.MEMORY_REPORT(detailed=>true);
R e s u l t   C a c h e   M e m o r y   R e p o r t
[Parameters]
Block Size         = 1K bytes
Maximum Cache Size  = 2M bytes (2K blocks)
Maximum Result Size = 102K bytes (102 blocks)
[Memory]
Total Memory = 174752 bytes [0.077% of the Shared Pool]
... Fixed Memory = 10696 bytes [0.005% of the Shared Pool]
....... Memory Mgr = 200 bytes
....... Bloom Fltr = 2K bytes
....... Cache Mgr  = 5552 bytes
....... State Objs = 2896 bytes
... Dynamic Memory = 164056 bytes [0.072% of the Shared Pool]
....... Overhead = 131288 bytes
........... Hash Table    = 64K bytes (4K buckets)
........... Chunk Ptrs    = 24K bytes (3K slots)
........... Chunk Maps    = 12K bytes
........... Miscellaneous = 28888 bytes
....... Cache Memory = 32K bytes (32 blocks)
........... Unused Memory = 29 blocks
........... Used Memory = 3 blocks
............... Dependencies = 1 blocks (1 count)
............... Results = 2 blocks
................... SQL    = 1 blocks (1 count)
................... Invalid = 1 blocks (1 count)

PL/SQL procedure successfully completed.
```

10. Execute your first query again. What do you observe?

    a) The query again uses the result that was previously cached.

```
SQL> @rc_query1
SQL> set echo on
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv c5
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.01
```

11. Clear the result cache. Query `V$RESULT_CACHE_OBJECTS` to verify the clear operation.

```
SQL> exec dbms_result_cache.flush;

PL/SQL procedure successfully completed.
```

```
Elapsed: 00:00:00.00
SQL> select type,status,name,object_no,row_count,row_size_avg
  2  from v$result_cache_objects order by 1;

no rows selected

Elapsed: 00:00:00.00
```

## Manual Solutions for Practice 5-7: Managing Resources

In this practice, you create an `APPUSER` consumer group and assign it to the default `DEFAULT_PLAN` resource plan. Then you map a couple of Oracle users and your major OS user to resource groups. Activate the resource plan and test your assignments.

Log in as the `SYS` user (with `oracle` as the password, connect as `SYSDBA`) and perform the necessary tasks either through Enterprise Manager Database Control or through SQL*Plus. All scripts for this practice are in the `/home/oracle/labs` directory.

If there is no `HR`, `SCOTT`, `OE`, or `PM` user, execute `lab_05_07.sql` to create sample users by using SQL*Plus.

```
$ cd /home/oracle/labs
$ sqlplus / as sysdba @lab_05_07.sql
Connected.

User created.


User created.


User created.

create user hr identified by pm account unlock
            *
ERROR at line 1:
ORA-01920: user name 'HR' conflicts with another user or role name



Grant succeeded.


Grant succeeded.


Grant succeeded.


Grant succeeded.
```

1. Using Enterprise Manager Database Control, create a resource group called `APPUSER`. At this point, **do not** add users to the group.

   **See the GUI solutions.**

2. Add the APPUSER and LOW_GROUP consumer groups to the DEFAULT_PLAN resource plan. Change the level 3 CPU resource allocation percentages: 60% for the APPUSER consumer group and 40% for the LOW_GROUP consumer group.

   **See the GUI solutions.**

3. Configure Consumer Group Mappings, so that the HR Oracle user belongs to the APPUSER consumer group and the SCOTT Oracle user to the LOW_GROUP consumer group. For the SCOTT user, confirm that his ORACLE_USER attribute has a higher priority than the CLIENT_OS_USER attribute.

   **See the GUI solutions.**

4. Configure Consumer Group Mappings so that the oracle OS user belongs to the SYS_GROUP consumer group.

   **See the GUI solutions.**

5. Assign the PM Oracle user to the following consumer groups: APPUSER, LOW_GROUP, and SYS_GROUP.

   **See the GUI solutions.**

6. Activate the DEFAULT_PLAN resource group.

   **See the GUI solutions.**

7. Test the Consumer Group Mappings. Start two SQL*Plus sessions: the first with the system/oracle@orcl connect string and the second with the scott/scott@orcl connect string. Test other mappings as well.

   a) To start a SQL*Plus session with the system/oracle@orcl connect string and to set your SQL prompt to "FIRST", enter:

```
$ sqlplus system@orcl
Enter password: oracle <<< not displayed

SQL> SET SQLPROMPT "FIRST>"
FIRST>
```

   b) To start a SQL*Plus session with the scott/oracle@orcl connect string and to set your SQL prompt to "SECOND", enter:

```
$ sqlplus scott@orcl
Enter password: scott<<< not displayed

SQL> SET SQLPROMPT "SECOND>"
SECOND>
```

c) In your `FIRST` SQL*Plus session, enter:

```
FIRST> SELECT schemaname,resource_consumer_group
  2  FROM v$session
  3  WHERE schemaname not in ('SYS','SYSMAN','DBSNMP');

SCHEMANAME                        RESOURCE_CONSUMER_GROUP
------------------------------ --------------------------------
SYSTEM                           SYS_GROUP
SCOTT                            LOW_GROUP
FIRST>
```

d) ***Question:*** To which consumer group does the `SCOTT` user belong?

***Answer:*** `SCOTT` is in the `LOW_GROUP` consumer group.

**Note:** Your output for this step (and the following steps) may not look exactly like the output shown. The information of concern here is the specific users being mentioned.

e) In the `SECOND` terminal window, connect as the `PM` user with the `pm` password:

```
SECOND>connect pm@orcl
Enter password: pm<<< not displayed

Connected.
SECOND>
```

f) In your `FIRST` SQL*Plus session, enter "/" to execute the previous SQL statement again.

```
FIRST>/

SCHEMANAME                        RESOURCE_CONSUMER_GROUP
------------------------------ --------------------------------
SYSTEM                           SYS_GROUP
PM                               SYS_GROUP
FIRST>
```

g) ***Question:*** To which consumer group does the `PM` user belong?

***Answer:*** `PM` is in the `SYS_GROUP` consumer group.

h) In the `SECOND` terminal window, connect as the `OE` user with `oe` as the password:

```
SECOND>connect oe@orcl
Enter password: oe<<< not displayed

Connected.
SECOND>
```

i) In your `FIRST` SQL*Plus session, enter "/" to execute the previous SQL statement again.

```
FIRST>/

SCHEMANAME                        RESOURCE_CONSUMER_GROUP
------------------------------ --------------------------------
SYSTEM                           SYS_GROUP
OE                              OTHER_GROUPS
FIRST> exit
```

j) Exit both the SQL*Plus sessions.

k) *Question:* When testing your `OE` Oracle user, you notice that `OE` is in the `OTHER_GROUPS` consumer group. Why is that?

   *Possible Answer:* The `OE` user is not explicitly assigned to another consumer resource group.

8. Revert to your original configuration by deactivating the `DEFAULT_PLAN` resource group, undoing all consumer group mappings (use `rsc_cleanup.sh`), and finally by deleting the `APPUSER` resource group.

   **See the GUI solutions.**

## Manual Solutions for Practice 5-8: Using SQL Access Advisor

If there is no `SH` schema on the `orcl` database, run the `SH.sh` script that is located at `/home/oracle/labs`. The script will create the `SH` schema. You can ignore the errors on the Data Pump Import utility.

```
$ cd /home/oracle/labs
$ ./SH.sh
```

1. From a terminal session, connected as the `oracle` user, execute the `sqlaccessadv_setup.sh` script. This script generates the necessary data that you use throughout this practice. In particular, it generates the SQL Tuning Set that is used to represent the workload that you want to analyze.

```
$ ./sqlaccessadv_setup.sh
…
SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> Connected.
SQL> SQL>
Grant succeeded.
SQL> SQL>
User altered.
SQL> SQL> Connected.
SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL>   2    3    4    5    6    7    8
9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24
PL/SQL procedure successfully completed.
SQL> SQL>

…
LAST_ANAL
---------
01-DEC-10
…
01-DEC-10
9 rows selected.
SQL> SQL> SQL> SQL> Disconnected from …
$
```

2. Using Enterprise Manager, create a SQL Access Advisor tuning task based on the captured workload that is in the `SH.SQLSET_MY_ACCESS_WORKLOAD` SQL tuning set by using the `SQLACCESS_WAREHOUSE` template.

   **See the GUI solutions.**

3. After this is done, investigate the proposed recommendations.

   **See the GUI solutions.**

## Manual Solutions for Practice 5-9: Creating Baselines

Using the `orcl` database, create baselines. (This exercise requires that the default snapshot collection has been running since the database was started.)

1. Create a baseline named Monday over past snapshots of Monday and compute statistics over the static baseline.

   **See the GUI solutions.**

2. Create a repeating baseline in the future. Set the following:
   Baseline Name: `afterwork`
   Start Time: 6:00 PM
   Duration: 12 hours
   Frequency: Daily
   Start Time: Set it to the Sunday preceding the class.
   End Time: Set it to the Saturday following the class.
   Retention Time: 28 days

   **See the GUI solutions.**

3. View the data dictionary views.
   a) Connect to the `orcl` database instance as the `SYS` user.

```
$ sqlplus / as sysdba
```

   b) Select the `BASELINE_NAME` and `BASELINE_TYPE` columns from `DBA_HIST_BASELINE`.

```
SQL> select BASELINE_NAME,BASELINE_TYPE from DBA_HIST_BASELINE;

BASELINE_NAME                                                    BASELINE_TYPE
----------------------------------------------------------------
-------------
Monday                                                          STATIC
SYSTEM_MOVING_WINDOW
MOVING_WINDOW

SQL> select TEMPLATE_NAME,TEMPLATE_TYPE
  2  from DBA_HIST_BASELINE_TEMPLATE;

TEMPLATE_NAME                  TEMPLATE_
------------------------------ ---------
afterwork                     REPEATING
```

## Manual Solutions for Practice 5-10: Using SQL Plan Management

**Background:** SQL Plan Management (SPM) is a new Oracle Database 11*g* feature that provides controlled execution plan evolution.

With SPM, the optimizer automatically manages execution plans and ensures that only known or verified plans are used.

When a new plan is found for a SQL statement, it will not be used until it has been verified to have comparable or better performance than the current plan.

1. Before you can start this practice, you need to set up a new user. Execute the `spm_setup.sh` script to set up the environment for this practice. This script creates the `spm` user that you use throughout the practice.

```
$ export ORACLE_SID=orcl
$ ./spm_setup.sh

SQL*Plus: Release 11.2.0.3.0 Production on Mon Mar 11 21:35:26 2013
Copyright (c) 1982, 2011, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
SQL> SQL> SQL> SQL> drop user spm cascade
          *
ERROR at line 1:
ORA-01918: user 'SPM' does not exist
SQL>
User created.
SQL>
Grant succeeded.
SQL>
System altered.
SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
$
```

2. The first component of SPM is Plan Capture. There are two main ways to capture plans: automatically (on the fly), or bulk load. You look at automatic capture first. Connect to the `orcl` database as the `spm` user with `spm` as the password and enable automatic plan capture so that the SPM repository is automatically populated for any repeatable SQL statement.

```
$ sqlplus spm/spm

SQL> show parameter baseline

NAME                                TYPE       VALUE
------------------------------------ -----------
```

```
------------------------------
optimizer_capture_sql_plan_baselines boolean     FALSE
optimizer_use_sql_plan_baselines     boolean     TRUE

SQL> alter session set optimizer_capture_sql_plan_baselines=TRUE;

Session altered.

SQL>
```

3. Execute the following query in your SQL*Plus session (no space in `/*LOAD…`):
   `select /*LOAD_AUTO*/ * from sh.sales where quantity_sold > 40 order by prod_id;`
   Use the `spm_query1.sql` script to execute the query.

```
SQL> @spm_query1
SQL>
SQL> select /*LOAD_AUTO*/ * from sh.sales where quantity_sold > 40 order
by prod_id;

no rows selected

SQL>
```

4. Because this is the first time that you have seen this SQL statement, it is not yet repeatable, so there is no plan baseline for it. To confirm this, you can check that the plan baseline was not loaded. Check whether there are any plan baselines that exist for your statement.

```
SQL> select signature, sql_handle, sql_text, plan_name, origin, enabled,
accepted, fixed, autopurge from dba_sql_plan_baselines where sql_text
like 'select /*LOAD_AUTO*/%';

no rows selected

SQL>
```

5. Re-execute the query in step 3 by using `spm_query1.sql`.

```
SQL> @spm_query1
SQL> set echo on
SQL>
SQL> select /*LOAD_AUTO*/ * from sh.sales where quantity_sold > 40 order
by prod_id;

no rows selected

SQL>
```

6. The SQL statement is now known to be repeatable and a plan baseline is automatically captured. Check that the plan baseline was loaded for the previous statement. What do you observe?

a) You can see from the output that a baseline has been created and enabled for this SQL statement. You can also tell that this plan was captured automatically by checking the values of the **origin** column.

```
SQL> select signature, sql_handle, sql_text, plan_name, origin, enabled,
accepted, fixed, autopurge from dba_sql_plan_baselines where sql_text
like 'select /*LOAD_AUTO*/%';

 SIGNATURE SQL_HANDLE
---------- ------------------------------
SQL_TEXT
----------------------------------------------------------------------
-----------
PLAN_NAME                      ORIGIN         ENA ACC FIX AUT
------------------------------ -------------- --- --- --- ---
8.0622E+18 SQL_6fe28d438dfc352f
select /*LOAD_AUTO*/ * from sh.sales where quantity_sold > 40 order by
prod_id
SQL_PLAN_6zsnd8f6zsd9g54bc8843 AUTO-CAPTURE   YES YES NO  YES
SQL>
```

7. Now, change or alter the optimizer mode to use `FIRST_ROWS` optimization and re-execute your statement. Describe what happened:

   a) This triggers the SQL statement to execute with a different plan:

```
SQL> alter session set optimizer_mode = first_rows;

Session altered.

SQL> @spm_query1
SQL> set echo on
SQL>
SQL> select /*LOAD_AUTO*/ * from sh.sales where quantity_sold > 40 order
by prod_id;

no rows selected

SQL>
```

   b) Because the SQL statement has a new plan, another plan baseline is automatically captured. You can confirm this by checking the plan baseline again:

```
SQL> select signature, sql_handle, sql_text, plan_name, origin, enabled,
accepted, fixed, autopurge from dba_sql_plan_baselines where sql_text
like 'select /*LOAD_AUTO*/%';

 SIGNATURE SQL_HANDLE
---------- ------------------------------
SQL_TEXT
----------------------------------------------------------------------
-----------
PLAN_NAME                      ORIGIN         ENA ACC FIX AUT
```

```
------------------------------ -------------- --- --- --- ---
8.0622E+18 SQL_6fe28d438dfc352f
select /*LOAD_AUTO*/ * from sh.sales where quantity_sold > 40 order by
prod_id
SQL_PLAN_6zsnd8f6zsd9g11df68d0 AUTO-CAPTURE   YES NO  NO  YES

8.0622E+18 SQL_6fe28d438dfc352f
select /*LOAD_AUTO*/ * from sh.sales where quantity_sold > 40 order by
prod_id
SQL_PLAN_6zsnd8f6zsd9g54bc8843 AUTO-CAPTURE   YES YES NO  YES
SQL>
```

c) Now you see two plan baselines for your query, but notice that the second plan has not been accepted. This new plan will have to be validated before it is acceptable as a good plan to use.

8. Now reset the optimizer mode to default values and disable auto capture of plan baselines.

```
SQL> alter session set optimizer_mode = all_rows;
Session altered.

SQL> alter session set optimizer_capture_sql_plan_baselines = FALSE;

Session altered.

SQL>
```

9. Purge the plan baselines and confirm that the SQL plan baseline is empty. Use `purge_auto_baseline.sql`:

```
SQL> @purge_auto_baseline
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> exec :cnt :=
dbms_spm.drop_sql_plan_baseline('SQL_6fe28d438dfc352f');

PL/SQL procedure successfully completed.

SQL>
SQL> select signature, sql_handle, sql_text, plan_name, origin, enabled,
accepted, fixed, autopurge
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select /*LOAD_AUTO*/%';
no rows selected

SQL>
```

10. Now, you will see how to directly load plan baselines from the cursor cache. Before you begin, you need some SQL statements. Still connected to your SQL*Plus session, check the

execution plan for the following SQL statement, and then execute it (use the `explain_spm_query3.sql` and `spm_query3.sql` scripts):
`select /*LOAD_CC*/ * from sh.sales where quantity_sold > 40 order by prod_id;`

```
SQL> @explain_spm_query3
SQL> set echo on
SQL>
SQL> explain plan for
  2  select /*LOAD_CC*/ * from sh.sales where quantity_sold > 40 order by
prod_id;
Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic'));

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------
-----------
Plan hash value: 3803407550
--------------------------------------
| Id  | Operation           | Name  |
--------------------------------------
|   0 | SELECT STATEMENT    |       |
|   1 |  SORT ORDER BY      |       |
|   2 |   PARTITION RANGE ALL|      |
|   3 |    TABLE ACCESS FULL | SALES |
--------------------------------------
10 rows selected.

SQL>
SQL> @spm_query3
SQL> set echo on
SQL>
SQL> select /*LOAD_CC*/ * from sh.sales where quantity_sold > 40 order
by prod_id;

no rows selected

SQL>
```

11. Now change the optimizer mode to use `FIRST_ROWS` optimization and re-execute step 10. What do you observe?

```
SQL> alter session set optimizer_mode = first_rows;

Session altered.

SQL>
SQL> @explain_spm_query3
SQL> set echo on
SQL>
SQL> explain plan for
  2  select /*LOAD_CC*/ * from sh.sales where quantity_sold > 40 order by
prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic'));

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------
-----------
Plan hash value: 899219946

---------------------------------------------------------------
| Id  | Operation                          | Name           |
---------------------------------------------------------------
|   0 | SELECT STATEMENT                    |               |
|   1 |  SORT ORDER BY                     |               |
|   2 |   PARTITION RANGE ALL               |               |
|   3 |    TABLE ACCESS BY LOCAL INDEX ROWID| SALES          |
|   4 |     BITMAP CONVERSION TO ROWIDS     |               |
|   5 |      BITMAP INDEX FULL SCAN        | SALES_PROMO_BIX |

PLAN_TABLE_OUTPUT
----------------------------------------------------------------------
-----------
---------------------------------------------------------------

12 rows selected.

SQL> @spm_query3
SQL> set echo on
SQL>
SQL> select /*LOAD_CC*/ * from sh.sales where quantity_sold > 40 order
by prod_id;

no rows selected
```

12. Reset the optimizer mode to `ALL_ROWS`:

```
SQL> alter session set optimizer_mode = all_rows;

Session altered.

SQL>
```

13. Now that the cursor cache is populated, you need to get the SQL ID for your SQL statement by using the following SQL.Use the SQL ID to filter the content of the cursor cache and load the baselines with these two plans.

Use the `load_cc_baseline.sql` script.

```
SQL> @load_cc_baseline
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> variable sqlid varchar2(20);
SQL>
SQL> begin
  2   select distinct sql_id into :sqlid from v$sql where sql_text like
'select /*LOAD_CC*/%';
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL>
SQL> print sqlid;

SQLID
----------------------------------------------------------------------
-----------
9qx77nb12bh6z

SQL>
SQL> execute :cnt := dbms_spm.load_plans_from_cursor_cache(sql_id =>
:sqlid);

PL/SQL procedure successfully completed.

SQL>
```

14. Confirm that the baselines were loaded.

```
SQL> select signature, sql_handle, sql_text, plan_name, origin, enabled,
accepted, fixed, autopurge from dba_sql_plan_baselines where sql_text
like 'select /*LOAD_CC*/%';

 SIGNATURE SQL_HANDLE
---------- ------------------------------
SQL_TEXT
----------------------------------------------------------------------
-----------
PLAN_NAME                      ORIGIN        ENA ACC FIX AUT
------------------------------ -------------- --- --- --- ---
1.7783E+19 SQL_f6cb7f742ef93547
select /*LOAD_CC*/ * from sh.sales where quantity_sold > 40 order by prod_id
SQL_PLAN_gdkvzfhrgkda711df68d0 MANUAL-LOAD    YES YES NO  YES

1.7783E+19 SQL_f6cb7f742ef93547
select /*LOAD_CC*/ * from sh.sales where quantity_sold > 40 order by prod_id
```

```
SQL_PLAN_gdkvzfhrgkda754bc8843 MANUAL-LOAD   YES YES NO  YES
SQL>
```

15. Purge the plan baselines by using `purge_cc_baseline.sql`.

```
SQL> @purge_cc_baseline
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> exec :cnt :=
dbms_spm.drop_sql_plan_baseline('SQL_f6cb7f742ef93547');

PL/SQL procedure successfully completed.

SQL>
SQL> print cnt;

       CNT
----------
         2

SQL>
SQL> REM Check that plan baselines were purged:
SQL>
SQL> select signature, sql_handle, sql_text, plan_name, origin, enabled,
accepted, fixed, autopurge
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select /*LOAD_CC*/%';

no rows selected

SQL>
```

16. Drop the `spm` user to clean up.

```
SQL> conn / as sysdba
Connected.
SQL> drop user spm cascade;

User dropped.

SQL>
```

## Manual Solutions for Practice 5-11: Using the SQL Performance Analyzer

In this practice, you simulate exporting a SQL Tuning Set (STS) from a 10*g* database and import it back into an 11*g* test environment. There, you access the performance of the SQL statements that you imported before upgrading the 10*g* database.

If you do not use an `spfile`, at first, create an `spfile` from `pfile` and restart the `orcl` database.

```
SQL> conn / as sysdba
SQL> create spfile from pfile;
File created.

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

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
Database opened.
SQL> show parameter spfile

NAME                                 TYPE       VALUE
------------------------------------ -----------
------------------------------
spfile                               string
/u01/app/oracle/product/11.2.0
                                                         /dbhome_1/dbs/spfileorcl.ora
SQL>
```

1. From a terminal window, which is referred to as the first session, execute the `setup_SPAbig10g.sh` script to set up your simulated 10*g* environment. In this simulated environment, the `OPTIMIZER_FEATURES_ENABLE` parameter is set to `10.2.0.2`.

```
$ cd /home/oracle/labs
$ ./setup_SPAbig10g.sh
rm: cannot remove `/u01/app/oracle/admin/orcl/dpdump/apps.dmp': No such
file or directory
rm: cannot remove `/u01/app/oracle/admin/orcl/dpdump/appsandstage.dmp':
No such file or directory

SQL*Plus: Release 11.2.0.3.0 Production on Mon Mar 11 21:35:26 2013
```

```
Copyright (c) 1982, 2011, Oracle.  All rights reserved.
SQL> Connected.
SQL> SQL> SQL> SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> SQL> ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             394266656 bytes
Database Buffers          432013312 bytes
Redo Buffers                6606848 bytes
Database mounted.
Database opened.
SQL> SQL> SQL> SQL> BEGIN dbms_sqltune.drop_sqlset('STS_JFV','SYS'); END;

*
ERROR at line 1:
ORA-13754: "SQL Tuning Set" "STS_JFV" does not exist for user "SYS".
ORA-06512: at "SYS.DBMS_SQLTUNE_INTERNAL", line 13171
ORA-06512: at "SYS.DBMS_SQLTUNE", line 4409
ORA-06512: at line 1

SQL> SQL> drop user apps cascade
        *
ERROR at line 1:
ORA-01918: user 'APPS' does not exist

SQL>
Directory created.

SQL> SQL> SQL>
Import: Release 11.2.0.3.0 - Production on Mon Mar 11 21:36:54 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights
reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.3.0
- Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01":  system/********
directory=DATA_DIR dumpfile=apps.dmp
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
```

```
. . imported "APPS"."FACT_PD_OUT_ITM_293"            175.0 MB 1501663
rows

. . imported "APPS"."ADM_PG_FEATUREVALUE"             14.26 MB  172392
rows
. . imported "APPS"."LU_ELEMENTGROUP_REL"             3.478 MB   84468
rows
. . imported "APPS"."LU_OUTLET_293"                  3.404 MB   22289
rows
. . imported "APPS"."LU_ITEM_293"                   1.280 MB    5355
rows
. . imported "APPS"."LU_ELEMENTRANGE_REL"             768.7 KB    7957
rows
. . imported "APPS"."LU_ELEMENTRANGE_REL_Q2"          768.7 KB    7957
rows
. . imported "APPS"."ADM_CC_FEATUREVALUE"             250.2 KB    3047
rows
. . imported "APPS"."LU_PG_FEATUREVALUE_15_Q2"        123.5 KB    2505
rows
. . imported "APPS"."LU_PERIOD_293"                  8.593 KB     31
rows
. . imported "APPS"."PLAN_TABLE"                       0 KB      0 rows
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/FUNCTIONAL_AND_BITMAP/INDEX
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/FUNCTIONAL_AND_BITMAP/INDEX_STAT
ISTICS
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Job "SYSTEM"."SYS_IMPORT_FULL_01" successfully completed at 18:54:56


SQL> SQL>
LAST_ANAL
---------

04-JUL-07
04-JUL-07
04-JUL-07
04-JUL-07
04-JUL-07
04-JUL-07
04-JUL-07

8 rows selected.

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

```
SQL> SQL>   2
Database altered.
SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
$
```

2. **(Perform steps 2 and 3 at the same time)** Generate a SQL Tuning Set (STS) called `STS_JFV` that captures SQL statements from the cursor cache for approximately 12 minutes every five seconds. Make sure that you try to capture only statements from the `SQL_JFV` module in the `APPS` schema. Also, this STS should belong to the `SYS` user. Use the `capsts10g.sh` script to perform this step.

```
$./capsts10g.sh

SQL*Plus: Release 11.2.0.3.0 Production on Mon Mar 11 21:43:18 2013

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

SQL> Connected.
SQL> SQL> SQL> SQL> SQL>   2    3    4    5    6    7    8    9   10   11   12
```

3. (**Perform steps 2 and 3 at the same time**) From a second terminal window, connected as the `oracle` user, execute your workload by using the `wrkl10g_jfv.sh` script. This script runs a workload of 45 statements that will be captured in `STS_JFV` automatically.

```
$ ./wrkl10g_jfv.sh

SQL*Plus: Release 11.2.0.3.0 Production on Mon Mar 11 21:44:20 2013

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

SQL> Connected.
SQL> SQL> SQL>
System altered.

SQL>
System altered.

SQL> SQL> SQL> Connected.
SQL> SQL> SQL> SQL>
Session altered.

SQL> SQL> SQL> SQL> SQL> SQL> @Start 2010:NOV-29:18:56:31

SQL> SQL> SQL> SQL> SQL> SQL> SQL>
System altered.

SQL> SQL> SQL> @Statement 1
SQL>
```

```
PL/SQL procedure successfully completed.

SQL> SQL> SQL>   2    3    4    5    6    7    8    9   10   11   12   13   14
Elapsed: 00:00:00.02

Statistics
----------------------------------------------------------
        908  recursive calls
          1  db block gets
        273  consistent gets
        141  physical reads
        240  redo size
        542  bytes sent via SQL*Net to client
        524  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         13  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL> @1 Results in 15 Ticks
SQL> SQL> SQL> SQL> @Statement 2
```

4. After approximately 12 minutes, both sessions should have finished. Connect to the `orcl` database as the `sys` user, check the content of `STS_JFV`, *and* stage it in a table called `APPS.STS_JFV_TAB`.

```
$ sqlplus / as sysdba

SQL> select name,statement_count from dba_sqlset;

NAME                           STATEMENT_COUNT
------------------------------ ---------------
SQLSET_MY_SQLACCESS_WORKLOAD                 5
STS_JFV                                     40

SQL> exec DBMS_SQLTUNE.CREATE_STGTAB_SQLSET('STS_JFV_TAB','APPS');


PL/SQL procedure successfully completed.

SQL> exec
DBMS_SQLTUNE.PACK_STGTAB_SQLSET('STS_JFV','SYS','STS_JFV_TAB','APPS')
;


PL/SQL procedure successfully completed.
```

5. Using Data Pump Export, export the `APPS` schema to the default Data Pump directory (`DATA_PUMP_DIR`).

```
$ expdp system/oracle directory=DATA_PUMP_DIR dumpfile=appsandstage
schemas=APPS

Export: Release 11.2.0.3.0 - Production on Mon Mar 11 21:46:24 2013
Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
Starting "SYSTEM"."SYS_EXPORT_SCHEMA_01":  system/********
directory=DATA_PUMP_DIR dumpfile=appsandstage schemas=APPS
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 238.4 MB
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/FUNCTIONAL_AND_BITMAP/INDEX
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/FUNCTIONAL_AND_BITMAP/INDEX_STATIST
ICS
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
. . exported "APPS"."FACT_PD_OUT_ITM_293"            175.0 MB 1501663 rows
. . exported "APPS"."ADM_PG_FEATUREVALUE"            14.26 MB  172392 rows
. . exported "APPS"."LU_ELEMENTGROUP_REL"            3.478 MB   84468 rows
. . exported "APPS"."LU_OUTLET_293"                  3.404 MB   22289 rows
. . exported "APPS"."LU_ITEM_293"                    1.280 MB    5355 rows
. . exported "APPS"."STS_JFV_TAB"                    456.6 KB     750 rows
. . exported "APPS"."LU_ELEMENTRANGE_REL"            768.7 KB    7957 rows
. . exported "APPS"."LU_ELEMENTRANGE_REL_Q2"         768.7 KB    7957 rows
. . exported "APPS"."ADM_CC_FEATUREVALUE"            250.2 KB    3047 rows
. . exported "APPS"."LU_PG_FEATUREVALUE_15_Q2"       123.5 KB    2505 rows
. . exported "APPS"."LU_PERIOD_293"                  8.593 KB      31 rows
. . exported "APPS"."PLAN_TABLE"                         0 KB       0 rows
Master table "SYSTEM"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
************************************************************************
******
Dump file set for SYSTEM.SYS_EXPORT_SCHEMA_01 is:
  /u01/app/oracle/product/11.2.0/dbhome_1/rdbms/log/ appsandstage.dmp
Job "SYSTEM"."SYS_EXPORT_SCHEMA_01" successfully completed at 19:10:45
```

6. `setup_SPAbig11g.sh` script to perform this step.

```
$./setup_SPAbig11g.sh

SQL*Plus: Release 11.2.0.3.0 Production on Mon Mar 11 21:47:24 2013

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

SQL> SQL> Connected.
SQL> SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             339740704 bytes
Database Buffers          486539264 bytes
Redo Buffers                6606848 bytes
Database mounted.
Database opened.
SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release
11.2.0.3.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
$
```

7. Drop the `apps` user from the `orcl` database. Using Data Pump Import, import the `APPS` schema into your system.

```
$ sqlplus / as sysdba
SQL> drop user apps cascade;

User dropped.

SQL> exit
$ impdp system/oracle directory=DATA_PUMP_DIR dumpfile=appsandstage

Import: Release 11.2.0.3.0 - Production on Mon Mar 11 21:47:22 2013

Copyright (c) 1982, 2011, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01":  system/********
directory=DATA_PUMP_DIR dumpfile=appsandstage
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
. . imported "APPS"."FACT_PD_OUT_ITM_293"            175.0 MB 1501663 rows
. . imported "APPS"."ADM_PG_FEATUREVALUE"            14.26 MB  172392 rows
. . imported "APPS"."LU_ELEMENTGROUP_REL"            3.478 MB   84468 rows
. . imported "APPS"."LU_OUTLET_293"                  3.404 MB   22289 rows
. . imported "APPS"."LU_ITEM_293"                    1.280 MB    5355 rows
. . imported "APPS"."STS_JFV_TAB"                    456.6 KB     750 rows
. . imported "APPS"."LU_ELEMENTRANGE_REL"            768.7 KB    7957 rows
. . imported "APPS"."LU_ELEMENTRANGE_REL_Q2"         768.7 KB    7957 rows
. . imported "APPS"."ADM_CC_FEATUREVALUE"            250.2 KB    3047 rows
. . imported "APPS"."LU_PG_FEATUREVALUE_15_Q2"       123.5 KB    2505 rows
. . imported "APPS"."LU_PERIOD_293"                  8.593 KB      31 rows
. . imported "APPS"."PLAN_TABLE"                         0 KB       0 rows
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/FUNCTIONAL_AND_BITMAP/INDEX
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/FUNCTIONAL_AND_BITMAP/INDEX_STATIST
ICS
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Job "SYSTEM"."SYS_IMPORT_FULL_01" successfully completed at 19:21:06
```

8. Unpack the previously imported staging table in the `SYS` schema.

```
$ sqlplus / as sysdba

SQL> exec
DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET('STS_JFV','SYS',TRUE,'STS_JFV_TAB',
'APPS');

PL/SQL procedure successfully completed.
```

9. As the `SYS` user, use Enterprise Manager Database Control to test the behavior of `STS_JFV` in the simulated 10*g* environment and compare it to the 11*g* environment. You will do this by changing the `OPTIMIZER_FEATURES_ENABLE` parameter. What are your conclusions?

   **See the GUI solutions.**

## Manual Solutions for Practice 7-1: Adding a Physical Standby Database to your Configuration

1. Add a new physical standby database to your configuration.

   **Primary Database**
   Database Name: `orcl`
   Instance Name: `orcl`
   Database Unique Name: `orcl`
   Grid Control Target Name: `orcl.oracle.com`

   **Physical Standby Database**
   Database Name: `orcl`
   Instance Name: `stdby`
   Database Unique Name: `stdby`
   Type of standby: Physical
   Grid Control Target Name: `stdby.oracle.com`

   **Settings:**
   Host: Odd PC
   File Location: `/u01/app/oracle/oradata/stdby`

**Answer:**

a) First, create a text initialization parameter file (`initorcl.ora`).

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba
SQL> create pfile from spfile;

File created.

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> !rm $ORACLE_HOME/dbs/spfileorcl.ora
SQL> startup
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
Database opened.
```

b) Confirm whether the database is in ARCHIVE mode. If you find that the database is in NOARCHIVE mode, change to ARCHIVE mode.

```
SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     14
Next log sequence to archive   16
Current log sequence           16
```

c) Add the following initialization parameters to initorcl.ora.

```
#for PRIMARY
DB_UNIQUE_NAME = orcl
SERVICE_NAMES = orcl.oracle.com
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(orcl,stdby)'
LOG_ARCHIVE_DEST_1 =
 'LOCATION=/u01/app/oracle/admin/orcl/arch
  VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=orcl'
LOG_ARCHIVE_DEST_2 =
 'SERVICE=stdby LGWR ASYNC
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=stdby REOPEN=30'
LOG_ARCHIVE_DEST_STATE_1 = ENABLE
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE
LOG_ARCHIVE_FORMAT = %t_%s_%r.arc

# for STANDBY
FAL_SERVER = stdby
FAL_CLIENT = orcl
DB_FILE_NAME_CONVERT =
  '/u01/app/oracle/oradata/stdby',
  '/u01/app/oracle/oradata/orcl'
LOG_FILE_NAME_CONVERT =
  '/u01/app/oracle/oradata/stdby',
  '/u01/app/oracle/oradata/orcl'
STANDBY_FILE_MANAGEMENT = AUTO
```

d) Create an initialization parameter file for the standby database. The parameter file name is initstdby.ora. You can copy and modify the initialization parameter file from the primary database. Use the sga_target and pga_aggregate_target parameters instead of memory_target because your machine does not support memory_target for stdby.

```
db_name=orcl
control_files = (/u01/app/oracle/oradata/stdby/control01.ctl,
                 /u01/app/oracle/oradata/stdby/control02.ctl)
sga_target=550M
pga_aggregate_target=150M
```

```
processes = 150
db_block_size=8192
db_domain=oracle.com
db_recovery_file_dest='/u01/app/oracle/fast_recovery_area'
db_recovery_file_dest_size=10G
diagnostic_dest='/u01/app/oracle'
open_cursors=300
remote_login_passwordfile='EXCLUSIVE'
undo_tablespace='UNDOTBS1'
compatible ='11.2.0'

#for PRIMARY
DB_UNIQUE_NAME = stdby
SERVICE_NAMES = stdby.oracle.com
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(stdby,orcl)'
LOG_ARCHIVE_DEST_1 =
 'LOCATION=/u01/app/oracle/admin/stdby/arch
  VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=stdby'
LOG_ARCHIVE_DEST_2 =
 'SERVICE=orcl LGWR ASYNC
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=orcl REOPEN=30'
LOG_ARCHIVE_DEST_STATE_1 = ENABLE
LOG_ARCHIVE_DEST_STATE_2 = ENABLE
REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE
LOG_ARCHIVE_FORMAT = %t_%s_%r.arc
# for STANDBY
FAL_SERVER = orcl
FAL_CLIENT = stdby
DB_FILE_NAME_CONVERT =
  '/u01/app/oracle/oradata/orcl',
  '/u01/app/oracle/oradata/stdby'
LOG_FILE_NAME_CONVERT =
  '/u01/app/oracle/oradata/orcl',
  '/u01/app/oracle/oradata/stdby'
STANDBY_FILE_MANAGEMENT = AUTO
```

e) Create an archive log files destination `/u01/app/oracle/admin/orcl/arch`. Create an additional archive log destination for the `stdby` database.

```
$ mkdir -p /u01/app/oracle/admin/orcl/arch
$ mkdir -p /u01/app/oracle/admin/stdby/arch
```

f) Shut down the `orcl` database.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

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

```
SQL>exit
$
```

g) Back up and copy the data files of the `orcl` database to the **`/u01/app/oracle/oradata/stdby`** directory.

```
$ cd $ORACLE_BASE/oradata
$ mkdir stdby
$ cp orcl/*.dbf stdby/
$
```

h) Create a standby control file and save it in the **`/u01/app/oracle/oradata/stdby`** directory.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba
SQL> startup
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
Database opened.

SQL> alter database create standby controlfile as
  2    '/u01/app/oracle/oradata/stdby/control01.ctl';

Database altered.

SQL> !cp /u01/app/oracle/oradata/stdby/control01.ctl
/u01/app/oracle/oradata/stdby/control02.ctl

SQL>
```

i) Copy the password file from the primary database.

```
$ cd $ORACLE_HOME/dbs
$ cp orapworcl orapwstdby
```

j) Configure Oracle Net. Add the following for the `tnsnames.ora` file located in `$ORACLE_HOME/network/admin`.

```
STDBY =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS =
        (PROTOCOL = TCP)(HOST = <your database server hostname)(PORT
= 1521))
    )
    (CONNECT_DATA =
      (SERVER = dedicated)
      (SERVICE_NAME = stdby.oracle.com)
    )
  )
```

k) Start up the physical standby database.

```
$ export ORACLE_SID=stdby
$ sqlplus / as sysdba
SQL> startup mount
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL>
```

l) Start the Redo Apply for the standby database.

```
SQL> recover managed standby database disconnect;
Media recovery complete.
```

m) Archive the redo log file at the primary database.

```
SQL> alter system switch logfile;
```

n) After the physical standby database creation is complete, use Verify to validate the configuration and add standby redo logs.

```
(Primary database)
SQL>  alter database add standby logfile
  2  '/u01/app/oracle/oradata/orcl/stdby01.log' size 100M;

Database altered.

SQL>  alter database add standby logfile
  2  '/u01/app/oracle/oradata/orcl/stdby02.log' size 100M;

Database altered.

SQL> alter database add standby logfile
  2  '/u01/app/oracle/oradata/orcl/stdby03.log' size 100M;

Database altered.

(Standby database)
SQL> recover managed standby database cancel;
Media recovery complete.
SQL> alter database add standby logfile
  2   '/u01/app/oracle/oradata/stdby/stdby01.log' size 100M;

Database altered.
```

```
SQL> alter database add standby logfile
  2   '/u01/app/oracle/oradata/stdby/stdby02.log' size 100M;

Database altered.

SQL> alter database add standby logfile
  2   '/u01/app/oracle/oradata/stdby/stdby03.log' size 100M;

Database altered.

SQL> recover managed standby database disconnect;
Media recovery complete.
SQL>
```

2. Configure Data Guard Broker and configure an additional listener.

   a) You will configure Data Guard Broker in Practice 7-7: Enabling Fast-Start Failover. You can skip this step for now.

## Manual Solutions for Practice 7-2: Using Real-Time Query

1. Enable Real-Time Query and insert a new row in the `hr.regions` table of the primary database. Then confirm that the inserted new row can be queried against the physical standby database while Redo Apply is active.

   a) Enable Real-Time Query in the standby database.

```
$ export ORACLE_SID=stdby
SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;

SQL> ALTER DATABASE OPEN;

Database altered.

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT
LOGFILE DISCONNECT;

Database altered.

SQL> SELECT open_mode FROM V$DATABASE;

OPEN_MODE
--------------------
READ ONLY WITH APPLY
SQL>
```

   b) In your standby database window, invoke SQL*Plus and connect as **SYSDBA**. Query the **hr.regions** table.

```
$ export ORACLE_SID=stdby
$ sqlplus / as sysdba
SQL> select * from hr.regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
SQL>
```

   c) In your primary database window, invoke SQL*Plus and connect as **SYSDBA**. Insert a row in the `hr.regions` table.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba
SQL> select * from hr.regions;

 REGION_ID REGION_NAME
```

```
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
SQL> insert into hr.regions values(5,'New');

1 row created.

SQL> commit;

Commit complete.
SQL> select * from hr.regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
         5 New
SQL>
```

d) Return to your standby database window and query the `hr.regions` table again. In your standby database, you should see the new row that you inserted in the `hr.regions` table in the primary database.

```
SQL> select * from hr.regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
         5 New
SQL>
```

e) Return your physical standby database to `MOUNT` mode. Exit SQL*Plus.

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

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             511707168 bytes
Database Buffers          314572800 bytes
```

```
Redo Buffers                 6606848 bytes
Database mounted.
SQL> recover managed standby database disconnect;
Media recovery complete.
SQL> exit
$
```

## Manual Solutions for Practice 7-3: Performing Switchover

1. Perform the switchover to the standby database.

   a) Query the status of the `orcl` database for the switchover and perform the switchover from the primary database to the physical standby database.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

SQL> select SWITCHOVER_STATUS from v$database;

SWITCHOVER_STATUS
--------------------
TO STANDBY

SQL> alter database commit to switchover to physical standby with
session shutdown wait;

Database altered.

SQL> shutdown abort
ORACLE instance shut down
SQL> startup mount
ORACLE instance started.

Total System Global Area  523108352 bytes
Fixed Size                  1337632 bytes
Variable Size             314574560 bytes
Database Buffers          201326592 bytes
Redo Buffers                5869568 bytes
Database mounted.

SQL> recover managed standby database disconnect;
Media recovery complete.
SQL> select DATABASE_ROLE from v$database;

DATABASE_ROLE
----------------
PHYSICAL STANDBY
SQL>
SQL> select SWITCHOVER_STATUS from v$database;

SWITCHOVER_STATUS
--------------------
TO PRIMARY
```

   b) Query the status of the `stdby` database for the switchover and perform the switchover from the physical standby database to the primary database.

```
$ export ORACLE_SID=stdby
$ sqlplus / as sysdba

SQL> select SWITCHOVER_STATUS from v$database;

SWITCHOVER_STATUS
--------------------
TO PRIMARY

SQL> alter database commit to switchover to primary with session
shutdown wait;

Database altered.

SQL> alter database open;

Database altered.

SQL> select SWITCHOVER_STATUS from v$database;

SWITCHOVER_STATUS
--------------------
TO STANDBY

SQL> select DATABASE_ROLE from v$database;

DATABASE_ROLE
----------------
PRIMARY
SQL>
```

2. Switch back to `orcl` (the original primary database) and confirm that `orcl` is the primary database.

   a) Query the status of the `stdby` database for the switchover and perform the switchover from the primary database to the physical standby database.

```
$ export ORACLE_SID=sydby
$ sqlplus / as sysdba

SQL> select SWITCHOVER_STATUS from v$database;

SWITCHOVER_STATUS
--------------------
TO STANDBY

SQL> alter database commit to switchover to physical standby with
session shutdown wait;

Database altered.
```

```
SQL> shutdown abort
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  523108352 bytes
Fixed Size                  1337632 bytes
Variable Size             314574560 bytes
Database Buffers          201326592 bytes
Redo Buffers                5869568 bytes
Database mounted.

SQL> select DATABASE_ROLE from v$database;

DATABASE_ROLE
----------------
PHYSICAL STANDBY

SQL> recover managed standby database disconnect;
Media recovery complete.
SQL>
```

b) Query the status of the `orcl` database for the switchover and perform the switchover from the physical standby database to the primary database.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

SQL> select SWITCHOVER_STATUS from v$database;

SWITCHOVER_STATUS
--------------------
TO PRIMARY

SQL> alter database commit to switchover to primary with session
shutdown wait;

Database altered.

SQL> alter database open;
Database altered.

SQL> select DATABASE_ROLE from v$database;

DATABASE_ROLE
----------------
PRIMARY

SQL>
```

## Manual Solutions for Practice 7-4: Creating and Managing a Snapshot Standby Database

1. Configure the Flashback Database on your primary and physical standby databases.

   a) Determine if Flashback Database is enabled for the `orcl` database.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

SQL> select flashback_on from v$database;

FLASHBACK_ON
------------------
YES
```

   b) If Flashback Database is not enabled, enable it by using "`alter database flashback on`".

   c) Determine if Flashback Database is enabled for the `stdby` database.

```
$ export ORACLE_SID=stdby
$ sqlplus / as sysdba

SQL> select flashback_on from v$database;

FLASHBACK_ON
------------------
NO
SQL> recover managed standby database cancel;
Media recovery complete.

SQL> alter database flashback on;

Database altered.
SQL> recover managed standby database disconnect;
Media recovery complete.

SQL> select flashback_on from v$database;

FLASHBACK_ON
------------------
YES
```

2. Convert your physical standby database to a snapshot standby database.

   a) If Redo Apply is active, disable Redo Apply (the MRP process).

```
SQL> recover managed standby database cancel;
```

   b) Convert the physical standby database to a snapshot standby database.

```
SQL> alter database convert to SNAPSHOT STANDBY;

Database altered.
```

c) Open your standby database. Confirm whether the database is open in read write mode.

```
SQL> alter database open;

Database altered.

SQL> select open_mode from v$database;

OPEN_MODE
--------------------
READ WRITE
```

3. Update your database.

**Primary:** `insert into hr.regions values(10,’Australia’);`
`commit;`

**Standby:** `insert into hr.regions values(20,’Australia’);`
`commit;`

Query the **`hr.regions`** table in your snapshot standby database.

a) In the primary database:

```
SQL> select * from hr.regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
         5 New

SQL> insert into hr.regions values(10,'Australia');

1 row created.

SQL> commit;

Commit complete.

SQL> select * from hr.regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
         5 New
        10 Australia

6 rows selected.

SQL>
```

b) In the standby database:

```
SQL> insert into hr.regions values(20,'Australia');

1 row created.

SQL> commit;

Commit complete.

SQL> select * from hr.regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
         5 New
        20 Australia

6 rows selected.
```

4. Convert your snapshot standby database back to a physical standby database.

   a) Return your physical standby database to **MOUNT** mode.

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

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL>
```

b) Convert the snapshot standby database to a physical standby database.

```
SQL> alter database convert to PHYSICAL STANDBY;
Database altered.

SQL> shutdown immediate
ORA-01507: database not mounted


ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL> recover managed standby database disconnect;
Media recovery complete.
SQL>
```

5. Verify the shipment of redo data.

a) Disable Redo Apply (the MRP process).

```
SQL> recover managed standby database cancel;
Media recovery complete.
SQL>
```

b) Open your standby database and query `hr.regions`.

```
SQL> select open_mode from v$database;

OPEN_MODE
--------------------
MOUNTED

SQL> alter database open;

Database altered.

SQL> select open_mode from v$database;

OPEN_MODE
--------------------
READ ONLY

SQL>
SQL> select * from hr.regions;
```

```
 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
         5 New
        10 Australia

6 rows selected.
```

c) Return your physical standby database to **MOUNT** mode.

```
$ export ORACLE_SID=stdby
$ sqlplus / as sysdba
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             511707168 bytes
Database Buffers          314572800 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL> recover managed standby database disconnect;
Media recovery complete.
SQL>
SQL> exit
$
```

## Manual Solutions for Practice 7-5: Configuring RMAN Parameters

1. In your RMAN session (connected to your primary database), configure the backup retention policy to allow recovery for seven days.

```
$ export ORACLE_SID=orcl
$ rman target /
RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW of 7days;

new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
new RMAN configuration parameters are successfully stored

RMAN>
```

2. Specify that the archived redo log files can be deleted after they are applied to the standby database.

```
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY to applied on all standby;

new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;
new RMAN configuration parameters are successfully stored

RMAN> exit
```

## Manual Solutions for Practice 7-6: Setting the Data Protection Mode

1. Set the protection mode to maximum availability.

   a) Modify the `log_archive_dest_2` parameter of the `orcl` database. Add the attributes `LGWR`, `SYNC`, and `AFFIRM`.

```
LOG_ARCHIVE_DEST_2 =
 'SERVICE=stdby LGWR SYNC AFFIRM
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=stdby';
```

   b) Change the data protection mode to maximum availability.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             494929952 bytes
Database Buffers          331350016 bytes
Redo Buffers                6606848 bytes
Database mounted.

SQL> alter database set standby database to maximize availability;

Database altered.

SQL> alter database open;

Database altered.

SQL>
```

   c) Confirm the protection mode.

```
SQL> select PROTECTION_MODE from V$DATABASE;

PROTECTION_MODE
--------------------
MAXIMUM AVAILABILITY
```

   d) Modify the `log_archive_dest_2` parameter of the `stdby` database. Add the attributes `LGWR`, `SYNC`, and `AFFIRM`.

```
LOG_ARCHIVE_DEST_2 =
 'SERVICE=orcl LGWR SYNC AFFIRM
  VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=orcl';
```

e) Change the data protection mode to maximum availability.

```
$ export ORACLE_SID=stdby
$ sqlplus / as sysdba

SQL> shutdown immediate
ORA-01109: database not open

Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             494929952 bytes
Database Buffers          331350016 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL> alter database set standby database to maximize availability;

Database altered.

SQL> recover managed standby database disconnect;
Media recovery complete.
```

f) Confirm the protection mode.

```
SQL> select PROTECTION_MODE from V$DATABASE;

PROTECTION_MODE
--------------------
MAXIMUM AVAILABILITY
```

## Manual Solutions for Practice 7-7: Enabling Fast-Start Failover

1. Set fast-start failover properties and enable fast-start failover.

   a) If you use pfile, create spfile to configure DGMGRL.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba
SQL> create spfile from pfile;
File created.

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

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
Database opened.
```

```
$ export ORACLE_SID=stdby
$ sqlplus / as sysdba
SQL> create spfile from pfile;
File created.

SQL> shutdown immediate
ORA-01109: database not open

Database dismounted.
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2217952 bytes
Variable Size             490735648 bytes
Database Buffers          335544320 bytes
Redo Buffers                6606848 bytes
Database mounted.
SQL> recover managed standby database disconnect;
Media recovery complete.
SQL>
```

b) Configure a database service for DGMGRL. Modify `listener.ora` as follows:

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = stdby_DGMGRL.oracle.com)
      (ORACLE_HOME = /u01/app/oracle/product/11.2.0/dbhome_1)
      (SID_NAME = stdby)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = orcl_DGMGRL.oracle.com)
      (ORACLE_HOME = /u01/app/oracle/product/11.2.0/dbhome_1)
      (SID_NAME = orcl)
    )
  )
```

c) Stop and start the listener.

```
$ lsnrctl stop

$ lsnrctl start

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=edaor80p1.us.oracle.com)(PO
RT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.3.0 -
Production
Start Date                29-MAR-2013 12:02:13
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/listener.ora
Listener Log File
/u01/app/oracle/diag/tnslsnr/edaor80p1/listener/alert/log.xml
Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=edaor80p1.us.oracle.com)(PO
RT=1521)))
Services Summary...
Service "orcl_DGMGRL.oracle.com" has 1 instance(s).
  Instance "orcl", status UNKNOWN, has 1 handler(s) for this
service...
Service "stdby_DGMGRL.oracle.com" has 1 instance(s).
  Instance "stdby", status UNKNOWN, has 1 handler(s) for this
service...
The command completed successfully
$
```

d) Set the `DG_BROKER_START` parameter to `TRUE` for both databases.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba
SQL>  select open_mode from v$database;

OPEN_MODE
--------------------
READ WRITE

SQL> show parameter dg_broker_start

NAME                           TYPE        VALUE
------------------------------ ----------- ------------------------------
dg_broker_start                boolean     FALSE

SQL> alter system set dg_broker_start=TRUE;

System altered.

SQL> show parameter dg_broker_start

NAME                            TYPE        VALUE
------------------------------ ----------- ------------------------------
dg_broker_start                boolean     TRUE
```

```
$ export ORACLE_SID=stdby
$ sqlplus / as sysdba
SQL>  select open_mode from v$database;

OPEN_MODE
--------------------
MOUNTED

SQL> show parameter dg_broker_start

NAME                            TYPE        VALUE
------------------------------ ----------- ------------------------------
dg_broker_start                boolean     FALSE

SQL> alter system set dg_broker_start=TRUE;

System altered.

SQL> show parameter dg_broker_start

NAME                           TYPE        VALUE
------------------------------ ----------- ------------------------------
dg_broker_start                boolean     TRUE
```

e) Invoke DGMGRL and connect as the SYS user. Create a broker configuration named DGConfig1 and include a profile for the primary database.

```
$ export ORACLE_SID=orcl
$ dgmgrl
DGMGRL> connect sys/oracle
Connected.
DGMGRL> create configuration 'orcl.oracle.com' as
>  primary database is 'orcl'
>  connect identifier is orcl;
Configuration "orcl.oracle.com" created with primary database "orcl"
DGMGRL> add database 'stdby' as
> connect identifier is stdby;
Database "stdby" added
DGMGRL> show configuration

Configuration - orcl.oracle.com

  Protection Mode: MaxAvailability
  Databases:
    orcl  - Primary database
    stdby - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
DISABLED
```

f) Enable the configuration and confirm the status.

```
DGMGRL> enable configuration;
Enabled.
DGMGRL> show configuration;

Configuration - orcl.oracle.com

  Protection Mode: MaxAvailability
  Databases:
    orcl  - Primary database
    stdby - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL>
```

g) Invoke DGMGRL and connect as the SYS user. Set the **FastStartFailoverTarget** database property on the primary database and your target standby database.

```
$ export ORACLE_SID=orcl
$ dgmgrl
DGMGRL> connect sys/oracle
Connected.
DGMGRL>  edit database orcl
>  set property FastStartFailoverTarget = stdby;
Property "faststartfailovertarget" updated
DGMGRL>  edit database stdby
> set property FastStartFailoverTarget = orcl;
Property "faststartfailovertarget" updated
DGMGRL>
```

h) Verify that the protection mode is set to maximum availability.

```
DGMGRL> show configuration

Configuration - orcl.oracle.com

  Protection Mode: MaxAvailability
  Databases:
    orcl  - Primary database
    stdby - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

i) Verify that the **LogXptMode** property is set to **SYNC** for the primary database and the physical standby database.

```
DGMGRL> show database orcl LogXptMode
  LogXptMode = 'SYNC'
DGMGRL> show database stdby LogXptMode
  LogXptMode = 'SYNC'
```

j) Enable fast-start failover.

```
DGMGRL> enable fast_start failover;
Enabled.
```

k) Check the status of the configuration by executing the **SHOW CONFIGURATION VERBOSE** command.

```
DGMGRL> show configuration verbose

Configuration - orcl.oracle.com

  Protection Mode: MaxAvailability
  Databases:
    orcl  - Primary database
      Warning: ORA-16819: fast-start failover observer not started

    stdby - (*) Physical standby database
      Warning: ORA-16819: fast-start failover observer not started

  (*) Fast-Start Failover target

Fast-Start Failover: ENABLED

  Threshold:          30 seconds
  Target:             stdby
  Observer:           (none)
  Lag Limit:          30 seconds (not in use)
  Shutdown Primary:   TRUE
  Auto-reinstate:     TRUE

Configuration Status:
WARNING
```

2. Start the observer.

   a) Invoke DGMGRL and connect to your physical standby database. Be sure to set the `ORACLE_SID` environment variable.

```
$ export ORACLE_SID=stdby
$ dgmgrl
DGMGRL> connect sys/oracle
Connected.
```

   b) Start the observer. Remember that control is not returned after starting the observer.

```
DGMGRL> start observer
Observer started
```

   c) Return to your DGMGRL session on your primary database and verify the status of your configuration.

```
DGMGRL> show configuration verbose

Configuration - orcl.oracle.com

  Protection Mode: MaxAvailability
  Databases:
    orcl  - Primary database
    stdby - (*) Physical standby database

  (*) Fast-Start Failover target

Fast-Start Failover: ENABLED

  Threshold:          30 seconds
  Target:             stdby
  Observer:           edaor80p1.us.oracle.com
  Lag Limit:        30 seconds (not in use)
  Shutdown Primary: TRUE
  Auto-reinstate:   TRUE

Configuration Status:
SUCCESS

DGMGRL>
```

d) Use the **SHOW FAST_START FAILOVER** command to further verify your configuration.

```
DGMGRL> show fast_start failover;

Fast-Start Failover: ENABLED

  Threshold:        30 seconds
  Target:           stdby
  Observer:         edaor80p1.us.oracle.com
  Lag Limit:        30 seconds (not in use)
  Shutdown Primary: TRUE
  Auto-reinstate:   TRUE

Configurable Failover Conditions
  Health Conditions:
    Corrupted Controlfile          YES
    Corrupted Dictionary           YES
    Inaccessible Logfile            NO
    Stuck Archiver                  NO
    Datafile Offline               YES

  Oracle Error Conditions:
    (none)

DGMGRL>
```

3. Test fast-start failover by aborting `orcl`.

a) Invoke SQL*Plus and log in to the primary database (`orcl`) as the **SYSDBA** user.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba
```

b) To simulate a failure of the primary database, shut down your primary database instance with the **ABORT** option.

```
SQL> shutdown abort
ORACLE instance shut down.
```

c) Return to the window on your physical standby database system where you started the observer. View the actions. The failover may take a few minutes.

```
13:19:36.21  Friday, March 29, 2013
Initiating Fast-Start Failover to database "stdby"...
```

```
Performing failover NOW, please wait...
Failover succeeded, new primary is "stdby"
13:19:38.56  Friday, March 29, 2013
```

d) Connect to the `orcl` database and execute the `startup mount` command.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba
SQL> startup mount
```

e) Open another terminal window. Set the **`ORACLE_SID`** environment variable to `stdby`. Invoke DGMGRL and connect as the `SYS` user.

```
$ dgmgrl
DGMGRL> connect sys/oracle
Connected.
```

f) Verify the status of your configuration.

```
DGMGRL> show configuration

Configuration - orcl.oracle.com

  Protection Mode: MaxAvailability
  Databases:
    stdby - Primary database
      Warning: ORA-16817: unsynchronized fast-start failover
configuration

    orcl  - (*) Physical standby database (disabled)
      ORA-16661: the standby database needs to be reinstated

Fast-Start Failover: ENABLED

Configuration Status:
WARNING
```

g) Return to the window on your physical standby database system where you started the observer. View the actions. The failover may take a few minutes. If you receive an error message for reinstate, you can issue the `reinstate` command manually.

```
13:21:17.38  Friday, March 29, 2013
Initiating reinstatement for database "orcl"...
Reinstating database "orcl", please wait...
Operation requires shutdown of instance "orcl" on database "orcl"
Shutting down instance "orcl"...
ORA-01109: database not open

Database dismounted.
ORACLE instance shut down.
Operation requires startup of instance "orcl" on database "orcl"
Starting instance "orcl"...
ORACLE instance started.
```

```
Database mounted.
Continuing to reinstate database "orcl" ...
Reinstatement of database "orcl" succeeded
13:22:18.66  Friday, March 29, 2013
```

- Optional step: If an error is encounterd during reinstate, execute the `reinstate` command manually: -- FROM here---

```
DGMGRL> reinstate database 'orcl';
Reinstating database "orcl", please wait...
Reinstatement of database "orcl" succeeded
---------------------------------------------To here--------

DGMGRL> show configuration

Configuration - orcl.oracle.com

  Protection Mode: MaxAvailability
  Databases:
    stdby - Primary database
    orcl  - (*) Physical standby database

Fast-Start Failover: ENABLED

Configuration Status:
SUCCESS
```

4. Perform a switchover to your `orcl` database and stop the observer.

```
DGMGRL> switchover to orcl;
Performing switchover NOW, please wait...
New primary database "orcl" is opening...
Operation requires shutdown of instance "stdby" on database "stdby"
Shutting down instance "stdby"...
ORA-01109: database not open

Database dismounted.
ORACLE instance shut down.
Operation requires startup of instance "stdby" on database "stdby"
Starting instance "stdby"...
ORACLE instance started.
Database mounted.
Switchover succeeded, new primary is "orcl"
DGMGRL> show configuration

Configuration - orcl.oracle.com

  Protection Mode: MaxAvailability
  Databases:
    orcl  - Primary database
    stdby - (*) Physical standby database
```

```
Fast-Start Failover: ENABLED

Configuration Status:
SUCCESS

DGMGRL> stop observer
Done.
DGMGRL> disable fast_start failover;
Disabled.
```

```
Observer stopped
$
```